Article

Review of binomial decomposition-based algorithms for efficient linear complexity computation

Jose Luis Martin-Navarro [1,*] **and Amparo Fúster-Sabater** [2,†]

[1] Aalto University, Aalto, Finland; martinnavarroj@acm.org
[2] Instituto de Tecnologías Físicas y de la Información, C.S.I.C.; amparo@iec.csic.es
* Correspondence: martinnavarroj@acm.org
† Current address: Serrano 144, 28006 Madrid, Spain.

Version February 16, 2021 submitted to Mathematics

Abstract: Lightweight cryptography algorithms are being developed as a response to the ubiquity of IoT. These algorithms have to combine the need of secure primitives with the resource constraints of all the interconnected devices. An essential part of these primitives are the pseudo-random number generators (PRNGs). In fact, the quality of the generators is critical for the security of many cryptographic schemes. Nevertheless, finding which sequence generators have the strongest characteristics is not a straightforward task. In this work, we will review different algorithms based on the binomial decomposition, an innovative technique for linear complexity calculation. They will be tested against a family of sequence generators, which is hard to be analyzed by standard methods. In this way, we can choose the best algorithm to test the security of different binary sequences.

Keywords: PRNG; Binomial Sequences; Complexity; Stream Ciphers; IoT

1. Introduction

Sensorization is not only one of the latest trends that brings a net of communications around us as Internet of Things (IoT), but also it is one of the main requirements for the third technological revolution. Different critical sectors like smart-grid, e-health or industrial automation will increase their dependence on these low-cost devices as well as the growth in dependence will also increase the security risks [1][2].

Ubiquitous devices like IoT are characterized by their constraints on energy consumption, processing power, memory and size, which make harder to keep them secure. Combining their network dependability with their low security features, they become the perfect target for gaining control over the applications and systems behind them [3]. A good example where a vulnerable IoT sensor was used to gain control over the whole system can be found in [4].

Different approaches in research [5], 5G technologies [6] or specific calls such as that of NIST for lightweight cryptography primitives [7] are addressing the security of IoT, taking into account the limited resources available on such devices. Lightweight cryptography and particularly stream ciphers are the keystones on which the different protocols of communication and orchestration are built [8].

In this work, we will introduce the Linear Feedback Shift Registers (LFSRs), key components in stream ciphers, often used as Pseudo Random Number Generators (PRNGs). Among the most recent PRNGs based on shift registers, we can list: the Grain-128AEAD [9] a stream cipher supporting authenticated encryption with associated data that includes both Linear and Nonlinear Shift Registers (LFSR and NFSR, respectively), the TinyJAMBU [9] a family of Lightweight Authenticated Encryption Algorithms whose keyed permutation is based on an 128-bit NLFSR or the Espresso [10] a PRNG for 5G wireless communication systems including a 256-bit LFSR and a 20-variable nonlinear output function.

The two first generators are second-round candidates in the lightweight crypto standardization process launched by NIST.

Next, we will present the generalized shelf shrinking generator, a particular family of ciphers with strong cryptographic characteristics which remain strong to the standard Berlekamp-Massey algorithm [11]. Then, we will improve an innovative sequence decomposition introduced by Cardell et al. in [12] and will show how it can be used to analyze the properties of binary sequences. Finally, we will compare the different algorithms based on the sequence decomposition, including two novel algorithms based on the symmetry of the binomial sequences and the B-representation of binary sequences, respectively.

The study of the generalized shelf shrinking generator is not a random choice. Indeed, it produces not only sequences that are hard to be analyzed by the Berlekamp-Massey algorithm, but also it has been implemented in hardware [13] along on RFID devices [14] and programmable logic devices [15], as a key stream generator. Studying the robustness of these sequences could avoid vulnerabilities on the IoT devices and the services built on them.

The work purpose is to effectively compare the binomial decomposition-based algorithms, showing their strengths and possible use-cases. The first contribution of this work is the experimental study of the number of binomial components in a binomial decomposition (parameter r), which allows us to study the complexity of the BD algorithm. In addition, we present the half-interval search algorithm. Despite it is based on our previous design of the folding algorithm[16], in this work we complete the available knowledge on such an algorithm providing a mathematical proof of its behaviour and correctness. The matrix binomial decomposition algorithm is another novelty of this article, which is based on a recent representation of the generalized self-shrunken sequences [17]. Finally, after completing the gaps on the algorithm definitions, the last contribution of our work is the comparison among all the previous algorithms and the discussion about their different use-cases.

The paper is organized as follows. Section 2 includes a brief revision of LFSRs and sequence generators based on irregular decimation, a well-known kind of generators including the generalized self-shrinking generator. Section 3 describes the characteristics and generalities of the binomial sequences, binary sequences that constitute the foundations of the algorithms above mentioned. Section 4 introduces and analyzes four algorithms to calculate the linear complexity of binary sequences: (a) the standard Berlekamp–Massey algorithm, (b) the binomial decomposition BS-algorithm, an improved version of the algorithm developed in [12] that analyzes different properties of the binary sequences, (c) the half-interval search algorithm, a novel proposal based on the symmetry of the binomial sequences and d) the matrix binomial decomposition or m-BD algorithm based on the product of matrices. Section 5 includes the discussion and extensively comparison among the four previous algorithms, including experiments that test their performance. Finally, conclusions and future research lines in Section 6 end the paper.

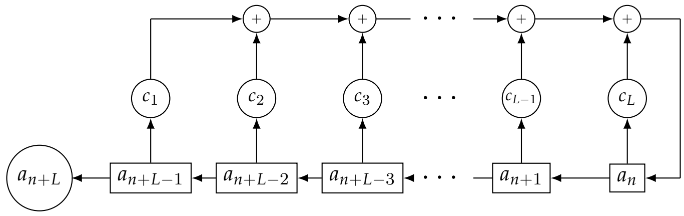
2. Shift registers and the concept of linear complexity

Pseudo-random binary sequences have extensive applications in secure communications, e.g. wireless systems, cryptography, error-correcting codes or circuit testing. Commonly used structures for the generation of such sequences are the Linear Feedback Shift Registers (LFSRs) [18]. In fact, LFSRs are essential components in the design of many sequence generators found in the literature. Good reliability, high speed and easy implementation are some of their practical advantages, which justify a so wide and generalized use. From a theoretical point of view, LFSRs are mathematical models readily analyzable by means of algebraic methods [18].

According to Figure 1, an LFSR is made up of the following components:

1. L binary stages, which are interconnected and numbered $(0, 1, 2, \ldots, L-1)$ from left to right. Each stage stores a unique bit.

Figure 1. A scheme of LFSR with L stages



2. The L-degree feedback or connection polynomial

$$p(x) = x^L + c_1 x^{L-1} + c_2 x^{L-2} + \ldots + c_{L-1} x + c_L$$

with coefficients c_i defined in the binary field $c_i \in \mathbb{F}_2$.
3. A non-zero initial state (stage contents) at the initial instant.

In brief, LFSRs generate sequences by means of successive linear feedbacks and shifts.

The output sequence of an LFSR is a binary sequence $\{a_n\}$ $(n = 0, 1, 2, \ldots)$ with $a_n \in \mathbb{F}_2$. When the polynomial $p(x)$ is a primitive polynomial [18], then the output sequence is a PN-sequence (or Pseudo-Noise sequence); besides, a PN-sequence has length $l = 2^L - 1$ bits where 2^{L-1} of them are ones and $2^{L-1} - 1$ are zeros.

The idea of pseudo-randomness in sequences of finite length implies the difficulty of predicting the subsequent digits of a sequence from the knowledge of the previous ones. A measure of unpredictability is the parameter linear complexity, notated LC. Roughly speaking, LC is related with the amount of sequence we need to process in order to recover all the sequence. In terms of security, this amount has to be as large as possible; the recommended value is half the length of the sequence.

The concept of linear complexity of a sequence is closely related to LFSRs. The formal definition of LC is now introduced: The linear complexity of a binary sequence $\{s_n\}$ $(n = 0, 1, 2, \ldots)$ with $s_n \in \mathbb{F}_2$ is the length of the shortest LFSR able to generate such a sequence. By definition, the LC of a PN-sequence generated by a LFSR with L stages is $LC = L$.

Although LFSRs are in themselves excellent generators of pseudo-random sequence, they are essentially linear structures. This is the reason why any kind of non-linearity must be introduced in the process of generation. Non-linear filters, clock-controlled generators, combination generators or dynamic LFSR-based generators are just some of the habitual examples of sequence generators involving non-linearity, see [19,20] and the references cited therein. Particular attention deserves the irregular decimation of PN-sequences as an efficient technique to erase the linearity inherent to LFSRs [21,22]. Among the different examples of decimation-based generators we can enumerate: 1) the shrinking generator [23] with two LFSRs for a mutual decimation, 2) the self-shrinking generator [24] with just one LFSR that decimates itself and 3) the generalized self-shrinking generator [25] that outputs a family of pseudo-random sequences, the so-called generalized self-shrunken sequences (GSS-sequences). Different cryptanalytic attacks against the previous generators can be found in the literature [26–30].

In this work, we focus on binary sequences whose length is a power of 2, characteristic exhibited by many of the sequences from the previous generators.

2.1. An LFSR-based sequence generator

A characteristic design of LFSR-based sequence generator is the generalized self-shrinking generator (GSSG). In fact, it is the most representative element in the class of decimation-based generators as well as a practical design with application in low-cost passive RFID tags, see [14].

Table 1. Family of generalized sequences for $p(x) = x^3 + x + 1$

p-rotation	$\{b_n\}$ sequences	GSS-sequences
0	**1**0**111**00	1111
1	**0**1**110**01	0110
2	**1**1**100**10	1100
3	**1**1**001**01	1001
4	**1**0**010**11	1010
5	**0**0**101**11	0101
6	**0**1**011**10	0011
PN-sequence	**1**0**111**00	

A GSSG consists of:

a) A PN-sequences $\{a_n\}$ generated by an L-stage LFSR and a shifted version of such a sequence, notated $\{b_n\}$. Both sequences are related by the expression $\{b_n\} = \{a_{n+p}\}$, p being an integer. Thus, $\{b_n\}$ is nothing but the PN-sequence $\{a_n\}$ circularly rotated p positions with $(p = 0, 1, 2 \ldots, 2^L - 2)$.

b) A simple decimation rule defined as:

$$\begin{cases} \text{If } a_n = 1 \text{ then } b_n \text{ is output,} \\ \text{If } a_n = 0 \text{ then } b_n \text{ is discarded and no bit is output.} \end{cases}$$

For every p, a new sequence $\{u_n\}_p = \{u_0, u_1, u_2, \ldots\}_p$ is generated. Each sequence $\{u_n\}_p$ is called the generalized self-shrunken sequence associated with the rotation p. When p ranges in the interval $[0, 1, \ldots, 2^L - 2]$, then we obtain all the elements of the family of GSS-sequences (in total $2^L - 1$ elements) based on the PN-sequence $\{a_n\}$.

Some important facts essentially extracted from [25] are enumerated:

1. All the generalized self-shrunken sequences are balanced apart from the identically 1 sequence [25, Theorem 1].
2. By construction, the family of generalized self-shrunken sequences consists of $2^L - 1$ sequences of 2^{L-1} bits each of them. Thus, the length of any generalized sequence will be 2^{L-1} or divisors. At any rate, the length of these sequences will always be a power of 2.
3. The family of generalized sequences plus the identically null sequence has structure of Abelian group where the group operation is the bit-wise sum mod 2. The neutral element is the identically null sequence and every sequence is its own inverse element [25, Theorem 2].
4. The sequence produced by the self-shrinking generator is a member of this family for $p = 2^{L-1}$, see [22].

Moreover, we can add that the LC of every GSS-sequence is upper-bounded by $2^{L-1} - (L - 2)$ [31, Theorem 2]. A simple example of GSS-sequences is next introduced.

Example 1. *With a LFSR whose primitive polynomial is $p(x) = x^3 + x + 1$ and initial state $(1, 0, 1)$, we can generate the GSS-sequences depicted in Table 1. Bits in bold in the sequences $\{b_n\}$ represent the digits of the corresponding GSS-sequence associated with the rotation p. The PN-sequence $\{a_n\}$ with length $l = 2^3 - 1$ and ones in bold appears at the bottom of the table.*

3. Binomial sequences

A new representation of binary sequences in terms of the so-called binomial sequences is now introduced. Such a representation applies only to sequences whose length is a power of 2. Next, we analyze the representation of the GSS-sequences by means of binomial sequences.

3.1. Introduction to binomial sequences

The binomial number $\binom{n}{k}$ (n, k being non-negative integers) is the coefficient of the power x^k in the expansion of the binomial power $(1+x)^n$. For $n \geq 0$, it is a well-known fact that $\binom{n}{0} = 1$ while $\binom{n}{k} = 0$ for all $k > n$.

From the binomial coefficients reduced modulo 2, the concept of binomial sequence is defined as follows: The *k-th* binomial sequence $\{\binom{n}{k}\}$ ($n = 0, 1, 2, \ldots$) is a binary sequence whose elements are binomial coefficients $\binom{n}{k}$ reduced modulo 2, that is

$$\left\{\binom{n}{k}\right\}_{n\geq 0} = \left\{\binom{0}{k}, \binom{1}{k}, \binom{2}{k}, \ldots\right\}_{mod\ 2},$$

where k is called the index of the binomial sequence. The k first terms of the binomial sequence are zeros while the term $\binom{k}{k}$ corresponds to the first 1.

Table 2. Binomial sequences with their lengths l_k and linear complexities LC_k

Binom. coeff.	Binomial sequences $\{\binom{n}{k}\}$	Length	Linear complexity
$\binom{n}{0}$	$\{1,1,1,1,1,1,1,1,\ldots\}$	$l_0 = 1$	$LC_0 = 1$
$\binom{n}{1}$	$\{0,1,0,1,0,1,0,1,\ldots\}$	$l_1 = 2$	$LC_1 = 2$
$\binom{n}{2}$	$\{0,0,1,1,0,0,1,1,\ldots\}$	$l_2 = 4$	$LC_2 = 3$
$\binom{n}{3}$	$\{0,0,0,1,0,0,0,1,\ldots\}$	$l_3 = 4$	$LC_3 = 4$
$\binom{n}{4}$	$\{0,0,0,0,1,1,1,1,\ldots\}$	$l_4 = 8$	$LC_4 = 5$
$\binom{n}{5}$	$\{0,0,0,0,0,1,0,1,\ldots\}$	$l_5 = 8$	$LC_5 = 6$
$\binom{n}{6}$	$\{0,0,0,0,0,0,1,1,\ldots\}$	$l_6 = 8$	$LC_6 = 7$
$\binom{n}{7}$	$\{0,0,0,0,0,0,0,1,\ldots\}$	$l_7 = 8$	$LC_7 = 8$

Table 2 shows the binomial sequences $\{\binom{n}{k}\}$ ($k = 0, 1, \ldots, 7$), with their lengths l_k and linear complexities LC_k, see [?].

Different properties of the binomial sequences are next enumerated.

1. Given the binomial sequence $\{\binom{n}{k}\}$ with $k = 2^m + i$ where m is a non-negative integer and the index i takes values in the interval $0 \leq i < 2^m$, then we have that [12, Proposition 3]:
    a) The binomial sequence $\{\binom{n}{k}\}$ has length $l = 2^{m+1}$.
    b) The formation rule of this binomial sequence is:

$$\left\{\binom{n}{2^m+i}\right\}_{0\leq n<2^{m+1}} = \begin{cases} 0 & \text{if} \quad 0 \leq n < 2^m + i, \\ \binom{n}{i}_{mod\,2} & \text{if} \quad 2^m + i \leq n < 2^{m+1}. \end{cases}$$

2. The linear complexity of the binomial sequence $\{\binom{n}{2^m+i}\}$ with m and i defined as above is $LC = 2^m + i + 1$, see [12, Theorem 13].
3. Every binary sequence $\{s_n\}_{n\geq 0}$ whose length is a power of 2 can be written as linear combination of binomial sequences [12, Theorem 2]. This combination is called the Binomial Decomposition of $\{s_n\}_{n\geq 0}$. Such a decomposition allows us to analyze fundamental properties of the sequence, e.g. length and linear complexity.
4. Given a sequence $\{s_n\}_{n\geq 0}$ with binomial decomposition $\{s_n\} = \sum_{i=1}^{r} \left\{\binom{n}{k_i}\right\}$, where $0 \leq k_1 < k_2 < \cdots < k_r$ are integer indices, then its linear complexity is given by $LC = k_r + 1$, see [12, Corollary 14].
5. Given a sequence $\{s_n\}_{n\geq 0}$ with binomial decomposition $\{s_n\} = \sum_{i=1}^{r} \left\{\binom{n}{k_i}\right\}$, where $0 \leq k_1 < k_2 < \cdots < k_r$ are integer indices, then its length l is that of the binomial sequence $\left\{\binom{n}{k_r}\right\}$, that is the length of the binomial sequence of maximum index in its binomial decomposition, see [? , Theorem 1].

All these properties will be used in the algorithms that compute the LC of every binary sequence $\{s_n\}_{n\geq 0}$.

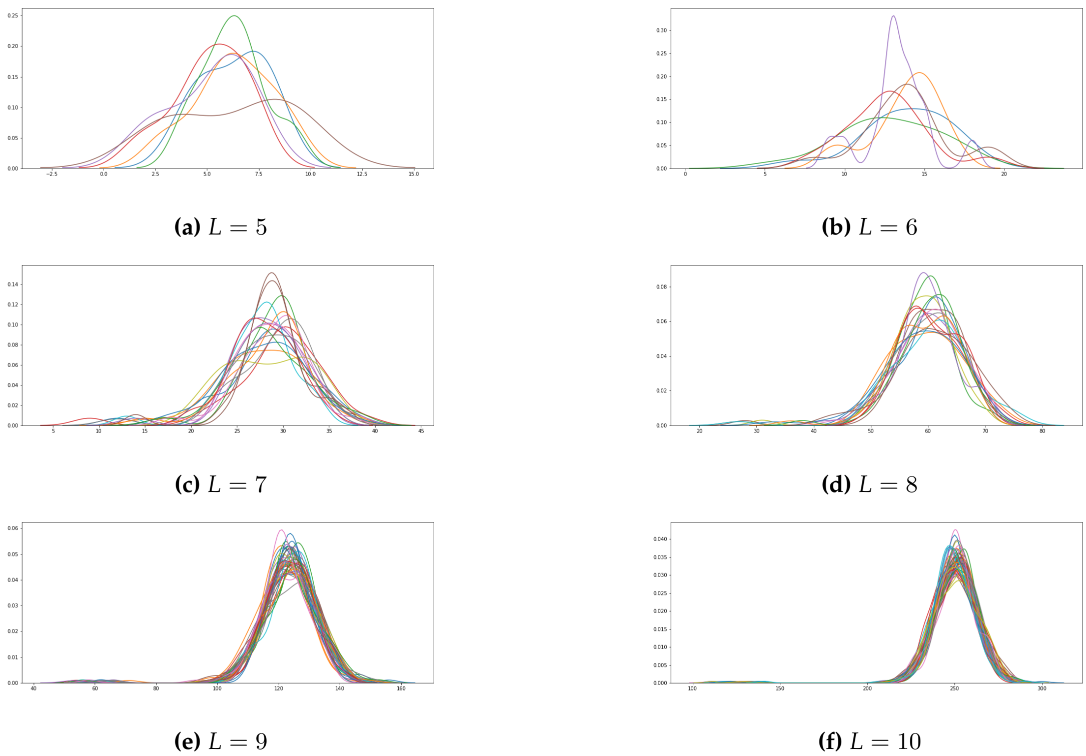
In addition, the binomial sequences can be found in the diagonals of the Sierpinski's triangle reduced modulo 2 [12, Section 4] as well as in certain linear cellular automata (e.g. linear automata with rules 102 and 60) as it has been studied in [22, Chapter 3]. See the previous references for more details.

3.2. Binomial decomposition of GSS-sequences

The number of binomial sequences, notated r, in the decomposition of any GSS-sequence has not been previously analyzed in the literature. The parameter is decisive in the comparison among the algorithms of Section 4, since the BD-algorithm complexity depends on the number of binomial sequences. In order to study the asymptotic behavior of this parameter, some experiments were carried out.

The analyzed sequences in such experiments were all the GSS-sequences coming from LFSRs with primitive feedback polynomials of degree L with L taking values in the interval $[5, 10]$. More precisely, we have considered the 6 primitive polynomials of degree 5, the 6 primitive polynomials of degree 6, the 18 primitive polynomials of degree 7, the 16 primitive polynomials of degree 8, the 48 primitive polynomials of degree 9 and the 60 primitive polynomials of degree 10. For each one of these primitive polynomials, the $2^L - 1$ GSS-sequences have been generated and decomposed in terms of their binomial sequences. On average, we observed a number of binomial sequences given by 2^{L-2}, $\forall L \in [5, 10]$.

Figure 2. Density of binomial sequences in the GSS-sequence decomposition



(a) $L = 5$ **(b)** $L = 6$ **(c)** $L = 7$ **(d)** $L = 8$ **(e)** $L = 9$ **(f)** $L = 10$

The plots corresponding to the number of binomial sequences in the decomposition of all these GSS-sequences are depicted in the Figure 2. For each chart, the x-axis represents the number of binomial sequences in a specific decomposition (parameter r) while the y-axis counts the number of times r occurs. For a given LFSR, each one of the colors represents all the sequences of the GSS-family generated by such an LFSR. In brief, for each value of L the chart represents the distribution of the parameter r for all the GSS-sequences generated by primitive polynomials of degree L.

The distribution of the number of binomial sequences in the GSS-sequences follows closely a normal distribution. Nevertheless, a smooth tail can be also noticed on the left of the figures, which means that for some GSS-sequences the density of binomial sequences will be lower.

The results of these experiments will be employed in some of the algorithms to compute the LC described in next section.

4. Different algorithms to compute the linear complexity of a sequence

In this section, we introduce different algorithms (both novel and already known algorithms) to compute the LC of any binary sequence with length $l = 2^m$, m being a non-negative integer. Analysis, foundations and characteristics of each algorithm are described in the subsequent sections.

Throughout next sections, the following notation will be systematically used.

1. For the sake of readability, in the sequel the binomial coefficient $\binom{n}{k}$ just denotes the *k-th* binomial sequence.
2. The term $\binom{n}{k}_{i,j}$ represents the sub-sequence of $\binom{n}{k}$ between the *i-th* and *j-th* bits.
3. The term $\binom{n}{k}_j$ stands for the sub-sequence corresponding to the j first bits of $\binom{n}{k}$.

4.1. Berlekamp-Massey algorithm

The most general and well-known method of computing the linear complexity of binary sequences is the Berlekamp-Massey algorithm [11]. Such an algorithm can be applied to sequences of any length, not only to sequences whose length is a power of 2. For a fixed binary sequence, this algorithm processes bit-by-bit the successive digits until it finds the shortest LFSR able to generate the whole sequence. At each particular step, the Berlekamp-Massey algorithm computes the length and the feedback polynomial of the shortest LFSR that produces the sub-sequence analyzed up to that particular bit. Both LFSR length and feedback polynomial degree will always be greater than those of the previous step.

In order to get the final value of LC, this algorithm has to process a number of bits equal to twice the value of the linear complexity of the sequence under consideration. For sequences whose LC is close to their length l, e.g. the GSS-sequences [22], the Berlekamp-Massey algorithm will process approximately $2 * l$ bits of each sequence with a computational complexity of $O(l^2)$, see [32].

4.2. Binomial decomposition algorithm or BD-algorithm

In order to compute the LC of a given sequence, the BD-algorithm [12] provides one with a simple procedure to determine the binomial decomposition of such a sequence. The mathematical results enumerated in the sub-section 3.1 constitute the core of this algorithm. More precisely, two properties are taken into account:

- According to Item 3 (in sub-section 3.1), the sequence *seq* of length $l = 2^m$ can be decomposed into r binomial sequences of the form:

$$seq = \binom{n}{k_1} + \cdots + \binom{n}{k_r}.$$

- According to Item 4 (in sub-section 3.1), the lineal complexity of *seq* is that of the binomial sequence of maximum index $\binom{n}{k_r}$ in its binomial decomposition. Since the indices of the binomial sequences are written in increasing order, then LC is computed by means of the following equation:

$$LC = k_r + 1. \tag{1}$$

Algorithm 1 : The BD-algorithm

Input: seq: the sequence to be analyzed
 $binom = [\emptyset], k_r = 0$
 for $i = 0;\ i < length(seq);\ i{+}{+}$ **do**
  if $seq_i == 1$ **then**
   $seq + = \binom{n}{i}$
   $binom.add(i)$
   $k_r = i$
  end if
 end for
 Output: $binom$ and $LC = k_r + 1$: binomial decomposition and LC of seq.

Table 3. A step-by-step application of the BD-algorithm to seq_{16}

Step	Ope.	seq	Bit position 0	4	8	12
1		seq	0 0 0 1	1 1 0 1	1 0 0 0	1 0 1 1
	+	$\binom{n}{3}$	0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1
2	=	seq	0 0 0 0	1 1 0 0	1 0 0 1	1 0 1 0
	+	$\binom{n}{4}$	0 0 0 0	1 1 1 1	0 0 0 0	1 1 1 1
3	=	seq	0 0 0 0	0 0 1 1	1 0 0 1	0 1 0 1
	+	$\binom{n}{6}$	0 0 0 0	0 0 1 1	0 0 0 0	0 0 1 1
4	=	seq	0 0 0 0	0 0 0 0	1 0 0 1	0 1 1 0
	+	$\binom{n}{8}$	0 0 0 0	0 0 0 0	1 1 1 1	1 1 1 1
5	=	seq	0 0 0 0	0 0 0 0	0 1 1 0	1 0 0 1
	+	$\binom{n}{9}$	0 0 0 0	0 0 0 0	0 1 0 1	0 1 0 1
6	=	seq	0 0 0 0	0 0 0 0	0 0 1 1	1 1 0 0
	+	$\binom{n}{10}$	0 0 0 0	0 0 0 0	0 0 1 1	0 0 1 1
7	=	seq	0 0 0 0	0 0 0 0	0 0 0 0	1 1 1 1
	+	$\binom{n}{12}$	0 0 0 0	0 0 0 0	0 0 0 0	1 1 1 1
end	=	seq	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
$seq = \binom{n}{3} + \binom{n}{4} + \binom{n}{6} + \binom{n}{8} + \binom{n}{9} + \binom{n}{10} + \binom{n}{12}$ $LC = k_r + 1 = 12 + 1 = 13$						

The result of the previous properties is the Algorithm 1. Indeed, it takes as input the sequence seq and checks for the bits that equal 1. If $seq_i = 1$, then it bit-wise sums the sequence seq with the binomial sequence $\binom{n}{i}$, so that $seq = seq + \binom{n}{i}$. The procedure stops when all the binomial sequences in the decomposition have been determined or, equivalently, when the resulting sequence seq is the identically null sequence. The algorithm outputs the binomial decomposition of the sequence under consideration as well as the value of its LC, via the equation (1).

A step-by-step application of Algorithm 1 to the binomial decomposition of $seq_{16} = \{0001110110001011\}$ with $l = 2^4$ is depicted in Table 3. Recall that the BD-algorithm computes LC after processing 13 bits of seq_{16} while the Berlekamp-Massey algorithm needs $2 * 13 = 26$ bits. In fact, the BD-algorithm performs the bit-wise sum of two sequences of l bits, that is l operations, for each binomial sequence that appears in the binomial decomposition. Thus, its computational complexity is $O(r * l)$, where r is the number of binomial sequences in the decomposition of the analyzed sequence with $r \ll l$.

Next, we show how the BD-algorithm can be improved and its complexity reduced.

4.2.1. Improvement of the BD-algorithm:

If we avoid the sum of the sub-sequences identically null, then the performance of this algorithm clearly improved. Due to the properties of the binomial coefficients described in sub-section 3.1, we know that $\binom{n}{k} = 0$ for all $n < k$. At the same time, notice that at the *i-th* step of the algorithm the k_i first terms of *seq* are zeros.

Therefore, combining these two facts the number of operations is substantially reduced. When the first 1 in the *i-th* position of *seq* is detected, then the algorithm bit-wise sums both sequences exclusively between the *i-th* and $(l-1)$-*th* bits, that is $(seq_{i,l-1} + \binom{n}{i}_{i,l-1})$, as the headers of both sequences (until the $(i-1)$-*th* bit) are zeros.

In this way, the number of additions at each step is incrementally reduced:

$$\sum_{i=1}^{r}(l - k_i) < r * l.$$

Moreover, for sequences whose LC is upper bounded the algorithm performance can be even improved. In fact, in that case we do not need to check any other bit after the index corresponding to this upper bound. For example, every sequence produced by a generalized self-shrinking generator with LFSR of length L has a LC upper bounded by $LC_{max} = 2^{L-1} - (L-2)$, see [31]. In that case, the maximum index k_{max} in its binomial decomposition is $k_{max} = l - \log l$, $l = 2^{L-1}$ being the sequence length. Hence, the final number of operations is again reduced to:

$$\sum_{i=1}^{r}(k_{max} - k_i) < \sum_{i=1}^{r}(l - k_i) < r * l.$$

The code of Algorithm 1 is just upgraded by converting the bit-wise sum of both sequences into the expression $seq = seq_{i,k_{max}} + \binom{n}{i}_{i,k_{max}}$, with k_{max} defined as before.

In brief, for this family of sequences the BD-algorithm requires $l - \log l$ bits of each sequence to compute its LC with a computational complexity less than $O(r * l)$.

4.3. *Half-interval search algorithm*

In this sub-section a novel algorithm to compute the LC, the so-called half-interval search algorithm, is described. Such an algorithm takes full advantage of the binomial sequence symmetry. A preliminary version of this algorithm by the same authors was introduced in [16,33]. First of all, we study the symmetry properties of the binomial sequences.

4.3.1. Symmetry of the binomial sequences:

In fact, the symmetry of these sequences gives rise to the following results.

Theorem 1. *Let $\binom{n}{k}_l$ denote the l first bits of the binomial sequence $\binom{n}{k}$ with $l = 2^m$, m being a positive integer. Such a sub-sequence can be divided into two new sub-sequences of length $\frac{l}{2}$:*

$$\binom{n}{k}_l = \left(\binom{n}{k}_{0,\frac{l}{2}-1}, \binom{n}{k}_{\frac{l}{2},l-1}\right), \tag{2}$$

then, two different configurations may appear:

1. *If k the index of the binomial sequence is $k < \frac{l}{2}$, then the two sub-sequences in equation (2) are equal.*
2. *If k the index of the binomial sequence is $k \geq \frac{l}{2}$, then the two sub-sequences in equation (2) are written as:*

$$\binom{n}{k}_l = \left(zeros_{\frac{l}{2}}, \binom{n}{i}_{\frac{l}{2}}\right), \tag{3}$$

where $zeros_{\frac{l}{2}}$ represents the sub-sequence identically null of length $\frac{l}{2}$ and i is an integer satisfying $0 \leq i < 2^{m-1}$.

Proof. Both cases are proved separately.

1. Since $k < \frac{l}{2}$, then k can be written as $k = 2^j + i$, where j and i are non-negative integers such that $j < m - 1$ and $0 \leq i < 2^j$. According to Item 1(a) in sub-section 3.1, the binomial sequence $\binom{n}{k} = \binom{n}{2^j+i}$ has length $\tilde{l} = 2^{j+1}$ where the maximum length is $\tilde{l}_{max} = 2^{m-1}$ when $j = m-2$ and the minimum length $\tilde{l}_{min} = 2^0$ when $j = 0$. At any rate, $\tilde{l}$ is a power of 2 as well as $\tilde{l} < 2^m$ and, therefore, the first and second sub-sequences in equation (2) are equal.
2. Since $k \geq \frac{l}{2} = 2^{m-1}$, then k can be written as $k = 2^{m-1} + i$ with $0 \leq i < 2^{m-1}$. According to Item 1(a) in sub-section 3.1, the binomial sequence $\binom{n}{k} = \binom{n}{2^{m-1}+i}$ has length $\tilde{l} = l = 2^m$. Moreover, according to Item 1(b) in sub-section 3.1

$$\binom{n}{k}_{\frac{l}{2},l-1} = \binom{n}{2^{m-1}+i}_{\frac{l}{2},l-1} = \binom{n}{i}_{\frac{l}{2}}.$$

Thus, the sub-sequence $\binom{n}{k}_l$ satisfies the equation (3) as well as the $\frac{l}{2}$ first terms are zeros.

□

Table 4. Theorem 1 applied to the binomial decomposition of seq_{16}

seq	0 0 0 1	1 1 0 1	1 0 0 0	1 0 1 1
$\binom{n}{3}_l = (\binom{n}{3}_{\frac{l}{2}}, \binom{n}{3}_{\frac{l}{2}})$	0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1
$\binom{n}{4}_l = (\binom{n}{4}_{\frac{l}{2}}, \binom{n}{4}_{\frac{l}{2}})$	0 0 0 0	1 1 1 1	0 0 0 0	1 1 1 1
$\binom{n}{6}_l = (\binom{n}{6}_{\frac{l}{2}}, \binom{n}{6}_{\frac{l}{2}})$	0 0 0 0	0 0 1 1	0 0 0 0	0 0 1 1
$\binom{n}{8}_l = (zeros_{\frac{l}{2}}, \binom{n}{0}_{\frac{l}{2}})$	0 0 0 0	0 0 0 0	1 1 1 1	1 1 1 1
$\binom{n}{9}_l = (zeros_{\frac{l}{2}}, \binom{n}{1}_{\frac{l}{2}})$	0 0 0 0	0 0 0 0	0 1 0 1	0 1 0 1
$\binom{n}{10}_l = (zeros_{\frac{l}{2}}, \binom{n}{2}_{\frac{l}{2}})$	0 0 0 0	0 0 0 0	0 0 1 1	0 0 1 1
$\binom{n}{12}_l = (zeros_{\frac{l}{2}}, \binom{n}{4}_{\frac{l}{2}})$	0 0 0 0	0 0 0 0	0 0 0 0	1 1 1 1
$seq_{16} = \binom{n}{3} + \binom{n}{4} + \binom{n}{6} + \binom{n}{8} + \binom{n}{9} + \binom{n}{10} + \binom{n}{12}$				

In Table 4, where $\frac{l}{2} = 8$, the binomial sequences $\binom{n}{3}$, $\binom{n}{4}$ and $\binom{n}{6}$ correspond to the condition 1) in Theorem 1, where the eight first bits are repeated, while the binomial sequences $\binom{n}{8}$, $\binom{n}{9}$, $\binom{n}{10}$ and $\binom{n}{12}$ correspond to the condition 2) in the same theorem with $k \geq 8$.

Next result introduces an interesting characteristic of the sub-sequence $\binom{n}{k}_{\frac{l}{2},l-1}$, which can be converted into another binomial sequence.

Proposition 1. *The sub-sequence $\binom{n}{k}_{\frac{l}{2},l-1}$ that is the second sub-sequence of $\binom{n}{k}_l$ in equation (2) with $k \geq \frac{l}{2}$ can be written as:*

$$\binom{n}{k}_{\frac{l}{2},l-1} = \binom{n}{k-\frac{l}{2}}_{\frac{l}{2}}.$$

Proof. According to the previous properties of the binomial sequences, we write:

$$\binom{n}{k}_{\frac{l}{2},l-1} = \binom{n}{2^{m-1}+i}_{\frac{l}{2},l-1} = \binom{n}{i}_{\frac{l}{2}} = \binom{n}{k-2^{m-1}}_{\frac{l}{2}} = \binom{n}{k-\frac{l}{2}}_{\frac{l}{2}}.$$

□

This will be the notation used in the sequel.

The sub-sequences $\binom{n}{k}_l$ can be classified into two disjoint sets depending on the value of the index k, as explained in Algorithm 2. In the first case, only the first half of the sub-sequence must be computed ($0 \leq n < \frac{l}{2}$) as the second half is exactly the same. In the second case, it is precisely the second half of the sub-sequence which has to be computed ($\frac{l}{2} \leq n < l$), since the $\frac{l}{2}$ first bits are zeros.

Algorithm 2 : Classification of the binomial sequences

Given the sub-sequence $\binom{n}{k}_l$:
if $k < \frac{l}{2}$ **then**
$\binom{n}{k}_l := (\binom{n}{k}_{\frac{l}{2}}, \binom{n}{k}_{\frac{l}{2}})$
else
$\binom{n}{k}_l := (zeros_{\frac{l}{2}}, \binom{n}{k-\frac{l}{2}}_{\frac{l}{2}})$
end if

According to the previous classification, a matrix representation of the binomial decomposition is now introduced:

$$\begin{pmatrix} \binom{n}{k_1} \\ \vdots \\ \binom{n}{k_r} \end{pmatrix} = \left(\begin{array}{c} \binom{n}{k_1} \\ \vdots \\ \binom{n}{k_{i-1}} \\ \hline \binom{n}{k_i} \\ \vdots \\ \binom{n}{k_r} \end{array}\right)_{k_{i-1} < \frac{l}{2} \leq k_i} = \left(\begin{array}{c|c} \binom{n}{k_1}_{\frac{l}{2}} & \binom{n}{k_1}_{\frac{l}{2}} \\ \vdots & \vdots \\ \binom{n}{k_{i-1}}_{\frac{l}{2}} & \binom{n}{k_{i-1}}_{\frac{l}{2}} \\ \hline zeros_{\frac{l}{2}} & \binom{n}{k_i - \frac{l}{2}}_{\frac{l}{2}} \\ \vdots & \vdots \\ zeros_{\frac{l}{2}} & \binom{n}{k_r - \frac{l}{2}}_{\frac{l}{2}} \end{array}\right) = \left(\begin{array}{c|c} M_0 & M_1 \\ \hline M_2 & M_3 \end{array}\right). \quad (4)$$

The different sub-matrices of the matrix representation in (4) are described as follows:

- M_0 and M_1 are $((i-1) \times \frac{l}{2})$ sub-matrices that, according to Theorem 1, satisfy the equality $M_0 = M_1$.
- M_2 is the $((r-i+1) \times \frac{l}{2})$ identically null sub-matrix.
- M_3 is the $((r-i+1) \times \frac{l}{2})$ sub-matrix representing the decomposition of a new sequence of length $\frac{l}{2}$ coming from the bit-wise sum of the two halves of *seq*. Therefore, from M_3 the matrix representation can be extended recursively.

$$\left(\begin{array}{c|c} M_0 & M_1 \\ \hline M_2 & M_3 \end{array}\right) = \left(\begin{array}{c|c} M_0 & M_1 \\ \hline M_2 & \begin{array}{c|c} M_{3,0} & M_{3,1} \\ \hline M_{3,2} & M_{3,3} \end{array} \end{array}\right) = \left(\begin{array}{c|c} M_0 & M_1 \\ \hline M_2 & \begin{array}{c|c} M_{3,0} & M_{3,1} \\ \hline M_{3,2} & \begin{array}{c|c} M_{3,3,0} & M_{3,3,1} \\ \hline M_{3,3,2} & M_{3,3,3} \end{array} \end{array} \end{array}\right) = \ldots \quad (5)$$

In fact, take M_3 and repeat the same process until the length of the resulting sequence equals 1 and, consequently, the sequence cannot be divided anymore.

Thus, the half-interval search algorithm takes fully advantage of the symmetry properties of the binomial sequences and reduces recursively the length of the sequence to be analyzed, see equation (5).

A numerical example of the matrix representation is next introduced.

Example 2. *For the sequence* $seq_{16} = \{0001110110001011\}$*, the matrix representation of its binomial decomposition is:*

$$\begin{pmatrix} \binom{n}{3} \\ \binom{n}{4} \\ \binom{n}{6} \\ \binom{n}{8} \\ \binom{n}{9} \\ \binom{n}{10} \\ \binom{n}{12} \end{pmatrix} = \left(\begin{array}{cc|cc} 0001 & 0001 & 0001 & 0001 \\ 0000 & 1111 & 0000 & 1111 \\ 0000 & 0011 & 0000 & 0011 \\ \hline 0000 & 0000 & 1111 & 1111 \\ 0000 & 0000 & 0101 & 0101 \\ 0000 & 0000 & 0011 & 0011 \\ 0000 & 0000 & 0000 & 1111 \end{array}\right) = \left(\begin{array}{c|c} M_0 & M_1 \\ \hline M_2 & M_3 \end{array}\right),$$

where

$$M_3 = \left(\begin{array}{c|c} M_{3,0} & M_{3,1} \\ \hline M_{3,2} & M_{3,3} \end{array}\right) = \left(\begin{array}{c|c} 1111 & 1111 \\ 0101 & 0101 \\ 0011 & 0011 \\ \hline 0000 & 1111 \end{array}\right),$$

and

$$M_{3,3} = \left(\begin{array}{c|c} M_{3,3,0} & M_{3,3,1} \\ \hline M_{3,3,2} & M_{3,3,3} \end{array}\right) = \left(\begin{array}{c|c} 11 & 11 \\ \hline \emptyset & \emptyset \end{array}\right).$$

When the two halves of seq are bit-wise summed, then the binomial sequences $\binom{n}{3}$, $\binom{n}{4}$ *and* $\binom{n}{6}$ *with repeated sub-sequences are cancelled. Thus, we have a new seq of length* $\frac{l}{2} = 8$ *including the binomial sequences* $\binom{n}{8}$, $\binom{n}{9}$, $\binom{n}{10}$ *and* $\binom{n}{12}$. *When the two halves of the resulting seq are bit-wise summed again, then we have a new seq of length* $\frac{l}{4} = 4$ *and the binomial sequences* $\binom{n}{8}$, $\binom{n}{9}$ *and* $\binom{n}{10}$ *with repeated sub-sequences are cancelled. The only resulting binomial sequence is* $\binom{n}{12}$ *what means that* $LC = 12 + 1$.

4.3.2. Description of the half-interval search algorithm:

From the symmetry properties of the binomial sequences, the half-interval search algorithm locates the binomial sequence of maximum index to compute the LC. At each step, it bit-wise sums both halves of the sequence. If the result is different from zero, then it performs the same procedure with the resulting sequence. Otherwise, it takes half the sequence obtained in the previous step to apply the same procedure. When only one bit is left the algorithm stops.

The pseudo-code of the algorithm, for a given binary sequence of length $l = 2^m$ can be found in Algorithm 3.

Algorithm 3 : The half-interval search algorithm

```
Input: seq: sequence to be analyzed
  k = 0
  while length(seq) > 1 do
    l = length(seq)
    sum = seq_{0, l/2 - 1} + seq_{l/2, l-1}
    if sum ≠ 0_{l/2} then
      seq = sum
      k+ = l/2
    else
      seq = seq_{0, l/2 - 1}
    end if
  end while
  Output: k: maximum index k and LC of seq.
```

At every step, the algorithm reduces by 2 the length of *seq*. The total number of steps is $\log l$ and the total number of operations for a sequence *seq* with length $l = 2^m$ is:

$$\frac{l}{2} + \frac{l}{4} + \frac{l}{8} + \frac{l}{16} + \cdots = \sum_{i=1}^{\log l} \frac{l}{2^i} \approx l$$

Next, an example of how the half-interval algorithm works is introduced.

Example 3. *Taking the sequence of the previous sub-sections we have:*

Input: $seq_{16} = 00011101\ 10001011$

- *Step 1:*
$$\begin{array}{rrr} & 0001 & 1101 \\ + & 1000 & 1011 \\ \hline sum = & 1001 & 0110 \end{array}$$
As sum = 1001 0110 $\neq$ $zeros_8$, then seq = sum = 1001 0110 and k = 8.
At this step, the binomial sequences $\binom{n}{3}$, $\binom{n}{4}$ and $\binom{n}{6}$ are cancelled.
- *Step 2:*
$$\begin{array}{rrr} & 10 & 01 \\ + & 01 & 10 \\ \hline sum = & 11 & 11 \end{array}$$
As sum = 11 11 $\neq$ $zeros_4$, then seq = sum = 11 11 and k = $8 + 4 = 12$.
At this step, the binomial sequences $\binom{n}{8}$, $\binom{n}{9}$ and $\binom{n}{10}$ are cancelled.
- *Step 3:*
$$\begin{array}{rrr} & 1 & 1 \\ + & 1 & 1 \\ \hline sum = & 0 & 0 \end{array}$$
As $sum = zeros_2$, then seq = 1 1.
At this step, there is no binomial sequence cancellation and the remaining binomial sequence is $\binom{n}{12}$.
- *Step 4:*
$$\begin{array}{rr} & 1 \\ + & 1 \\ \hline sum = & 0 \end{array}$$
As $sum = 0$, then seq = 1.
At this step, $length(seq) = 1$ and the algorithm stops.
- *Output: the maximum binomial sequence $\binom{n}{12}$ $\Rightarrow$ $LC = k + 1 = 12 + 1 = 13$.*

4.4. Matrix binomial decomposition or m-BD algorithm

This algorithm is based on the B-representation (or Binomial representation) [17] of a binary sequence $\{s_n\}_{n\geq 0}$ with length $l = 2^m$, m being a non-negative integer. Via the B-representation, the parameter LC of such a sequence is analyzed and computed.

We have seen that every sequence $\{s_n\}$ with length $l = 2^m$ can be written in terms of its binomial decomposition as:

$$\{s_n\} = \sum_{i=0}^{l-1} c_i \binom{n}{i}, \tag{6}$$

where c_i $(0 \leq i < l)$ are coefficients defined in the binary field $\mathbb{F}_2$ and $\binom{n}{i}$ $(0 \leq i < l)$ the corresponding binomial sequences. The greatest value of i, notated i_{max}, for which $c_{i_{max}} \neq 0$ while $c_i = 0$ for $i_{max} < i < l$, determines the value of the LC via the equation (1), that is

$$LC = i_{max} + 1. \tag{7}$$

Recall that the maximum linear complexity of $\{s_n\}_{n\geq 0}$ with length $l = 2^m$ will be $LC_{max} = 2^m$ when $c_{2^m-1} = 1$ while the minimum complexity of this kind of sequences will be $LC_{min} = 1$ when $c_0 = 1$ and $c_i = 0$ for $\forall i$ in the interval $0 < i < l$.

The B-representation provides one with a matrix method of computing the binary coefficients c_i. In fact, it defines a binary matrix, the so-called binomial matrix, constructed in a similar way to the construction of a binary Hadamard matrix.

In fact, consider $H_0 = [1]$ the binomial matrix for $m = 0$, that is, a $(2^0 \times 2^0)$ matrix with a unique entry. Next, we construct the binomial matrix for $m = 1$ as follows:

$$H_1 = \begin{bmatrix} H_0 & H_0 \\ 0 & H_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix},$$

where H_1 is a binary $(2^1 \times 2^1)$ matrix. Proceeding in the same way, we obtain the binomial matrix for m as

$$H_m = \begin{bmatrix} H_{m-1} & H_{m-1} \\ 0_{m-1} & H_{m-1} \end{bmatrix},$$

where H_{m-1} is the binomial matrix of size $(2^{m-1} \times 2^{m-1})$ as well as 0_{m-1} is the identically null matrix of the same size. Moreover, the matrix H_m can be written in terms of its columns as $H_m = (\tilde{h}_0, \tilde{h}_1, \ldots, \tilde{h}_{2^m-1})$.

As $\{s_n\}_{n \geq 0}$ is a binary sequence of length $l = 2^m$ and given the $(2^m \times 2^m)$ binomial matrix H_m, we compute the vector $\boldsymbol{c}$ whose 2^m components are the coefficients c_i by means of the equation (see [17, Sub-section 3.2]):

$$\boldsymbol{c} = [s_0, s_1, \ldots, s_{2^m-1}] \cdot H_m = [c_0, c_1, \ldots, c_{2^m-1}]_{mod\ 2}, \tag{8}$$

that is, the sequence $\{s_n\}$ is multiplied by the successive columns $\tilde{h}_i$ $(0 \leq i < 2^m)$ of the binomial matrix and the resulting products reduced mod 2.

Let us see an illustrative example.

Example 4. *Let $seq_{16} = \{0001110110001011\}$ be a sequence of length 2^4, so we must construct the binomial matrix for $m = 4$, that is*

$$H_4 = \begin{bmatrix}
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1 \\
0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \\
0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1 \\
0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1 \\
0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1 \\
0&0&0&0&0&1&0&1&0&0&0&0&0&1&0&1 \\
0&0&0&0&0&0&1&1&0&0&0&0&0&0&1&1 \\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1 \\
0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1 \\
0&0&0&0&0&0&0&0&0&1&0&1&0&1&0&1 \\
0&0&0&0&0&0&0&0&0&0&1&1&0&0&1&1 \\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1 \\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1
\end{bmatrix}.$$

From equation (8), we have that

$$\boldsymbol{c} = [s_0, s_1, s_2, \ldots, s_{15}] \cdot H_4 = [0\,0\,0\,1\,1\,0\,1\,0\,1\,1\,1\,0\,1\,0\,0\,0]\,.$$

Therefore, the vector $\boldsymbol{c} = [c_0, \ldots, c_{15}]$ corresponding to the sequence seq_{16} will have $c_3 = c_4 = c_6 = c_8 = c_9 = c_{10} = c_{12} = 1$ while the remaining components equal zero. The coefficients $c_i = 1$ correspond to the binomial sequences $\binom{n}{i}$ that appear in the binomial decomposition of seq_{16}.

In that case, the value of $i_{max} = 12$, or equivalently $c_{i_{max}} = c_{12} = 1$ and the LC of seq_{16} is $LC = 13$ as expected.

By construction, the binomial matrix is an upper triangular matrix closely related with the binomial sequences.

Remark 1. *The columns of the binomial matrix (read from right to left) correspond to the successive binomial sequences starting at the first* 1*. Thus, the binary vector* $\mathbf{c}$ *in equation (8) is just the product of the sequence* $\{s_n\}$*, written as a vector of* 2^m *components* $[s_0, s_1, \ldots, s_{2^m-1}]$*, multiplied by the* 2^m *first binomial sequences* $\binom{n}{i}$ *with* $0 \le i < 2^m$ *and* $n \ge i$*.*

4.4.1. Description of the m-BD algorithm:

In order to compute the LC of the sequence under consideration, the m-BD algorithm checks the successive coefficients c_i calculated in (8) starting at c_{2^m-1} and proceeding in decreasing order until the first coefficient $c_i = 1$ is found. In that case, $i_{max} = i$ and the LC is easily computed by means of the equation (7).

The final pseudo-code of the algorithm, for a given binary sequence of length $l = 2^m$ can be found in Algorithm 4.

Algorithm 4 : The m-BD algorithm

```
Input: seq = [s_0, s_1, ..., s_{2^m-1}] and the binomial matrix H_m = (h̃_0, h̃_1, ..., h̃_{2^m-1})
  i = 2^m - 1
  i_max = 0
  while i > 0 do
    c_i = [s_0, s_1, ..., s_{2^m-1}] · h̃_i
    if c_i == 0 then
      i − −
    else
      i_max = i
      i = 0
    end if
  end while
  Output: LC = i_max + 1: Linear complexity of seq.
```

At the same time, the computation of the coefficients c_i in the equation (8) allows us to characterize the binary sequences $\{s_n\}$ with maximum and quasi-maximum linear complexity.

4.4.2. Sequences with maximum LC:

The characterization of binary sequences $\{s_n\}_{n\ge0}$ with maximum linear complexity is described in the next result.

Theorem 2. *Let* $\{s_n\}_{n\ge0}$ *be a binary sequence with length* $l = 2^m$*, m being a non-negative integer. Such a sequence will have maximum linear complexity* $LC_{max} = 2^m$ *if and only if the sequence* $\{s_n\}$ *has an odd number of ones.*

Proof. ($\Rightarrow$) Maximum linear complexity implies that $c_{i_{max}} = c_{2^m-1} = 1$, but c_{2^m-1} is the product mod 2 of the sequence $[s_0, s_1, \ldots, s_{2^m-1}]$ by the last column $\tilde{h}_{2^m-1}$ of the binomial matrix (the identically 1 column), thus

$$c_{2^m-1} = \sum_{i=0}^{l-1} s_i. \quad (9)$$

Hence, $c_{2^m-1} = 1$ when the number of summands equal to 1 in equation (9) is an odd number.

($\Leftarrow$) If the number of terms $s_i = 1$ in the sequence $\{s_n\}$ is an odd number, then by equation (9) the coefficient $c_{2^m-1} = 1$. Consequently, $\{s_n\}$ will exhibit maximum linear complexity of value $LC_{max} = 2^m$. $\square$

Two corollaries follow directly from the previous theorem.

Corollary 1. *A binary sequence* $\{s_n\}_{n\geq 0}$ *with length* $l = 2^m$ *and an even number of ones will never attain the maximum linear complexity* $LC_{max} = 2^m$ *as* $c_{2^m-1} = 0$.

Corollary 2. *The linear complexity of every balanced binary sequence* $\{s_n\}_{n\geq 0}$ *with length* $l = 2^m$ *is upper bounded by* $LC < 2^m$.

Recall that, although balancedness is a suitable property for cryptographic sequences, a balanced sequence will never attain the maximum linear complexity.

4.4.3. Sequences with quasi-maximum LC:

The characterization of binary sequences $\{s_n\}_{n\geq 0}$ with quasi-maximum linear complexity, that is $LC = LC_{max} - 1$, is described in the next result.

Theorem 3. *Let* $\{s_n\}_{n\geq 0}$ *be a binary sequence with length* $l = 2^m$, *m being a non-negative integer. Such a sequence will have quasi-maximum linear complexity of value* $LC_{q-max} = 2^m - 1$ *if and only if* $\{s_n\}$ *satisfies the following conditions:*

1. *The sequence* $\{s_n\}$ *has an even number of ones.*
2. *It satisfies the equality:*

$$\sum_{i=0}^{l/2-1} s_{2\cdot i} = 1.$$

Proof. $(\Rightarrow)$

1. $\{s_n\}$ must have an even number of ones, otherwise by Theorem 2 the sequence would have maximum linear complexity.
2. Quasi-maximum linear complexity implies that $c_{2^m-2} = 1$, but c_{2^m-2} is the product mod 2 of the sequence $[s_0, s_1, \ldots, s_{2^m-1}]$ multiplied by the column $\tilde{h}_{2^m-2}$ in the binomial matrix (the $1\,0\,1\,0 \ldots 1\,0$ column), thus

$$c_{2^m-2} = \sum_{i=0}^{l/2-1} s_{2\cdot i}.$$

   Hence, $c_{2^m-2} = 1$ when the number of terms $(s_{2\cdot i})$ (terms with even indices) equal to 1 is an odd number.

$(\Leftarrow)$

1. If the sequence $\{s_n\}$ has an even number of ones, then $c_{2^m-1} = 0$.
2. If $\{s_n\}$ satisfies the equality

$$\sum_{i=0}^{l/2-1} s_{2\cdot i} = 1,$$

   then $c_{2^m-2} = 1$.

Thus, $c_{2^m-1} = 0$ and $c_{2^m-2} = 1$ jointly imply quasi-maximum linear complexity of value $LC_{q-max} = 2^m - 1$. □

5. Algorithm Comparison

All the algorithms explained in the previous section can be used to calculate the linear complexity of a given sequence with length a power of two. Now, they will be compared in different ways. The section is scheduled as follows:

First of all, the different computational features of these algorithms are discussed. Next, we describe the experiments we carried out to compare the actual performance of such algorithms. Finally, we consider diverse scenarios apart from LC calculation where each algorithm might be conveniently applied.

5.1. Algorithm analysis

In section 4, different algorithms for the computation of the linear complexity were presented (Berlekamp-Massey, BD, half-interval search and m-BD algorithms). Now, we will discuss the computational complexity and sequence length requirements for each one of them as shown in Table 5.

The length requirements (twice the length of the studied sequence) and complexity $O(l^2)$ of the Berlekamp-Massey algorithm have been already studied in the literature [11,32]. It is the only algorithm, among the considered algorithms, that can be applied to every sequence of any length, compared with the binomial decomposition methods that require a sequence of length a power of two.

Concerning the BD-algorithm, in order to calculate the linear complexity it needs at least $l - \log l$ bits of the original sequence and it runs with a computational complexity of $O(r \cdot l)$, l being the length of the sequence and r the number of binomial components in its decomposition. Although the parameter r has not been rigorously analyzed, in Figure 2 an experimental analysis of r was carried out for different GSS-sequences. The results show that such a parameter follows a normal distribution as well as it increases with the length of the sequence.

Table 5. Algorithm comparison

Algorithms	Length Required	Complexity	Seq. Restrictions
Berlekamp-Massey algorithm	$2 * l$	$O(l^2)$	None
BD-algorithm	$l - \log l$	$O(r \cdot l)$	Length power of 2
Half-interval search algorithm	$l - \log l$	$O(l)$	Length power of 2
m-BD algorithm	l	$O(l^2)$ - $\Omega(l)$	Length power of 2

On the other hand, the half-interval search algorithm does not depend neither on the parameter r nor on the decomposition of the sequence. In fact, this algorithm just requires the same number of bits as that of the BD-algorithm, but it works in a binary search fashion. Consequently, its complexity is linear in the length of the sequence, which means the best performance among all the algorithms that can calculate LC.

The main difference between BD and half-interval search algorithms is that the latter does not depend on the number of binomial sequences in its binomial decomposition. That means that its performance will be better than that of the BD-algorithm, in particular when the length of the sequence increases and so does the value of the parameter r.

Finally, the m-BD algorithm computes the successive products between two binary vectors until it gets the value of LC. Nevertheless, the worst case would occur whether it needed to check all the columns of the binomial matrix. That is the reason why we included in Table 5 both worst and best cases of computational complexity.

Although the Berlekamp-Massey algorithm is able to calculate the linear complexity of any sequence, it is not the best choice for particular sequences as the GSS-sequences with $O(l^2)$. It is under such circumstances when the binomial decomposition algorithms can be really useful.

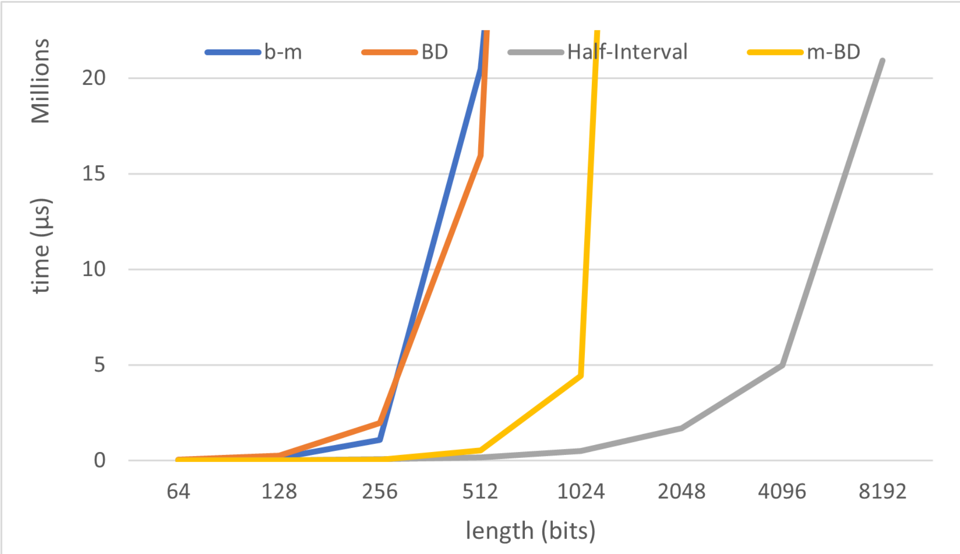
5.2. Experimental results

To support the understanding of these algorithms and test them, we ran all the algorithms described in the previous section.

The setup of the experiments is as follows: we used Jupyter Labs as a running environment in a Windows 10 machine with Intel Core i7-1065G7 as CPU. The algorithms were implemented in Python

3. They ran to calculate the LC for the same sequences several times in order to get the performance metric of such algorithms.

Figure 4. Comparison between the algorithms in the LC calculation for all the possible GSS-sequences of a given length (Half-Interval scale)

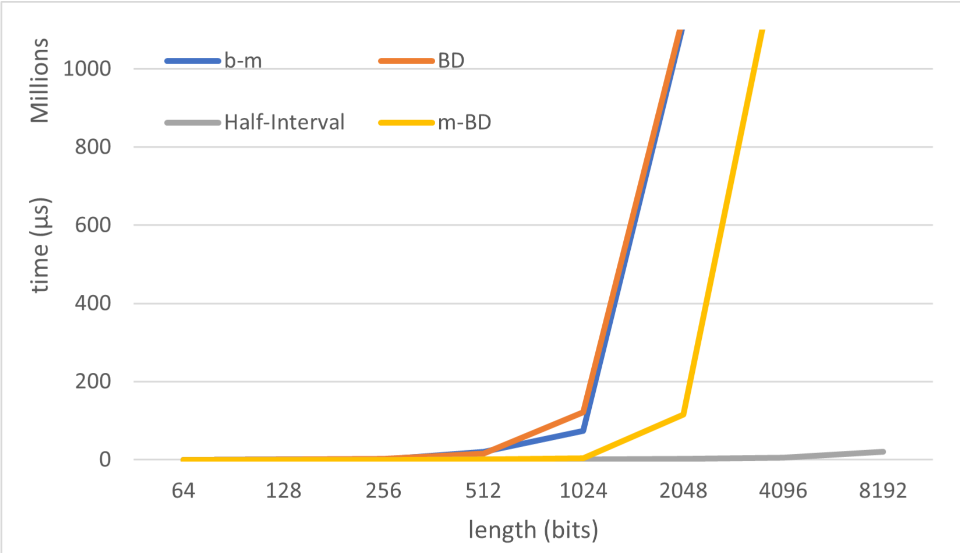


The results of the experiments can be seen in Figure 4 and Figure 6. Indeed, in Figure 4 where all algorithms are compared, we can see how as far as the length of the sequence increases, both the half-interval algorithm and the matrix binomial decomposition algorithm improve the performance exhibited by the Berlekamp-Massey algorithm. This proves that the binomial decomposition technique can be useful and a good alternative in the study of sequences that are particularly hard to be analyzed by the Berlekamp-Massey algorithm.

About the Berlekamp-Massey and the Binomial Decomposition algorithm, there is a bounce in their performance depending on the length of the sequences of the experiment. According to the study of the BD complexity, it is known that its performance depends on the parameter r, or in other words, it depends on the number of binomial sequences in the decomposition for each sequence. After the preliminary study on the parameter r, seen in Figure 2, the parameter r is expected to behave in a normal distribution fashion. Altogether this means that the BD algorithm can slightly change its performance depending on the r value of the sequences it is studying.

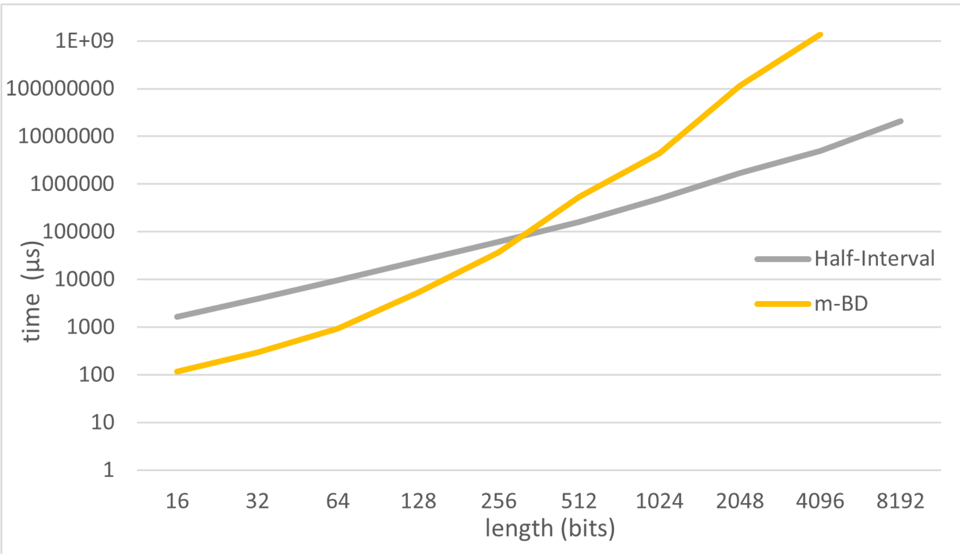
In addition, the theoretical improvement of the half-interval algorithm studied in previous section is confirmed. The huge performance gap between Berlekamp-Massey algorithm, BD-algorithm, m-BD and the half-interval search algorithm can be seen in Figure 4 and in Figure 5. Recall that this gap is particularly remarkable when the length of the sequence studied increases. For that reason we included Figure 4, scaled for a better comparison with the half-interval results (the best performant algorithm), and Figure 5, for a better comparison with m-BD (a novel contribution of this article).

Figure 5. Comparison between the algorithms in the LC calculation for all the possible GSS-sequences of a given length (m-BD scale)



Furthermore, we wanted to compare the half-interval algorithm with the new m-BD algorithm, which has not been previously studied neither its performance is known. In Figure 6, a logarithmic scaled graph is depicted. We see how the half-interval search algorithm outperforms the m-BD algorithm provided that the length of the sequence studied is increased. This behaviour seems to reveal that the increment in the sequence length makes worse the m-BD algorithm performance, since m-BD requires more tries to calculate the LC.

Figure 6. Comparison between half-interval search and m-BS algorithms



Although it is not the purpose of this work, it is worth noticing that the half-interval search algorithm can be parallelized in the computation of LC while the BD-algorithm performs the computation in a sequential way.

Another point that was not covered in the experiments is how the m-BD algorithm can take profit of some optimizations in the computation of matrix operations, which explain its great speed when the sequences are not too long. In addition, it could be enhanced while running in environments specially designed for it such as MATLAB.

5.3. Different Use-cases

After the analysis and the experiments to test the performance of the algorithms, it is also worth exploring different application scenarios, not only the linear complexity calculation. All the algorithms that use the binomial decomposition calculate the *LC* with the maximum binomial component.

A different case for these algorithms could be the study in depth of other types of binary sequences. In fact, having their full decomposition can help to analyze more parameters related to the security of the sequences, e.g. to calculate the density of components in the decomposition or the balancedness of such sequences. It is in this case where the BD-algorithm outperforms the others, since the way it calculates the *LC* is by means of the computation of all the binomial components.

Another interesting use-case for these algorithms is, for instance, processing a large amount of sequences in order to discern as fast as possible which ones have better/worse security. In that case, the m-BD algorithm is the best one, because it can determine whether the highest binomial component is present in the binomial decomposition previously to complete the *LC* calculation. So the m-BD algorithm may not be the fastest algorithm to calculate the *LC* of a particular sequence but it may be used to quickly detect which sequence has a *LC* lower than the others.

Finally, the m-BD algorithm could be of great use if the range of the linear complexity is known. In that case, this parameter would avoid unnecessary tries of the algorithm, which otherwise will profit from the matrix optimizations that modern libraries support.

6. Conclusions

In this work, different algorithms to compute the linear complexity of binary sequences have been introduced and analyzed. In general, they exhibit better performances than the well-known Berlekamp-Massey algorithm when applied to sequences suitable for cryptography.

Concerning the half-interval search algorithm presented in this article, it shows excellent results in both computational complexity and amount of sequence required. It was also tested in comparison with other algorithms by applying it to GSS-sequences, showing an improved performance when the length of the sequences increases.

The matrix binomial decomposition algorithm showed a good performance with short sequences. Nevertheless, its main characteristic, that is the way in which it identifies the binomial components of a sequence, can be useful in other scenarios apart from the *LC* calculation, e.g. to discern between a large amount of sequences which ones have a better complexity than the others.

Moreover, the binomial decomposition of binary sequences seems to be an innovative technique to extract information from a given sequence. In particular, the fractal character of the binomial sequences can be employed to calculate diverse parameters of a sequence without knowing the whole sequence.

In brief, the analysis of these algorithms is quite useful to find weaknesses in this type of binary sequences. Indeed, detecting such weaknesses in a cipher with practical applications could compromise the corresponding IoT device and, consequently, the services that rely on it.

Funding: Research partially supported by Ministerio de Economía, Industria y Competitividad, Agencia Estatal de Investigación, and Fondo Europeo de Desarrollo Regional (FEDER, UE) under project COPCIS (TIN2017-84844-C2-1-R) and by Comunidad de Madrid (Spain) under project CYNAMON (P2018/TCS-4566), also co-funded by European Union FEDER funds

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Chin, W.L.; Li, W.; Chen, H.H. Energy big data security threats in IoT-based smart grid communications. *IEEE Communications Magazine* **2017**, *55*, 70–75.
2. Meyer, D.; Haase, J.; Eckert, M.; Klauer, B. New attack vectors for building automation and IoT. IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society, pp. 8126–8131.
3. Gallegos-Segovia, P.L.; others. Internet of things as an attack vector to critical infrastructures of cities. 2017 International Caribbean Conference on Devices, Circuits and Systems (ICCDCS), pp. 117–120.
4. Rouf, I.; others. Security and Privacy Vulnerabilities of In-Car Wireless Networks: A Tire Pressure Monitoring System Case Study. USENIX Security Symposium, 2010, Vol. 10.
5. Cynthia, J.; Sultana, H.P.; Saroja, M.; Senthil, J. Security protocols for IoT. In *Ubiquitous Computing and Computing Security of IoT*; Springer, 2019; pp. 1–28.
6. Mavromoustakis, C.X.; Mastorakis, G.; Batalla, J.M. *Internet of Things (IoT) in 5G mobile technologies*; Vol. 8, Springer, 2016.
7. NIST Lightweight Cryptography project. https://csrc.nist.gov/Projects/Lightweight-Cryptography. Last accessed 15 February 2021.
8. McGinthy, J.M. Solutions for Internet of Things Security Challenges: Trust and Authentication. PhD thesis, Virginia Tech, 2019.
9. NIST Lightweight Cryptography project Round 2 Candidates. https://csrc.nist.gov/Projects/lightweight-cryptography/round-2-candidates. Last accessed 15 February 2021.
10. Dubrova, E.; Hell, M. Espresso: A stream cipher for 5G wireless communication systems. *Cryptography and Communications* **2017**, *9*, 273–289.
11. Massey, J. Shift-register synthesis and BCH decoding. *IEEE transactions on Information Theory* **1969**, *15*, 122–127.
12. Cardell, S.D.; Fúster-Sabater, A. Binomial Representation of Cryptographic Binary Sequences and Its Relation to Cellular Automata. *Complexity* **2019**.
13. Chang, K.Y.; others. Method and apparatus for generating keystream, 2009. US Patent 7,587,046.
14. Kang, Y.S.; Kim, H.W.; Chung, K.I. Apparatus and method for protecting RFID data, 2013. US Patent 8,386,794.
15. Falk, R.; Merli, D. Programmable logic device, key generation circuit and method for providing security information, 2016. EP3146520.
16. Martin-Navarro, J.L.; Fúster-Sabater, A. Folding-BSD Algorithm for Binary Sequence Decomposition. *Computers* **2020**, *9*, 100.
17. Cardell, S.D.; Climent, J.J.; Fúster-Sabater, A.; Requena, V. Representations of Generalized Self-Shrunken Sequences. *Mathematics* **2020**, *8*, 1006.
18. Golomb, S.W. *Shift register sequences*; Aegean Park Press, 1967.
19. Menezes, A.J.; van Oorschot, P.C.; Vanstone, S.A. *Handbook of Applied Cryptography*; CRC Press, Boca Raton, 1996.
20. Paar, C.; Pelzl, J. *Understanding Cryptography*; Springer, Berlin, 2010.
21. Cardell, S.D.; Fúster-Sabater, A. The t-Modified self-shrinking generator. International Conference on Computational Science (ICCS 2018). Springer, 2018, pp. 653–663.
22. Cardell, S.D.; Fúster-Sabater, A. *Cryptography with Shrinking Generators: Fundamentals and Applications of Keystream Sequence Generators Based on Irregular Decimation*; Springer, Series: Briefs in Mathematics, 2019.
23. Coppersmith, D.; Krawczyk, H.; Mansour, Y. The shrinking generator. Annual International Cryptology Conference. Springer, 1993, pp. 22–39.
24. Meier, W.; Staffelbach, O. The self-shrinking generator. In *Communications and Cryptography*; Springer, 1994; pp. 287–295.
25. Hu, Y.; Xiao, G. Generalized self-shrinking generator. *IEEE Transactions on Information Theory* **2004**, *50*, 714–719.
26. Mihaljevic, M.J. A faster cryptanalysis of the self-shrinking generator. Information Security and Privacy, First Australasian Conference, ACISP'96, Wollongong, NSW, Australia, June 24-26, 1996, Proceedings. Springer, 1996, Vol. 1172, *Lecture Notes in Computer Science*, pp. 182–189. doi:10.1007/BFb0023298.

27. Simpson, L.; Golic, J.D.; Dawson, E. A Probabilistic Correlation Attack on the Shrinking Generator. Information Security and Privacy, Third Australasian Conference, ACISP'98, Brisbane, Queensland, Australia, July 1998, Proceedings. Springer, 1998, Vol. 1438, *Lecture Notes in Computer Science*, pp. 147–158. doi:10.1007/BFb0053729.
28. Caballero, P.; Fúster-Sabater, A.; Pazo, M.E. New Attack Strategy for the Shrinking Generator. *Journal of Research and Practice in Information Technology* **2009**, *23*, 171–180.
29. Fúster-Sabater, A.; Pazo, M.E.; Caballero, P. A Simple Linearization of the Self-Shrinking Generator by Means of Linear Cellular Automata. *Neural Networks* **2010**, *23*, 461–464.
30. Cardell, S.D.; Fúster-Sabater, A.; Ranea, A.H. Linearity in decimation-based generators: an improved cryptanalysis on the shrinking generator. *Open Mathematics* **2018**, *16*, 646–655.
31. Fúster-Sabater, A.; Cardell, S.D. Linear complexity of generalized sequences by comparison of PN-sequences. *Revista de la Real Academia de Ciencias Exactas, Físicas y Naturales. Serie A. Matemáticas* **2020**, *114*, 79.
32. Cusick, T.W.; Stanica, P. *Cryptographic Boolean functions and applications*; Academic Press, 2017.
33. Martin-Navarro, J.L.; Fúster-Sabater, A. Folding-BSD Algorithm for Binary Sequence Decomposition. International Conference on Computational Science and Its Applications (ICCSA 2020). Springer, 2020, pp. 345–359.