computers

MDPI

Article

Folding-BSD Algorithm for Binary Sequence Decomposition

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

1 Department of Computer Science, School of Science, Aalto University, 02150 Espoo, Finland
2 Instituto de Tecnologías Físicas y de la Información, C.S.I.C., 28006 Madrid, Spain; amparo@iec.csic.es
* Correspondence: martinnavarroj@acm.org
† Current address: Spanish National Research Council (CSIC), Serrano 144, 28006 Madrid, Spain.

Received: 15 November 2020; Accepted: 9 December 2020; Published: 15 December 2020

Abstract: The Internet of Things (IoT) revolution leads to a range of critical services which rely on IoT devices. Nevertheless, they often lack proper security, becoming the gateway to attack the whole system. IoT security protocols often rely on stream ciphers, where pseudo-random number generators (PRNGs) are an essential part of them. In this article, a family of ciphers with strong characteristics that make them difficult to be analyzed by standard methods is described. In addition, we will discuss an innovative technique of sequence decomposition and present a novel algorithm to evaluate the strength of binary sequences, a key part of the IoT security stack. The density of the binomial sequences in the decomposition has been studied experimentally to compare the performance of the presented algorithm with previous works.

Keywords: PRNG; LFSR; binomial sequences; stream ciphers; IoT

1. Introduction

Breakthroughs in electronics and telecommunications fields have made Internet of Things (IoT) a reality, with an every day growing number of sensors and devices around us. Nowadays, diverse critical services, such as smart-grid, e-health, agricultural or industrial automation, depend on an IoT infrastructure. At any rate, as the services around IoT grow dramatically, so do the security risks [1,2].

Low-cost IoT devices, currently characterized by their resource constraints in processing power, memory, size and energy consumption, are also recognizable by their minimal or non-existent security. Combining this lack of security with their network dependability, they become the perfect target as a gateway to the whole network [3]. There are already some attacks where a vulnerable IoT sensor was used to gain control over the whole system (in [4], wireless sensors are used to gain access to an automotive system).

This is the reason why general research [5], 5G-related research [6] or specific calls such as that of NIST for lightweight cryptography primitives [7], are addressing this concerning topic. Although different protocols of communication and orchestration are being proposed [8], lightweight cryptography in general and stream ciphers in particular are the stepping stones on which such protocols are built to guarantee both device and network security.

Currently, linear feedback shift registers (LFSRs) are key components in stream ciphers, often used as pseudo-random number generators (PRNG). A representative example of LFSR-based practical design is the J3Gen, a pseudo-random number generator for low-cost passive radio frequency identification (RFID) tags. Such a generator has been analyzed and revised in [9,10], respectively.

In addition, we present the generalized self-shrinking generator, a family of ciphers suitable for cryptography [11]. This generator remains strong against different algorithms that analyze the

cryptographic strength (e.g., linear complexity) of its sequences. Such algorithms can be enumerated as follows: (1) the well-known Berlekamp–Massey algorithm [12], (2) the sequence decomposition algorithm developed by Cardell et al. in [13] and (3) the folding sequence decomposition algorithm introduced in this paper. To our knowledge, no other algorithms to calculate the parameter linear complexity of binary sequences are described in the literature.

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

The paper is organized as follows. Section 2 includes a succinct revision of LFSRs and sequence generators based on irregular decimation, a wide type of generators including the generalized self-shrinking generator. Section 3 describes the characteristics and generalities of the binomial sequences (BS), binary sequences that constitute the foundations of the two last algorithms mentioned above. The main contributions of this work are in Section 4, where we analyze three algorithms to calculate the linear complexity of binary sequences: (a) the standard Berlekamp–Massey algorithm, (b) the basic binomial sequence decomposition (b-BSD) algorithm, an improved version of the algorithm developed in [13], that analyzes different properties of the binary sequences and (c) the folding binomial sequence decomposition (f-BSD) algorithm, the novel proposal based on the symmetry of the binomial sequences. Section 5 includes the discussion and comparison among the three previous algorithms. Finally, conclusions and possible future research lines in Section 6 end the paper.

2. Sequence Generators Based on Linear Feedback Shift Registers

Linear feedback shift registers (LFSRs) [16] are linear structures currently used in the generation of pseudo-random sequences. The LFSRs are electronic devices included in the majority of sequence generators described in the literature. The main reasons for such a generalized use of LFSRs are: these devices provide high performance when used for sequence generation, they are particularly well-suited to hardware implementations, and such registers can be easily analyzed by means of basic algebraic techniques.

According to Figure 1, a LFSR consists of: (a) L interconnected stages numbered from left to right $(0, 1, \ldots, L-1)$ where each of them stores one bit, (b) the feedback polynomial $p(x)$ of degree L notated

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

with binary coefficients c_i and (c) the initial state or stage contents at the initial time. LFSRs generate sequences by means of shifts of the stage contents and linear feedback to the last stage.

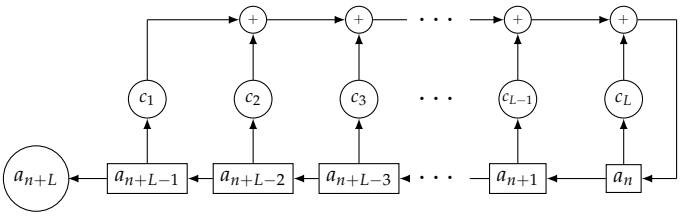


Figure 1. Sketch of a linear feedback shift register (LFSR) of length L.

The output of an LFSR with nonzero initial state is a binary sequence denoted by $\{a_n\}$ $(n = 0, 1, 2, \ldots)$. If the polynomial $p(x)$ is primitive [16], then the output sequence is called PN-sequence (or pseudo-noise sequence). In addition, the period of a PN-sequence is $T = 2^L - 1$ bits with 2^{L-1} ones and $2^{L-1} - 1$ zeros.

The linear complexity of a sequence, notated LC, quantifies the number of bits necessary to reconstruct the whole sequence. For particular applications of binary sequences, e.g., cryptographic applications (stream ciphers), LC must be as large as possible. Moreover, the linear complexity is a parameter closely

related to the concept of LFSR. In fact, the linear complexity of a sequence is defined as the length of the shortest LFSR that generates such a sequence.

Although an LFSR in itself is an excellent generator of pseudo-random sequence, in practice it has undesirable linearity properties which reduce the security of its use. Due to these linearities, LFSRs exhibit a very low *LC* of value just L. Consequently, in the process of generation of the pseudo-random sequence, any type of nonlinearity must be introduced. The irregular decimation of PN-sequences is one of the most popular techniques to destroy the inherent linearity of the LFSRs [17,18]. In brief, this procedure is based on the fact of removing some bits of a PN-sequence according to the bits of another one.

Inside the type of irregularly decimated generators, we can enumerate: (a) the shrinking generator introduced in [19] that includes two LFSRs, (b) the self-shrinking generator [20] that involves just one LFSR and (c) the generalized self-shrinking generator proposed in [21] that is considered as a simplification of the shrinking generator as well as a generalization of the self-shrinking generator. In this work, we focus on the last type of generator, that is, the generalized self-shrinking generator.

The Generalized Self-Shrinking Generator (GSSG)

The generalized self-shrinking generator can be described as follows:

1. It makes use of two PN-sequences: $\{a_n\}$ a PN-sequence produced by an LFSR with L stages and a shifted version of such a sequence denoted by $\{v_n\}$. In fact, $\{v_n\} = \{a_{n+p}\}$, thus $\{v_n\}$ corresponds to the own sequence $\{a_n\}$ rotated cyclically p positions to the left with $(p = 0, 1, \ldots, 2^L - 2)$.
2. In order to generate the output sequence, it relates both sequences by means of a simple decimation rule.

   For $n \geq 0$, the decimation rule is defined as follows:

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

Thus, for each value of p, an output sequence $\{s_n\}_p = \{s_0\, s_1\, s_2 \ldots\}_p$ is generated. Such a sequence is called the p generalized self-shrunken sequence (GSS-sequence) or simply generalized sequence associated with the shift p. Recall that $\{a_n\}$ remains fixed while $\{v_n\}$ is the sliding sequence or left-shifted version of $\{a_n\}$. If p ranges in the interval $[0, 1, \ldots, 2^L - 2]$, then we obtain the $2^L - 1$ members of the family of generalized sequences based on the PN-sequence $\{a_n\}$. Since the PN-sequence has 2^{L-1} ones, the period of any generalized sequence will be 2^{L-1} or divisors of this number, in any case, a power of 2. In addition, the *LC* of every GSS-sequence is upper-bounded by $2^{L-1} - (L - 2)$ [22] (Theorem 2). Next, an illustrative example of a family of GSS-sequences is introduced.

Example 1. *For an LFSR with primitive polynomial* $p(x) = x^4 + x + 1$ *and initial state* $(1, 1, 1, 1)$*, we generate the generalized sequences depicted in Table 1. The bits in bold in the different sequences* $\{v_n\}$ *are the digits of the corresponding GSS-sequence associated to the corresponding shift* p*. The PN-sequence* $\{a_n\}$ *with period* $T = 2^4 - 1$ *is written at the bottom of the table in columns 2 and 5.*

There are two particular facts that differentiate the GSS-sequences from the PN-sequences generated by LFSRs:

- The period of the GSS-sequences is a power of 2, in contrast to PN-sequences whose period is $2^L - 1$. This difference arises from the fact that PN-sequences cannot have a run of zeros of the length of its internal state.
- The *LC* of PN-sequences equals L, while that of the GSS-sequences is near to 2^{L-1} [22,23]. This property is desirable because such sequences exhibit a great *LC* with very low resources.

Table 1. GSS-sequences for an LFSR with polynomial $p(x) = x^4 + x + 1$, in bold the bits of the sequence that form the GSS-sequence.

Shift p	$\{v_n\}$ Sequences	GSS-Sequences	Shift p	$\{v_n\}$ Sequences	GSS-Sequences
0	**1111**0001001**1**0**1**0	11111111	8	**0011**0101**1**11**0**0**0**1	0011110 0
1	**1110**0010**0**1**10**1**0**1	11100100	9	**0110**1011**1**1**00**0**1**0	0110100 1
2	**1100**0100**1**1**01**0**1**1	11000011	10	**1101**0111**1**0**00**1**0**0	1101100 0
3	**1000**1001**1**0**10**1**1**1	10001101	11	**1010**1111**0**0**01**0**0**1	1010101 0
4	**0001**0011**0**1**01**1**1**1	00011011	12	**0101**1111**0**0**0**1**0**1**1	0101010 1
5	**0010**0110**1**0**1**1**1**0	00100111	13	**1011**1100**0**1**0**0**1**1**0	1011000 1
6	**0100**1101**0**1**1**1**0**0	01001110	14	**0111**1001**0**0**1**1**0**1	0111001 0
7	**1001**1010**1**1**1**0**0**0	10010110	−−	**0000**0000**0**0**0**0**0**0	00000000
	11110001001**1**0**1**0			**1111**0001001**1**0**1**0	

3. Binomial Sequences: Characteristics and Generalities

In this section, we introduce the binomial sequences as a new representation of the binary sequences whose period is a power of 2. Next, the close relationship between binomial sequences and Sierpinski's triangle is also considered.

3.1. Concepts and Fundamentals of the Binomial Sequences

The binomial number $\binom{n}{i}$ is the coefficient of the i-th power x^i in the polynomial expansion of $(1+x)^n$. For every non-negative integer n, it is a well-known fact that $\binom{n}{0} = 1$, as well as $\binom{n}{i} = 0$ for all $i > n$.

The concept of binomial sequence is defined in terms of the binomial coefficients $\binom{n}{i}$ reduced modulo 2.

Definition 1. *Given an integer $i \geq 0$, the sequence $\{b_n\}_i$ $(n = 0, 1, 2, \ldots)$ is called the i-th binomial sequence if its elements are binomial coefficients reduced modulo* 2*, that is $b_n = \binom{n}{i} \bmod 2$.*

Binomial sequences are currently notated $\{\binom{n}{i}\}_{mod\,2}$ $(n = 0, 1, 2, \ldots)$ and the integer i is the index of the binomial sequence. When n takes successive values, the binomial coefficients modulo 2 define the terms of a binary sequence with its corresponding period and linear complexity.

Table 2 shows the eight first binomial sequences $\{\binom{n}{i}\}$, $i = 0, 1, \ldots, 7$, see [24], where T_i and LC_i denote period and linear complexity, respectively.

Table 2. The eight first binomial sequences, their periods and linear complexities.

Binomial Coeff.	Binomial Sequences $\{\binom{n}{i}\}$	Period	Linear Complexity
$\binom{n}{0}$	$\{1, 1, 1, 1, 1, 1, 1, 1, \ldots\}$	$T_0 = 1$	$LC_0 = 1$
$\binom{n}{1}$	$\{0, 1, 0, 1, 0, 1, 0, 1, \ldots\}$	$T_1 = 2$	$LC_1 = 2$
$\binom{n}{2}$	$\{0, 0, 1, 1, 0, 0, 1, 1, \ldots\}$	$T_2 = 4$	$LC_2 = 3$
$\binom{n}{3}$	$\{0, 0, 0, 1, 0, 0, 0, 1, \ldots\}$	$T_3 = 4$	$LC_3 = 4$
$\binom{n}{4}$	$\{0, 0, 0, 0, 1, 1, 1, 1, \ldots\}$	$T_4 = 8$	$LC_4 = 5$
$\binom{n}{5}$	$\{0, 0, 0, 0, 0, 1, 0, 1, \ldots\}$	$T_5 = 8$	$LC_5 = 6$
$\binom{n}{6}$	$\{0, 0, 0, 0, 0, 0, 1, 1, \ldots\}$	$T_6 = 8$	$LC_6 = 7$
$\binom{n}{7}$	$\{0, 0, 0, 0, 0, 0, 0, 1, \ldots\}$	$T_7 = 8$	$LC_7 = 8$

Next, fundamental properties of the binomial sequences that will be used throughout the work are introduced.

1. Given the binomial sequence $\{\binom{n}{2^r+i}\}$, with $0 \leq i < 2^r$ and r being a non-negative integer, we have that [13] (Proposition 3):

(a) This binomial sequence has period $T = 2^{r+1}$.
(b) The period of such a binomial sequence has the following structure:

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

2. Every binary sequence whose period is a power of 2 can be written as a linear combination of a finite number of binomial sequences [13] (Theorem 2). Such a combination is called the binomial sequence decomposition (BSD) of the sequence under consideration.
3. The linear complexity of the binomial sequence $\left\{\binom{n}{i}\right\}$ with $i \ge 0$ is $LC = i + 1$, see [13] (Theorem 13).
4. Given a sequence with BSD $\sum_{k=0}^{t}\left\{\binom{n}{i_k}\right\}$, where $i_0 < i_1 < \cdots < i_t$ are integer indexes, then its linear complexity is given by $LC = i_t + 1$, see [13] (Corollary 14).
5. Given a sequence with BSD $\sum_{k=0}^{t}\left\{\binom{n}{i_k}\right\}$, where $i_0 < i_1 < \cdots < i_t$ are integer indexes, then its period T is that of the binomial sequence $\left\{\binom{n}{i_t}\right\}$, see [24] (Theorem 1).

According to the previous properties, two important remarks are derived:

Remark 1. *The period T of the binomial sequences $\left\{\binom{n}{i}\right\}$ ($i \ge 0$) can be quantified as follows:*

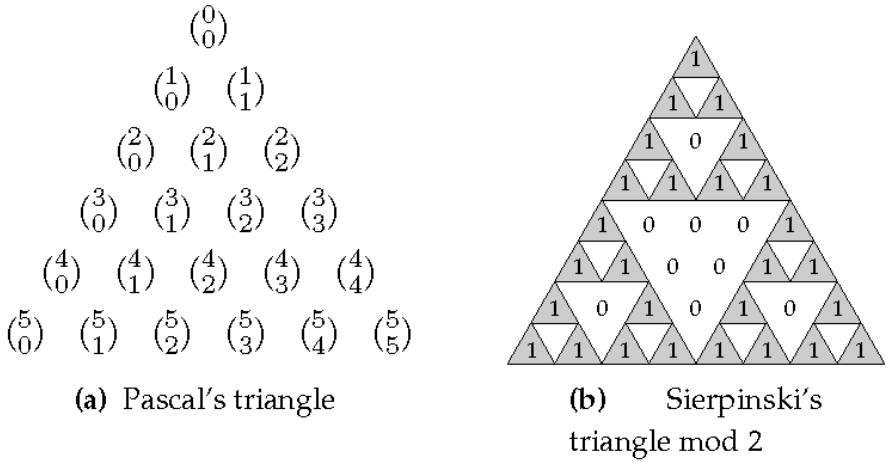
- 2^0 *binomial sequences with period* $T = 2^1$.
- 2^1 *binomial sequences with period* $T = 2^2$.
- 2^2 *binomial sequences with period* $T = 2^3$.
- *...and so on.*
- *In general,* 2^i *binomial sequences with period* $T = 2^{i+1}$ *with* $i \ge 0$.

On the other hand, the linear complexity of the binomial sequences is different for each binomial sequence $\left\{\binom{n}{i}\right\}$ and takes the value $LC = i + 1$.

Remark 2. *The generalized sequences are binary sequences whose period is a power of* 2. *Consequently, they can be written in terms of binomial sequences satisfying all the previous properties.*

3.2. The Sierpinski's Triangle

When the binomial coefficients are arranged into rows for the successive values of $n = 0, 1, 2, \ldots$ (a row for each value of n), then the generated structure is the well-known Pascal's triangle depicted in Figure 2a. If we color the odd numbers and shade the even ones in such a triangle, then we get the Sierpinski's triangle whose version reduced mod 2 is depicted in Figure 2b.



(a) Pascal's triangle

(b) Sierpinski's triangle mod 2

Figure 2. Pascal and Sierpinski's triangles.

Recall that the successive diagonals of the Sierpinski's triangle in Figure 2b correspond to the successive binomial sequences $\left\{\binom{n}{i}\right\}$, $(i = 0, 1, 2, \ldots)$ with $n \geq i$. That is, each diagonal of the triangle starts at the first 1 of the corresponding binomial sequence. Consequently, the binomial sequences can be found inside the Sierpinski's triangle mod 2, as well as they can also be found inside certain linear cellular automata, e.g., cellular automata with rules 102 and 60 in Wolfram's notation. See [13] for more details.

4. Algorithms to Calculate the Linear Complexity

In this section, we describe and develop different algorithms to calculate the linear complexity of binary sequences. Except for the first algorithm (Berlekamp–Massey algorithm), the remaining may be applied exclusively to sequences whose length (period) is a power of 2. They are based on the characteristics and properties of the binomial sequences.

Once the Berlekamp–Massey algorithm has been considered, this section introduces in detail the basic binomial sequence decomposition algorithm (b-BSD) and possible improvements to its implementation.

Next, a new approach to the binomial sequence decomposition is developed, giving rise to the folding binomial sequence decomposition algorithm (f-BSD). Such an algorithm improves the throughput of previous methods thanks to the symmetry of the binomial sequences.

A comparison among Berlekamp–Massey, b-BSD and f-BSD algorithms will be presented in the next section of this work.

Previously to the algorithms, we introduce a particular notation to be used systematically in the sequel.

NOTATION: For the sake of simplicity, in the successive algorithms of this section, the binomial coefficient $\binom{n}{k}$ will denote the corresponding k-th binomial sequence. Then, the term $\binom{n}{k}_{i,j}$ stands for the binary sub-sequence of $\binom{n}{k}$ between the bits i and j, while $\binom{n}{k}_j$ denotes the j first bits of the binomial sequence $\binom{n}{k}$.

4.1. Berlekamp–Massey Algorithm

Traditionally, the general method of calculating the linear complexity of a sequence is the Berlekamp–Massey algorithm [12]. It can be applied to sequences of any length, not necessarily a power of 2. Consequently, it is the most general among the algorithms that calculate the linear complexity of a sequence.

Given a binary sequence, this algorithm provides one with a general solution to the problem of synthesizing the shortest LFSR that generates such a sequence. At each step, the algorithm incrementally adjusts the length of the LFSR and the feedback polynomial to generate the sub-sequence analyzed to that point. The algorithm terminates when the whole sequence has been generated.

In order to work, this algorithm needs to process $2 * LC$ bits of the sequence with a computational complexity of $O(LC^2)$ [25].

4.2. Basic Binomial Sequence Decomposition (b-BSD)

Based on the mathematical results provided in the previous section, a basic binomial sequence decomposition algorithm (b-BSD) can be designed in order to calculate the LC of a given sequence. In particular, two facts are used:

- A sequence of length l can be decomposed in $t + 1$ binomial sequences (item 2. in Section 3.1):

$$seq_l = \binom{n}{k_0} + \cdots + \binom{n}{k_t}.$$

- The linear complexity of a sequence can be calculated from the binomial sequence of maximum index in its BSD (item 3. in Section 3.1). Since the binomial sequences are written in increasing order, then LC satisfies the following expression:

$$LC = max_{0 \leq i \leq t}\left(\binom{n}{k_i}\right) + 1 = k_t + 1. \quad (1)$$

The resulting algorithm can be seen in Algorithm 1. It takes the sequence to be analyzed as input and checks for every bit equal to 1. When bit_i is equal to 1, it sums the sequence with the corresponding binomial sequence ($seq = seq + \binom{n}{i}$), stopping when all the binomial sequences have been already found. In that case, the resulting sequence seq is the identically null sequence.

Once the BSD has been obtained, the binomial sequence $\binom{n}{k_t}$ allows us, via the Equation (1), to calculate LC. A step-by-step example of the algorithm decomposing a sequence $seq_{16} = 00100100\ 10111101$ of length $l = 16$ can be seen in Table 3.

Algorithm 1 Basic Binomial Sequence Decomposition (b-BSD).

Require: seq : intercepted bits
  $binom = [\emptyset]$
  for $i = 0$; $i < length(seq)$; i++ **do**
    if $seq_i \neq 0$ **then**
      $seq+ = \binom{n}{i}$
      $binom.add(i)$
    end if
  end for
  return $binom$: Binomial decomposition of the intercepted bits

Table 3. Step by step b-BSD example on seq_{16}.

Step	Op.	Seq.	Bit Position			
			0	**4**	**8**	**12**
0		seq	0 0 1 0	0 1 0 0	1 0 1 1	1 1 0 1
	+	$\binom{n}{2}$	0 0 1 1	0 0 1 1	0 0 1 1	0 0 1 1
1	=	seq	0 0 0 1	0 1 1 1	1 0 0 0	1 1 1 0
	+	$\binom{n}{3}$	0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1
2	=	seq	0 0 0 0	0 1 1 0	1 0 0 1	1 1 1 1
	+	$\binom{n}{5}$	0 0 0 0	0 1 0 1	0 0 0 0	0 1 0 1
3	=	seq	0 0 0 0	0 0 1 1	1 0 0 1	1 0 1 0
	+	$\binom{n}{6}$	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	1 0 0 1
	+	$\binom{n}{8}$	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	0 1 1 0
	+	$\binom{n}{9}$	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	0 0 1 1
	+	$\binom{n}{10}$	0 0 0 0	0 0 0 0	0 0 1 1	0 0 1 1
end	=	seq	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0

$$seq = \binom{n}{2} + \binom{n}{3} + \binom{n}{5} + \binom{n}{6} + \binom{n}{8} + \binom{n}{9} + \binom{n}{10}$$
$$LC = k_t + 1 = 10 + 1 = 11.$$

Thus, the b-BSD algorithm is able to calculate LC, as the Berlekamp–Massey algorithm does, but after having processed only LC bits of the sequence instead of $2 * LC$. The complexity of b-BSD

algorithm, which performs the sum of two sequences of l bits (l additions) for every binomial sequence, is $O(t * l)$, t being the number of binomial sequences in which the main sequence is decomposed with $t \ll l$.

Moreover, the logic of the algorithm can be improved by avoiding the sum of the sub-sequences that are zero. On the one hand, thanks to item 1 (b) in Section 3.1, we know that $\binom{n}{k} = 0$, $\forall n < k$. On the other hand, at each step of the b-BSD algorithm, the sequence begins with zeros. That is, at step i, the k_i first terms of the sequence are zeros.

If these two facts are combined, then the number of operations performed by the algorithm can be reduced. When the algorithm detects the first 1 in the i-th position of seq, instead of performing the sum of two sequences of l bits ($seq + \binom{n}{i}$), it just sums both sequences between the i-th and $(l-1)$-th bits ($seq_{i,l-1} + \binom{n}{i}_{i,l-1}$), as the beginning of both sequences (from bit 0 up to bit $i-1$) are made up of zeros.

Compared with $t * l$, the number of operations is reduced as follows:

$$\sum_{k_i=k_0}^{k_t} (l - k_i) < t * l.$$

In addition, for sequences whose LC is upper bounded, we do not need to perform the sum of any binomial sequence after the binomial of maximum index is attained. This is the case for every GSS-sequence produced by the generalized self-shrinking generator. Since the linear complexity of this family of sequences satisfies the inequality:

$$LC \leq 2^{L-1} - (L-2),$$

and its maximum length is $l = 2^{L-1}$ (L being the number of stages in the LFSR that generates the GSS-sequences), then

$$LC \leq l - \log l + 1.$$

Therefore, the maximum index k_{max} in the BSD of these sequences is:

$$k_{max} = l - \log l, \tag{2}$$

and the final number of operations in the algorithm will be:

$$\sum_{k_i=k_0}^{k_t} (k_{max} - k_i) < \sum_{k_i=k_0}^{k_t} (l - k_i) < t * l.$$

Thus, at each algorithm step, the number of operations is incrementally reduced.

To upgrade the code in Algorithm 1, only the summation limits must be changed, which will now be $seq = seq_{i,k_{max}} + \binom{n}{i}_{i,k_{max}}$, with k_{max} given in Equation (2).

In brief, the b-BSD algorithm will require at most $(l - \log l)$ bits of the sequence to calculate the LC with a complexity less than $O(t * l)$.

4.3. Folding Binomial Sequence Decomposition (f-BSD): Theoretical Foundations

Despite the improvement in both complexity and length requirements between b-BSD and Berlekamp–Massey, there is still room for enhancing the decomposition mechanism.

In the next subsection, a new algorithm design is explained, improving the results of the b-BSD algorithm by taking advantage of the symmetry of the binomial sequences.

In this subsection, we develop new properties of the binomial sequences, particularly symmetric properties, on which the f-BSD algorithm is based.

First of all, a result that relates the index k of a binomial sequence with the period of such a sequence is introduced.

Theorem 1. *For every binomial sequence* $\binom{n}{k}$ *with* $k > 0$*, there exists an integer* $m > 0$*, such that* 2^m *is the period of the binomial sequence, as well as the index k satisfying the inequality:*

$$2^{m-1} \leq k < 2^m.$$

Proof. In fact, any integer $k > 0$ can be written in the range $2^{m-1} \leq k < 2^m$, where 2^{m-1} and 2^m are two consecutive powers of 2 and $m \geq 1$. In this case,

$$k = 2^{m-1} + j, \tag{3}$$

j being an integer in the interval $0 \leq j < 2^{m-1}$. Thus, the binomial sequence can be written as:

$$\binom{n}{k} = \binom{n}{2^{m-1} + j},$$

and, according to item 1. (b) in Section 3.1, its period will be $l = 2^m$. □

Therefore, the index k determines the period of the binomial sequence $\binom{n}{k}$. Moreover, recall that in Equation (3)

- if $j = 0$, then k takes the minimum value $k = 2^{m-1}$,
- if $j = 2^{m-1} - 1$, then k takes the maximum value $k = 2^m - 1$.

Thus, k ranges in the interval $k \in [2^{m-1}, 2^m - 1]$ and the next corollary follows directly:

Corollary 1. *The number of different binomial sequences* $\binom{n}{k}$ *with the same period* ($l = 2^m$) *is* 2^{m-1}.

Symmetry of $\binom{n}{k}$

There are different properties regarding the symmetric structure of the binomial sequences and their relation to the powers of 2 that are explained in the following.

Every binomial sequence $\binom{n}{k}$ with length $l = 2^m$ ($m > 0$) can be divided into two sub-sequences of length $\frac{l}{2}$ as follows:

$$\binom{n}{k}_l = \left(\binom{n}{k}_{0, \frac{l}{2}-1}, \binom{n}{k}_{\frac{l}{2}, l-1} \right). \tag{4}$$

On the other hand, recall that the binomial sequence $\binom{n}{k}$ starts with k zeros. Therefore, if $k \geq \frac{l}{2}$, then such a sequence can be divided in the following way:

$$\binom{n}{k}_l = (zeros_{\frac{l}{2}}, \binom{n}{k}_{\frac{l}{2}, l-1}),$$

where $zeros_{\frac{l}{2}}$ represents the sub-sequence identically null of length $\frac{l}{2}$.

In fact, this is a simplification of a stronger result, defined in the next theorem.

Theorem 2. *Every binomial sequence* $\binom{n}{k}_l$ *can be divided into two sequences of length* $\frac{l}{2}$ *as follows:*

1. *If* $k \geq \frac{l}{2}$*, then*

$$\binom{n}{k}_l = \left(zeros_{\frac{l}{2}}, \binom{n}{k - \frac{l}{2}}_{\frac{l}{2}} \right).$$

2. *If* $\frac{l}{2} < k$, *then*

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

Proof. 1. Since $k \geq \frac{l}{2}$, it can be written as $k = 2^{m-1} + i$ with $0 \leq i < 2^{m-1}$. Then,

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

2. Since $k < \frac{l}{2} = 2^{m-1}$, it can be written as $k = 2^j + i$ with $0 \leq i < 2^j$, where i and j are integers as well as $j < m-1$. The binomial sequence $\binom{n}{k} = \binom{n}{2^j+i}$ has length 2^{j+1}, that is a power of 2, as well as $2^{j+1} < 2^m$. Thus, the first and second sub-sequences in Equation (4) are equal.

□

In Table 4, where the binomial decomposition of seq_{16} is represented with $\frac{l}{2} = 8$, the condition 1 of the Theorem 2 is observable on the binomial sequences $\binom{n}{8}$, $\binom{n}{9}$ and $\binom{n}{10}$. In contrast, the binomial sequences $\binom{n}{2}$, $\binom{n}{3}$, $\binom{n}{5}$ and $\binom{n}{6}$ satisfy the condition 2, where the eight first bits repeat themselves.

Table 4. Binomial Sequence Decomposition at a glance.

Seq	**0 0 1 0**	**0 1 0 0**	**1 0 1 1**	**1 1 0 1**
$\binom{n}{2} = (\binom{n}{2}_{\frac{l}{2}}, \binom{n}{2}_{\frac{l}{2}})$	0 0 1 1	0 0 1 1	0 0 1 1	0 0 1 1
$\binom{n}{3} = (\binom{n}{3}_{\frac{l}{2}}, \binom{n}{3}_{\frac{l}{2}})$	0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1
$\binom{n}{5} = (\binom{n}{5}_{\frac{l}{2}}, \binom{n}{5}_{\frac{l}{2}})$	0 0 0 0	0 1 0 1	0 0 0 0	0 1 0 1
$\binom{n}{6} = (\binom{n}{6}_{\frac{l}{2}}, \binom{n}{6}_{\frac{l}{2}})$	0 0 0 0	0 0 1 1	0 0 0 0	0 0 1 1
$\binom{n}{8} = (zeros_{\frac{l}{2}}, \binom{n}{0}_{\frac{l}{2}})$	0 0 0 0	0 0 0 0	1 1 1 1	1 1 1 1
$\binom{n}{9} = (zeros_{\frac{l}{2}}, \binom{n}{1}_{\frac{l}{2}})$	0 0 0 0	0 0 0 0	0 1 0 1	0 1 0 1
$\binom{n}{10} = (zeros_{\frac{l}{2}}, \binom{n}{2}_{\frac{l}{2}})$	0 0 0 0	0 0 0 0	0 0 1 1	0 0 1 1
$seq = \binom{n}{2} + \binom{n}{3} + \binom{n}{5} + \binom{n}{6} + \binom{n}{8} + \binom{n}{9} + \binom{n}{10}$				

Putting together the facts regarding the symmetry of the binomial sequences, they can be classified in two groups depending on the value of their index. This fact is explained in Algorithm 2.

Algorithm 2 Binomial Sequences Classification.

For a given 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

If the binomial sequences are divided as seen in Algorithm 2, then the binomial representation of a sequence results in a block matrix $M = \left(\begin{array}{cc} M_0 & M_1 \\ \hline M_2 & M_3 \end{array}\right)$, that is, a matrix representation of the BSD, as follows:

$$\begin{pmatrix} \binom{n}{k_0} \\ \vdots \\ \binom{n}{k_t} \end{pmatrix} = \left(\begin{array}{c} \binom{n}{k_0} \\ \vdots \\ \binom{n}{k_{i-1}} \\ \hline \binom{n}{k_i} \\ \vdots \\ \binom{n}{k_t} \end{array}\right)_{k_{i-1}<\frac{l}{2}\leq k_i} = \left(\begin{array}{c|c} \binom{n}{k_0}_{\frac{l}{2}} & \binom{n}{k_0}_{\frac{l}{2}} \\ \vdots & \vdots \\ \binom{n}{k_{i-1}}_{\frac{l}{2}} & \binom{n}{k_{i-1}}_{\frac{l}{2}} \\ \hline 0_{\frac{l}{2}} & \binom{n}{k_i-\frac{l}{2}}_{\frac{l}{2}} \\ \vdots & \vdots \\ 0_{\frac{l}{2}} & \binom{n}{k_t-\frac{l}{2}}_{\frac{l}{2}} \end{array}\right) = \left(\begin{array}{c|c} M_0 & M_1 \\ \hline M_2 & M_3 \end{array}\right). \quad (5)$$

Three important characteristics about the matrix representation shown in (5) are the core of the folding BSD algorithm.

- $M_0 = M_1$.
- $M_2 = 0$.
- As the length of the sequence is of the form $l = 2^m$, the matrix representation can be extended in a recursive way, taking M_3 and repeating the same process until it cannot be divided anymore (length = 1).

$$\left(\begin{array}{c|c} M_0 & M_1 \\ \hline M_2 & M_3 \end{array}\right) = \left(\begin{array}{c|cc} M_0 & \multicolumn{2}{c}{M_1} \\ \hline M_2 & M_{3,0} & M_{3,1} \\ & M_{3,2} & M_{3,3} \end{array}\right) = \left(\begin{array}{c|cc} M_0 & \multicolumn{2}{c}{M_1} \\ \hline & M_{3,0} & \multicolumn{1}{c}{M_{3,1}} \\ M_2 & \multicolumn{2}{c}{\begin{array}{cc} M_{3,3,0} & M_{3,3,1} \\ M_{3,3,2} & M_{3,3,3} \end{array}} \\ & M_{3,2} & \end{array}\right) = \ldots \quad (6)$$

The following expression (7) is an example of the matrix representation of the sequence decomposition of Table 4.

$$\begin{pmatrix} \binom{n}{2} \\ \binom{n}{3} \\ \binom{n}{5} \\ \binom{n}{6} \\ \binom{n}{8} \\ \binom{n}{9} \\ \binom{n}{10} \end{pmatrix} = \left(\begin{array}{cc|cc} 0011 & 0011 & 0011 & 0011 \\ 0001 & 0001 & 0001 & 0001 \\ 0000 & 0101 & 0000 & 0101 \\ 0000 & 0011 & 0000 & 0011 \\ \hline 0000 & 0000 & 1111 & 1111 \\ 0000 & 0000 & 0101 & 0101 \\ 0000 & 0000 & 0011 & 0011 \end{array}\right) = \left(\begin{array}{c|c} M_0 & M_1 \\ \hline M_2 & M_3 \end{array}\right) \rightarrow M_3 = \left(\begin{array}{c|c} 1111 & 1111 \\ 0101 & 0101 \\ 0011 & 0011 \\ \hline \emptyset & \emptyset \end{array}\right) = \ldots \quad (7)$$

In order to calculate the *LC* of the given sequence, only the highest binomial sequence of its decomposition is needed. Thus, the f-BSD algorithm will benefit from the symmetry of the binomial sequences, by reducing recursively the length of the sequence to analyze, as depicted in the matrix expression (6) .

4.4. Folding Binomial Sequence Decomposition Algorithm

The previous subsection described all the elements needed by the f-BSD algorithm. In fact, the algorithm locates the maximum binomial sequence to calculate *LC*. At every step, it sums the first half of the sequence with the second half. If the result is different from zero, then it continues with the resulting sequence. Otherwise, it continues with half the previous sequence. The procedure ends when only one bit is left.

At every step, the folding mechanism reduces the length of the studied sequence by 2. It performs a bit sum of the successive halves of the sequence, with a total of $\log l$ steps. Given a sequence *seq* with length l, the number of operations of the algorithm can be calculated as follows:

$$\frac{l}{2} + \frac{\frac{l}{2}}{2} + \frac{\frac{l}{4}}{2} + \cdots = \frac{l}{2} + \frac{l}{4} + \frac{l}{8} + \cdots = \sum_{i=0}^{\log l} \frac{l}{2^i} \approx l$$

The final pseudo-code of the algorithm, for a given binary sequence of length l (although we can reduce it to $(l - \log l)$, as explained in Section 4.2) and complexity O(l), can be found in Algorithm 3. The way that this algorithm searches for the maximum binomial sequence is similar to that of the binary search algorithm. The difference results in the binary search only performing one comparison in each step, while our algorithms need to perform $\frac{length(l)}{2^i}$ operations per step.

Let us see how it works with an example.

Example 2. *Take the same sequence as before* $seq_{16} = 00100100\ 10111101$.

- *Step 1:*
$$\begin{array}{rcccc} & 0010 & 0100 & & \\ + & 1011 & 1101 & & \\ \hline & 1001 & 1001 & & \end{array}$$
As $aux = 1001\,1001 \neq 0_{0,7}$, *then* $seq = aux = 1001\,1001$ *and* $k = 8$.
- *Step 2:*
$$\begin{array}{rcccc} & & 10 & 01 & \\ + & & 10 & 01 & \\ \hline aux = & & 00 & 00 & \end{array}$$
As $aux = 0_{0,3}$, *then* $seq = 1001$.
- *Step 3:*
$$\begin{array}{rcccc} & & 1 & 0 & \\ + & & 0 & 1 & \\ \hline aux = & & 1 & 1 & \end{array}$$
As $aux \neq 0_{0,1}$, *then* $seq = aux = 1\,1$ *and* $k = 8 + 2$.
- *Step 4:*
$$\begin{array}{rcc} & & 1 \\ + & & 1 \\ \hline aux = & & 0 \end{array}$$
As $aux = 0$, *then* $seq = aux = 0$.
- *End: the maximum binomial sequence is* $\binom{n}{10} \rightarrow LC = k + 1 = 10 + 1 = 11$.

A representation of the algorithm workflow can be seen in Figure 3.

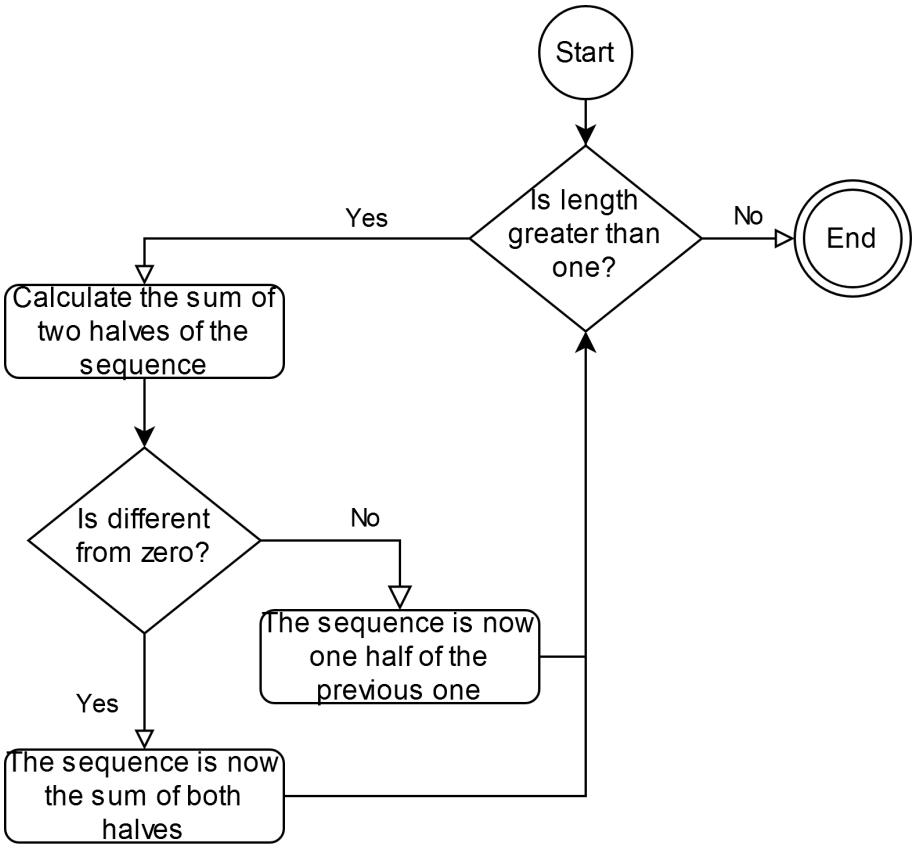


Figure 3. f-BSD algorithm flowchart.

Algorithm 3 Folding Binomial Sequences Classification.

Require: seq : intercepted bits
  $k = 0$
  while $length(seq) > 1$ **do**
    $l = length(seq)$
    aux = $seq_{\frac{l}{2},l-1}$ + $seq_{0,\frac{l}{2}-1}$
    if $aux \neq 0_{\frac{l}{2}}$ **then**
      $seq = aux$
      $k+ = \frac{l}{2}$
    else
      $seq = seq_{0,\frac{l}{2}-1}$
    end if
  end while
  return k: maximum binomial sequence

5. Algorithm Discussion

The three algorithms for linear complexity calculation we have visited in our work have different properties and capabilities. Thus, it is worth discussing all of them. In Table 5, there is a simple comparison between the sequence length required and the computational complexity in each one of these algorithms.

Table 5. Algorithm comparison.

Algorithms	Length Required	Complexity
Berlekamp–Massey	$2 \times l$	O(l^2)
b-BSD	$l - \log l$	O($t \times l$)
f-BSD	$l - \log l$	O(l)

The Berlekamp–Massey algorithm is the classic algorithm to calculate the linear complexity of any sequence. Nevertheless, for sequences whose LC is near their length, e.g., the GSS-sequences, it needs twice the length l of the sequence and its computational complexity is O(l^2). When it is applied to LFSR-based sequences, this result can become impractical, particularly when these sequences may exhibit a length of 2^{128} bits, for instance, the GSS-sequences in cryptographic applications.

Concerning the basic binomial sequence decomposition algorithm (b-BSD), it allows an improvement in the amount of sequence required regarding the Berlekamp–Massey algorithm. Nevertheless, its computational complexity can be tricky to assess, because it depends on the number t of binomial sequences that appear in the BSD of the sequence under consideration.

5.1. Number of Sequences in the Decomposition

The number of sequences in the binomial decomposition, t, has not been studied previously in the literature. This is one of the facts that makes it difficult to measure the actual improvement of the b-BSD algorithm, whose computational complexity depends on this specific parameter.

In order to study the behavior of t and its dependency on the original sequence, some experiments were carried out.

In our experiments, all the employed sequences were the GSS-sequences coming from LFSRs with polynomials of degrees between five and ten. As a result, we observed on average a number of binomial sequences given by 2^{n-2}, n being the degree of the LFSR polynomials $\forall n \in [5, 10]$. The plots corresponding to the number of binomial sequences in the decomposition of all these GSS-sequences are depicted in the Figure 4.

The distribution of the number of sequences in a binomial decomposition seems to be close to a normal distribution, although a thin tail on the left can be noticed, which means that for a few sequences, the number of binomial sequences will be lower. In those particular cases, this algorithm will perform better than in the general case.

At any rate, recall that this result $t \approx 2^{n-2}$ is backed up by experimental results and no mathematical proof exists regarding the t parameter behavior.

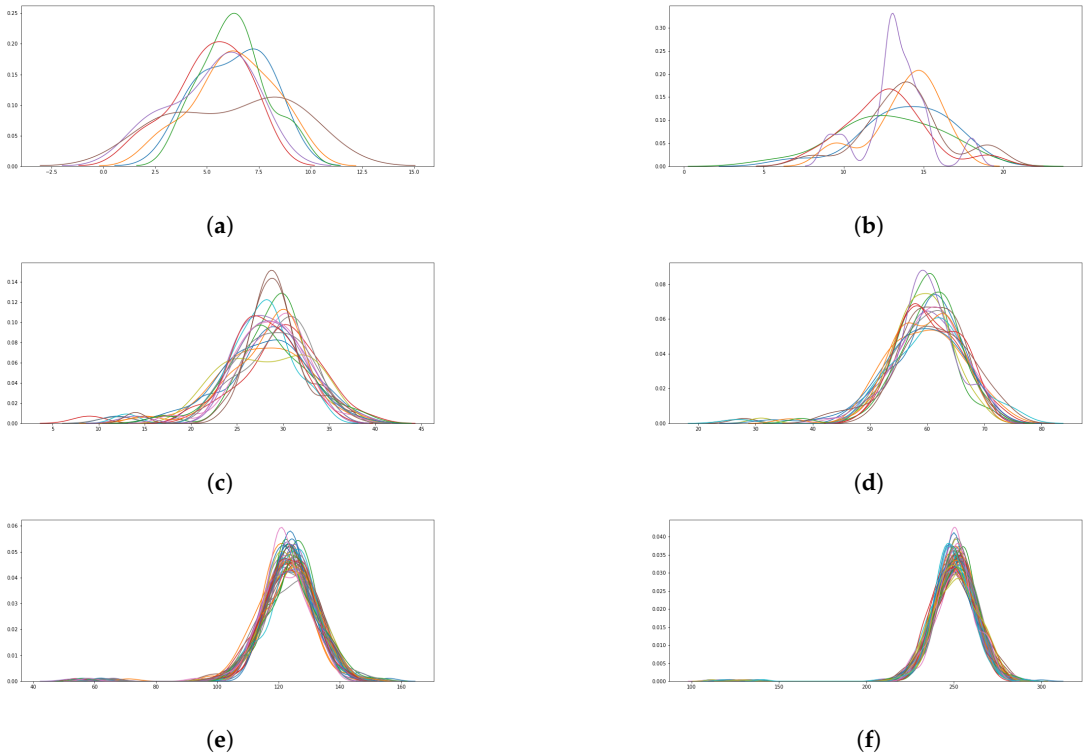


Figure 4. Number of Binomial Sequences in the decomposition $\forall n \in [5, 10]$. (**a**) $n = 5$. (**b**) $n = 6$. (**c**) $n = 7$. (**d**) $n = 8$. (**e**) $n = 9$. (**f**) $n = 10$.

5.2. f-BSD Performance

On the other hand, the folding binomial sequence decomposition algorithm (f-BSD) has a set of characteristics that improves its performance when compared with both Berlekamp–Massey and b-BSD algorithms. In fact, it requires the same length as that of the b-BSD algorithm and improves considerably this requirement when it is compared with the Berlekamp–Massey algorithm. Another interesting characteristic is that its computational complexity does not depend on the number t of binomial sequences as the b-BSD algorithm does, which introduces a penalty factor of about 2^{n-2} in the b-BSD algorithm performance.

The reason for its improvement is that, at each step of the algorithm, the length of the resulting sequence is divided by two, as can be seen in Figure 3. Consequently, the number of operations to be performed and the memory needed are progressively reduced.

Although the final purpose of this work is not the implementation of the different algorithms, it is worth pointing out that the f-BSD algorithm can be transformed into a concurrent algorithm for the linear complexity calculation. This fact theoretically improves its performance when the f-BSD algorithm is implemented in an environment with concurrent computation capabilities, as opposed to the b-BSD algorithm, which performs its calculations in a sequential way and cannot operate in a concurrent fashion.

Finally, a comparison of our work and previous works [26,27] can be summarized in Table 6.

In such a table, we point out the main contributions of our current work compared to past works: the introduction of the f-BSD algorithm, the complexity analysis of b-BSD (already introduced in [13]), backed up by our experiments on the number of sequences in the BSD, and the complexity analysis of f-BSD, which in the end shows how the f-BSD outperforms the previous algorithms.

Table 6. Comparison between proposed and existing algorithms to calculate LC.

Authors	Year	Objective	Merits	Demerits	5G-Focused
Berlekamp et al. [12] (Berlekamp–Massey algorithm)	1969	To calculate the linear complexity of binary sequences.	Suitable for sequences of any length.	High requirements of sequence length. High requirements of memory during execution.	N
Cardell et al. [13] (b-BSD algorithm)	2019	To calculate the linear complexity of binary sequences.	Improvement in sequence length requirements.	Applicable to sequences with period a power of 2. Dependence on t parameter. No experimental results of t Sequential.	Y
This proposal (f-BSD algorithm)	2020	To calculate the linear complexity of binary sequences.	Improvement in sequence length requirements. Independence on t. Concurrent. It outperforms previous algorithms.	Applicable to sequences with period a power of 2.	Y

6. Conclusions

In this work, a new algorithm, the folding BSD algorithm, has been introduced and developed. It exhibits a better performance than similar algorithms to compute the lineal complexity of a binary sequence, especially on sequences that are particularly difficult to be decomposed by the Berlekamp–Massey algorithm. This is a big step in the study of the binary sequences with period a power of two, and makes it easier to detect flaws in this kind of sequences. Detecting such vulnerabilities in a cipher implemented in practical applications could compromise the corresponding IoT devices and the services behind them.

Moreover, the binomial decomposition of sequences as a way to extract information from a given sequence is an innovative but powerful tool, and it is left for future work its application to other kinds of binary sequences.

An experimental approximation to the complexity of the b-BSD algorithm based on the density of binomial sequences has been introduced, but this topic requires further research to find hard proof on the number of sequences that appear in a binomial decomposition.

In regard to the f-BSD algorithm presented in this article, it shows better theoretical characteristics in both complexity and length of the sequence required. Future works may study the algorithm performance in real world scenarios by applying it to different binary sequences and taking advantage of the algorithm parallel capabilities.

To compare the performance of the novel f-BSD algorithm with b-BSD, we carried out experiments to analyze the number of decomposed sequences, as this parameter deeply affects the performance of the b-BSD algorithm. As a result of our experiments, we were able to approximate the value of the parameter and the performance of b-BSD, which helps to show how the new algorithm f-BSD improves its performance when the length of the sequence starts to grow.

Author Contributions: Conceptualization, J.L.M.-N. and A. F.-S.; methodology, J.L.M.-N. and A.F.-S.; software, J.L.M.-N.; validation, A.F.-S.; formal analysis, A.F.-S.; investigation, J.L.M.-N.; resources, A.F.-S.; data curation, J.L.M.-N. and A.F.-S.; writing—original draft preparation, A.F.-S.; writing—review and editing, J.L.M.-N. and

A.F.-S.; visualization, J.L.M.-N.; supervision, A.F.-S.; project administration, A.F.-S.; funding acquisition, J.L.M.-N. and A.F.-S. All authors have read and agreed to the published version of the manuscript.

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 Commun. Mag.* **2017**, *55*, 70–75. [CrossRef]
2. Meyer, D.; Haase, J.; Eckert, M.; Klauer, B. New attack vectors for building automation and IoT. In Proceedings of the IECON 2017 43rd Annual Conference of the IEEE Industrial Electronics Society, Beijing, China, 29 October–1 November 2017; pp. 8126–8131.
3. Gallegos-Segovia, P.L.; Bravo-Torres, J.F.; Argudo-Parra, J.J.; Sacoto-Cabrera, E.J.; Larios-Rosillo, V.M. Internet of things as an attack vector to critical infrastructures of cities. In Proceedings of the 2017 International Caribbean Conference on Devices, Circuits and Systems (ICCDCS), Cozumel, Mexico, 5–7 June 2017; pp. 117–120.
4. Rouf, I.; Miller, R.D.; Mustafa, H.A.; Taylor, T.; Oh, S.; Xu, W.; Gruteser, M.; Trappe, W.; Seskar, I. Security and Privacy Vulnerabilities of In-Car Wireless Networks: A Tire Pressure Monitoring System Case Study. In Proceedings of the USENIX Security Symposium, Washington, DC, USA, 11–13 August 2010.
5. Cynthia, J.; Sultana, H.P.; Saroja, M.; Senthil, J. Security protocols for IoT. In *Ubiquitous Computing and Computing Security of IoT*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 1–28.
6. Mavromoustakis, C.X.; Mastorakis, G.; Batalla, J.M. *Internet of Things (IoT) in 5G Mobile Technologies*; Springer: Berlin/Heidelberg, Germany, 2016; Volume 8.
7. NIST Lightweight Cryptography Project. Available online: https://csrc.nist.gov/Projects/Lightweight-Cryptography (accessed on 30 March 2020).
8. McGinthy, J.M. Solutions for Internet of Things Security Challenges: Trust and Authentication. Ph.D. Thesis, Virginia Tech, Blacksburg, VA, USA, 2019.
9. Peinado, A.; Munilla, J.; Fúster-Sabater, A. EPCGen2 Pseudorandom Number Generators: Analysis of J3Gen. *Sensors* **2014**, *14*, 6500–6515. [CrossRef] [PubMed]
10. Peinado, A.; Munilla, J.; Fúster-Sabater, A. Revision of J3Gen and Validity of the Attacks by Peinado et al. *Sensors* **2015**, *15*, 11988–11992. [CrossRef] [PubMed]
11. Cardell, S.D.; Requena, V.; Fúster-Sabater, A.; Orúe, A.B. Randomness Analysis for the Generalized Self-Shrinking Sequences. *Symmetry* **2019**, *11*, 1460. [CrossRef]
12. Massey, J. Shift-register synthesis and BCH decoding. *IEEE Trans. Inf. Theory* **1969**, *15*, 122–127. [CrossRef]
13. Cardell, S.D.; Fúster-Sabater, A. Binomial Representation of Cryptographic Binary Sequences and Its Relation to Cellular Automata. *Complexity* **2019**, *2019*, 2108014. [CrossRef]
14. Chang, K.Y.; Lee, M.K.; Lee, H.R.; Do, Won, H.O.N.G.; Kang, J.S.; Cho, H.S.; Chung, K.I. Method and Apparatus for Generating Keystream. U.S. Patent 7,587,046, 8 September 2009.
15. Kang, Y.S.; Kim, H.W.; Chung, K.I. Apparatus and Method for Protecting RFID Data. U.S. Patent 8,386,794, 26 February 2013.
16. Golomb, S.W. *Shift Register Sequences*; Aegean Park Press: Laguna Hills, CA, USA, 1967.
17. Cardell, S.D.; Fúster-Sabater, A. The t-Modified self-shrinking generator. In Proceedings of the International Conference on Computational Science, Krakow, Poland, 16–18 June 2021; Springer: Berlin/Heidelberg, Germany, 2018; pp. 653–663.
18. Cardell, S.D.; Fúster-Sabater, A. *Cryptography with Shrinking Generators: Fundamentals and Applications of Keystream Sequence Generators Based on Irregular Decimation*; Springer: Berlin/Heidelberg, Germany, 2019.
19. Coppersmith, D.; Krawczyk, H.; Mansour, Y. The shrinking generator. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 22–26 August 1993; Springer: Berlin/Heidelberg, Germany, 1993; pp. 22–39.

20. Meier, W.; Staffelbach, O. The self-shrinking generator. In *Communications and Cryptography*; Springer: Berlin/Heidelberg, Germany, 1994; pp. 287–295.
21. Hu, Y.; Xiao, G. Generalized self-shrinking generator. *IEEE Trans. Inf. Theory* **2004**, *50*, 714–719. [CrossRef]
22. Fúster-Sabater, A.; Cardell, S.D. Linear complexity of generalized sequences by comparison of PN-sequences. *Rev. Real Acad. Cienc. Exactas Fis. Nat. Ser. A Matemáticas* **2020**, *114*, 79. [CrossRef]
23. Cardell, S.D.; Climent, J.J.; Fúster-Sabater, A.; Requena, V. Representations of Generalized Self-Shrunken Sequences. *Mathematics* **2020**, *6*, 1006. [CrossRef]
24. Fúster-Sabater, A. Generation of cryptographic sequences by means of difference equations. *Appl. Math. Inf. Sci.* **2014**, *8*, 475. [CrossRef]
25. Cusick, T.W.; Stanica, P. *Cryptographic Boolean Functions and Applications*; Academic Press: Cambridge, MA, USA, 2017.
26. Martin-Navarro, J.L.; Fúster-Sabater, A. Folding-BSD Algorithm for Binary Sequence Decomposition. In Proceedings of the International Conference on Computational Science and Its Applications, Santa Barbara, CA, USA, 22–26 August 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 345–359.
27. Martín-Navarro, J.L.; Fúster-Sabater, A.; Cardell, S.D. An Innovative Linear Complexity Computation for Cryptographic Sequences. In Proceedings of the Conference on Complex, Intelligent, and Software Intensive Systems, Loze, Poland, 1–3 July 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 339–349.