

Folding-BSD Algorithm for Binary Sequence Decomposition

Jose Luis Martin-Navarro and Amparo Fúster-Sabater(✉)

Instituto de Tecnologías Físicas y de la Información, C.S.I.C.,
Serrano 144, 28006 Madrid, Spain
jomarna6@inf.upv.es, amparo@iec.csic.es

Abstract. The 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 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, key part of the IoT security stack.

Keywords: PRNG · LFSR · Binomial sequences · Stream ciphers · IoT

1 Introduction

Breakthroughs in electronics and telecommunications fields made Internet of Things (IoT) a reality with an every day growing number of sensors and devices around us. Nowadays, diverse critical services 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 constrains 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).

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. The first author was supported by JAE "Introduction to research scholarships for students" of the Spanish Ministry of Science and Innovation.

© Springer Nature Switzerland AG 2020
O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12249, pp. 345–359, 2020.
https://doi.org/10.1007/978-3-030-58799-4_26

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.

In this work, first we will introduce Linear Feedback Shift Registers (LFSR), key components in stream ciphers, often used as Pseudo Random Number Generators (PRNG). 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 [9]. Then, we will analyze an innovative sequence decomposition introduced by Cardell et al. in [10] and will show how it can be used to analyze the properties of binary sequences. Finally, we will propose a novel algorithm based on the symmetry of the Binomial Sequences (BS) and discuss the comparison among algorithms.

The study of the generalized shelf shrinking generator is not a random choice. Indeed, it produces not only sequences that are hard to analyzed by the Berlekamp-Massey algorithm, but also it has been implemented in hardware [11] along on RFID devices [12]. Studying the robustness of these sequences could prevent vulnerabilities on the IoT devices and the services built on them.

2 LFSR-Based Sequence Generators: The Generalized Self-shrinking Generator

Linear Feedback Shift Registers (LFSRs) [13] are linear structures currently used in the generation of pseudorandom sequences. The LFSRs are electronic devices included in most of the sequence generators proposed in the literature. Main reasons for a so generalized use are: LFSRs provide high performance when used for sequence generation, they are particularly well-suited to hardware implementations and such registers can be readily analysed by means of algebraic techniques.

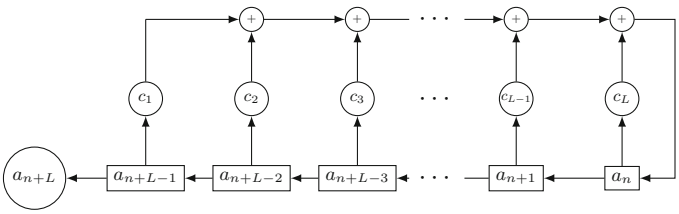

Fig. 1. LFSR of length L

According to Fig. 1, a LFSR consists of L interconnected stages numbered $(0, 1, \ldots, L-1)$ (from left to right) able to store one bit, the 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 binary coefficients c_i and the initial state (stage contents at the initial instant). In addition, a clock controls the data shift. LFSRs generate sequences by means of shifts and linear feedbacks.

The output of an LFSR with nonzero initial state is a binary sequence notated $\{a_n\}$ $(n = 0, 1, 2, \ldots)$. If the polynomial $p(x)$ is primitive [13], then the output sequence is called PN-sequence (pseudo-noise sequence). Moreover, a PN-sequence has period $T = 2^L - 1$ bits with 2^{L-1} ones and $2^{L-1} - 1$ zeros. On the other hand, linear complexity (LC) of a sequence is a parameter closely related to the LFSR. In fact, LC is defined as the length of the shortest LFSR able to generate such a sequence.

Although an LFSR in itself is an excellent generator of pseudorandom sequence, nevertheless it has undesirable linearity properties which reduce the security of its use. In practice, the introduction of any type of nonlinearity in the process of generation of the pseudorandom sequence is needed. The irregular decimation of PN-sequences is one of the most popular techniques to destroy the inherent linearity of the LFSRs [14,15].

Inside the type of irregularly decimated generators, we can enumerate: a) the shrinking generator introduced in [16] that includes two LFSRs, b) the self-shrinking generator [17] that involves just one LFSR and c) the generalized self-shrinking generator proposed in [18] that is considered as a generalization of the self-shrinking generator as well as a simplification of the shrinking generator. In this work, we focus on the last type of generator that is the generalized self-shrinking generator.

2.1 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, $\{v_n\} = \{a_{n+p}\}$ corresponds to the own sequence $\{a_n\}$ but rotated cyclically p positions to the left with $(p = 0, 1, \ldots, 2^L - 2)$.
2. It relates both sequences by means of a simple decimation rule to generate the output sequence.
   For $n \geq 0$, we define the decimation rule 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

Table 1. GSS-sequences for an LFSR with polynomial $p(x) = x^4 + x + 1$

shift p	$\{v_n\}$ sequences	GSS-sequences	shift p	$\{v_n\}$ sequences	GSS-sequences
0	**1111**0001**0011**0**1**0	11111111	8	**0011**0101**1110**0**0**1	00111100
1	**1110**0010**0110**1**0**1	11100100	9	**0110**1011**1100**0**1**0	01101001
2	**1100**0100**1101**0**1**1	11000011	10	**1101**0111**1000**1**0**0	11011000
3	**1000**1001**1010**1**1**1	10001101	11	**1010**1111**0001**0**0**1	10101010
4	**0001**0011**0101**1**1**1	00011011	12	**0101**1110**0010**0**1**1	01010101
5	**0010**0110**1011**1**1**0	00100111	13	**1011**1100**0100**1**1**0	10110001
6	**0100**1101**0111**1**0**0	01001110	14	**0111**1000**1001**1**0**1	01110010
7	**1001**1010**1111**0**0**0	10010110	−−	**0000**0000**0000**0**0**0	00000000
	11110001**0011**0**1**0			**1111**0001**0011**0**1**0	

$\{a_n\}$ remains fixed while $\{v_n\}$ is the sliding sequence or left-shifted version of $\{a_n\}$. When 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} of divisors, that is a power of 2. Next a simple example 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-sequences associated to their corresponding shifts p. The PN-sequence $\{a_n\}$ with period $T = 2^4 - 1$ is written at the bottom of the table.

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 2^{L-1}. This property is desirable because such sequences exhibit a great LC with low resources. As a result, the Berlekamp-Massey algorithm would need to process 2^L bits with a computational complexity of $O(2^{2*L})$.

3 Binomial Sequences and Sierpinski's Triangle

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

3.1 Binomial Sequences

The binomial number $\binom{n}{i}$ is the coefficient of the 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 $i > n$.

(a) Pascal's triangle

$$\binom{0}{0}$$
$$\binom{1}{0}\quad\binom{1}{1}$$
$$\binom{2}{0}\quad\binom{2}{1}\quad\binom{2}{2}$$
$$\binom{3}{0}\quad\binom{3}{1}\quad\binom{3}{2}\quad\binom{3}{3}$$
$$\binom{4}{0}\quad\binom{4}{1}\quad\binom{4}{2}\quad\binom{4}{3}\quad\binom{4}{4}$$
$$\binom{5}{0}\quad\binom{5}{1}\quad\binom{5}{2}\quad\binom{5}{3}\quad\binom{5}{4}\quad\binom{5}{5}$$

(b) Sierpinski's triangle mod 2

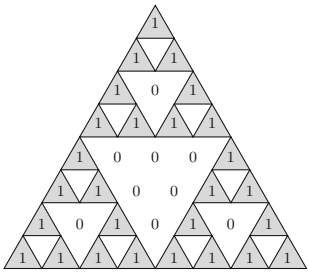

Fig. 2. Pascal and Sierpinski's triangles

The binomial coefficients reduced modulo 2 allow us to define the concept of binomial sequence.

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

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

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

Table 2 shows the eight first binomial sequences $\{\binom{n}{i}\}$, $i = 0, 1, \ldots, 7$, see [19], with their corresponding periods and linear complexities, denoted by T_i and LC_i, respectively.

Next, different properties of the binomial sequences are introduced.

1. Given the binomial sequence $\left\{\binom{n}{2^L+k}\right\}$, with $0 \leq k < 2^L$, we have that [10, Proposition 1.b]:
   a) Such a binomial sequence has period $T = 2^{L+1}$.

b) The first period of such a binomial sequence has the following structure:

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

2. The linear complexity of the sequence $\left\{\binom{n}{i}\right\}$ with $0 \leq i$ is $LC = i + 1$, see [10, Theorem 3].
3. Every binary sequence whose period is a power of 2 can be written as a linear combination of a finite number of binomial sequences [10, Theorem 1]. Such a combination is called the Binomial Sequence Decomposition.
4. Given a sequence with binomial representation $\sum_{k=0}^{t}\left\{\binom{n}{i_k}\right\}$, where $i_0 < i_1 < \cdots < i_t$ are integer indexes, then its linear complexity is given by $LC = i_t + 1$, see [19].
5. Given a sequence with binomial representation $\sum_{k=0}^{t}\left\{\binom{n}{i_k}\right\}$, 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 [19].

Notice that 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 $n = 0, 1, 2, \ldots$, then the generated structure is the Pascal's triangle (see Fig. 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 Fig. 2b.

Recall that the successive diagonals of the Sierpinski's triangle in Fig. 2a correspond to the successive binomial sequences $\left\{\binom{n}{i}\right\}$, $(i = 0, 1, 2, \ldots)$ starting at the first 1. That is, the binomial sequences can be found inside the Sierpinski's triangle mod 2, and also are related to certain cellular automata [10].

4 Algorithms to Calculate the Linear Complexity

NOTATION: For the sake of simplicity, in 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 $j-1$ while $\binom{n}{k}_{j}$ is just for the case $i = 0$.

The general method of computing the linear complexity of any sequence is the Berlekamp-Massey algorithm [9]. In order to work, this algorithm needs to process $2*n$ bits of the sequence with a computational complexity of $O(n^2)$ [20].

This section first introduces in detail the basic Binomial Sequence Decomposition algorithm (b-BSD) as well as an improvement on the algorithm implementation.

Second, 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.

Finally, a comparison among Berlekamp-Massey, b-BSD and f-BSD algorithms is presented. Such a comparison allows us to discuss the improvement of the f-BSD algorithm presented in this article.

4.1 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 n can be decomposed in $t+1$ binomial sequences (third item in 3.1):
$$seq_n = \binom{n}{k_0} + \cdots + \binom{n}{k_t}.$$
- The lineal complexity of a sequence can be calculated from the maximum binomial sequence of its BSD (second item in Sect. 3.1). Since the binomial sequences are in order, then LC satisfies the following expression:
$$LC = max_{0 \leq i \leq t}\left(\binom{n}{k_i}\right) + 1 = k_t + 1.$$

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 $bit_i == 1$, it sums the sequence with the corresponding binomial sequence ($seq + \binom{n}{i}$) stopping when all the binomial sequences have been found. A step-by-step example of the algorithm decomposing a sequence of length 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 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15
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
$seq = \binom{n}{2} + \binom{n}{3} + \binom{n}{5} + \binom{n}{6} + \binom{n}{8} + \binom{n}{9} + \binom{n}{10}$						

Thus, the b-BSD algorithm is able to calculate the LC, as the Berlekamp-Massey algorithm does, but with only n bits of the sequence instead of $2*n$. The complexity of b-BSD algorithm, which performs the sum of two sequences of n bits (n additions) for every binomial sequence, is $O(t*n)$, t being the number of binomial sequences in which the main sequence is decomposed with $t \ll n$.

Moreover, the logic of the algorithm can be improved by avoiding the sum of the sub-sequence that are zero. On the one hand, thanks to the characteristic 1.b) of Sect. 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 algorithm operations 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 n bits ($seq_n + \binom{n}{i}$), it just sums both sequences between the $i-th$ and $(n-1)-th$ bits $(seq_{i,n} + \binom{n}{i}_{i,n})$, as the head of both sequences ($[0, i-1]$) is made up of zeros.

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

$$\sum_{k_i=k_1}^{k_t} (n - k_i) < t * n.$$

In addition, we do not need to perform the sum of any bit after the max binomial. The reason is that, for every binary sequence produced by the generalized self-shrinking generator, the maximum binomial sequence can be calculated as $k_{max} = n - \log n$. The final number of operations will be:

$$\sum_{k_i=k_1}^{k_t} (k_{max} - k_i) < \sum_{k_i=k_1}^{k_t} (n - k_i) < t * n.$$

To upgrade the code in Algorithm 1 only the sum of both sequences is changed, which will be now $seq = seq_{i,max} + \binom{n}{i}_{i,max}$, with $max = n - \log n$.

Summing it up, for a sequence with a characteristic polynomial of degree σ, with length $n = 2^{\sigma-1}$, the b-BSD algorithm will require $n - \log n$ bits of the sequence to calculate the LC with a complexity of $O(t * n)$.

4.2 Folding Binomial Sequence Decomposition (f-BSD)

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 sub-section 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 order to fully understand the f-BSD algorithm, a particular matrix representation of the decomposition, based on the symmetric properties of binomial sequences, is presented.

Symmetry of $\binom{n}{k}$: There are two properties regarding the symmetric structure of binomial sequences and their relation to the powers of 2 that are explained in the next sub-section.

In all the binomial sequences ($\binom{n}{k}$ with $k < \frac{n}{2}$ and n being the length of the sequence), it is possible to observe the following structure:

$$\binom{n}{k}_n = \left(\binom{n}{k}_{0,\frac{n}{2}}, \binom{n}{k}_{\frac{n}{2},n} \right).$$

In Table 4, where $\frac{n}{2} = 8$, this phenomenon is observable on binomial sequences $\binom{n}{2}$, $\binom{n}{3}$, $\binom{n}{5}$ and $\binom{n}{6}$, where the eight first bits repeat themselves.

In fact, this is a simplification of a stronger result, defined in Theorem 1.

Theorem 1. *For every binomial sequence $\binom{n}{k}$ there exists an integer $l \in \mathbb{N}$ such that 2^l is the period of the binomial sequence as well as satisfies the inequality*

$$\forall \binom{n}{k}, \exists\ l \in \mathbb{N} \implies 2^{l-1} < k \leq 2^l.$$

The result follows directly from item 1 in 3.1. On the other hand, recall that the binomial sequences $\binom{n}{k}$ with $\frac{n}{2} \leq k \leq n$ start with k zeros, so they can be divided in the following way: $\binom{n}{k} = zeros + seq = \left(zeros\ _{0,\frac{n}{2}}, \binom{n}{k}_{\frac{n}{2},n}\right)$.

From the item 1.b in Subsect. 3.1 there is an interesting characteristic of the sub-sequence $\binom{n}{k}_{\frac{n}{2},n}$, which can be converted in another binomial sequence with the following expression: $\binom{n}{k}_{\frac{n}{2},n} = \binom{n}{k-\frac{n}{2}}_{\frac{n}{2}}$. Taking into consideration the period of the sequence, the Theorem 2 arises:

Theorem 2. *Every binomial sequence* $\binom{n}{k}_n, k \leq n$ *with period* $2^l, 2^{l-1} < k \leq 2^l$, *can be divided into two binomial sequences of length* $\frac{n}{2}$ *as follows:*

$$\binom{n}{k}_n = \left(zeros_{2^{l-1}}, \binom{n}{k-2^{l-1}}_{2^{l-1},n}\right).$$

In particular, as the sequences analyzed have a length of n a power of 2 bits, they can be divided into two sequences of length $\frac{n}{2}$. Again, it can be observed in Table 4 on the binomial sequences $\binom{n}{8}$, $\binom{n}{9}$ and $\binom{n}{10}$.

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
$\binom{n}{2} = \binom{n}{2}_{\frac{n}{2}}, \binom{n}{2}_{\frac{n}{2}}$	0 0 1 1	0 0 1 1	0 0 1 1	0 0 1 1
$\binom{n}{3} = \binom{n}{3}_{\frac{n}{2}}, \binom{n}{3}_{\frac{n}{2}}$	0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1
$\binom{n}{5} = \binom{n}{5}_{\frac{n}{2}}, \binom{n}{5}_{\frac{n}{2}}$	0 0 0 0	0 1 0 1	0 0 0 0	0 1 0 1
$\binom{n}{6} = \binom{n}{6}_{\frac{n}{2}}, \binom{n}{6}_{\frac{n}{2}}$	0 0 0 0	0 0 1 1	0 0 0 0	0 0 1 1
$\binom{n}{8} = zeros_{\frac{n}{2}}, \binom{n}{0}_{\frac{n}{2}}$	0 0 0 0	0 0 0 0	1 1 1 1	1 1 1 1
$\binom{n}{9} = zeros_{\frac{n}{2}}, \binom{n}{1}_{\frac{n}{2}}$	0 0 0 0	0 0 0 0	0 1 0 1	0 1 0 1
$\binom{n}{10} = zeros_{\frac{n}{2}}, \binom{n}{2}_{\frac{n}{2}}$	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 their division. It is explained in Algorithm 2.

Algorithm 2. Binomial Sequences Classification

For a given sequence $\binom{n}{k}_n$:
if $k \leq \frac{n}{2}$ **then**
$\binom{n}{k}_n := (\binom{n}{k}_{\frac{n}{2}}, \binom{n}{k}_{\frac{n}{2}})$
else
$\binom{n}{k}_n := (zeros_{\frac{n}{2}}, (\binom{n}{k-\frac{n}{2}}_{\frac{n}{2}})$
end if

If the binomial sequences are divided as seen in Algorithm 2, then it results in the $\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)$ matrix representation of the BSD, as shown in (1).

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

Three important characteristics about the matrix representation shown in (1) form the core of the folding BSD algorithm.

- $M_0 = M_1$.
- $M_2 = 0$.
- As the length of the sequences is $n = 2^{\sigma-1}$, the matrix representation can be extended in a recursive way, taking M_3 and repeating the same process until it cannot be divided more (length = 1).

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

The following expression (3) is an example of the matrix representation of the sequence decomposition of Table 4.

$$\begin{pmatrix} \binom{n}{2} \\ \binom{n}{3} \\ \binom{n}{5} \\ \binom{n}{6} \\ \binom{n}{8} \\ \binom{n}{9} \\ \binom{n}{10} \end{pmatrix} = \left(\begin{array}{c|c} 0011\ 0011 & 0011\ 0011 \\ 0001\ 0001 & 0001\ 0001 \\ 0000\ 0101 & 0000\ 0101 \\ 0000\ 0011 & 0000\ 0011 \\ \hline 0000\ 0000 & 1111\ 1111 \\ 0000\ 0000 & 0101\ 0101 \\ 0000\ 0000 & 0011\ 0011 \end{array}\right) = \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right) \rightarrow D = \left(\begin{array}{c|c} 1111 & 1111 \\ 0101 & 0101 \\ 0011 & 0011 \\ \hline \emptyset & \emptyset \end{array}\right) = \ldots \quad (3)$$

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 (2).

4.3 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 second half of the sequence. Otherwise, it continues with the first half, finishing when only one bit is left.

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

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

The final pseudo-code of the algorithm, for a given binary sequence of length n (although we can reduce it to $n - \log n$ as explained in Sect. 4.1) and complexity $O(n)$, can be found in Algorithm 3. The way this algorithm searches for the maximum binomial sequence is similar to that of the binary search algorithm. The difference results in that the binary search only performs one comparison in each step, while our algorithms needs to sum $\frac{length(n)}{2^i}$ operations per step.

Lets see how it works with an example. Taking the sequence of the previous examples $seq = 00100100\ 10111101$

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

4.4 Algorithm Comparison

When putting together the three algorithms that can calculate the complexity of a sequence (Berlekamp-Massey, b-BSD and f-BSD), it is interesting to compare the computational complexity and length requirements of each of them as shown in Table 5.

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 $O(n^2)$. It is in that situation where the Binomial Sequence

Algorithm 3. Folding Binomial Sequences Classification

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

Table 5. Algorithm comparison

Algorithms	Length required	Complexity
Berlekamp-Massey	2 * n	O(n^2)
b-BSD	$n - \log n$	O($t * n$)
f-BSD	$n - \log n$	O(n)

Decomposition can be really useful, in particular the folding algorithm presented in this part of the work.

A particular difference between b-BSD and f-BSD is that the later performance does not depend on the number of binomial sequences in the decomposition. That means that in general its performance will be better, but it could depend on the particular sequence under study.

Although it is not the purpose of this work, it is worth saying that the f-BSD algorithm can be parallelized in the calculus of the LC of a given sequence, while b-BSD performs the calculus in a sequential way.

5 Conclusion

In this work, the folding-BSD algorithm has been introduced. It exhibits much better performance on sequences particularly hard to be decomposed by the Berlekamp-Massey algorithm. This is a big step in the study of binary sequences with period a power of two, and makes it easier to find vulnerabilities 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 kind of binary sequences. Also, there is still

room for improvement on how the fractal structure of the binomial sequences can be profited to decompose a binary sequence without handling the whole sequence.

About the f-BSD algorithm presented in this article, it shows a 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, taking advantage of the parallel capabilities of the algorithm.

References

1. Chin, W.-L., Li, W., Chen, H.-H.: Energy big data security threats in IOT-based smart grid communications. IEEE Commun. Mag. **55**(10), 70–75 (2017)
2. Meyer, D., Haase, J., Eckert, M., Klauer, B.: New attack vectors for building automation and IOT. In: IECON 2017–43rd Annual Conference of the IEEE Industrial Electronics Society, pp. 8126–8131
3. Gallegos-Segovia, P.L., et al.: Internet of things as an attack vector to critical infrastructures of cities. In: 2017 International Caribbean Conference on Devices, Circuits and Systems (ICCDCS), pp. 117–120 (2017)
4. Rouf, I., et al.: Security and privacy vulnerabilities of in-car wireless networks: a tire pressure monitoring system case study. In: USENIX Security Symposium, vol. 10 (2010)
5. Cynthia, J., Parveen Sultana, H., Saroja, M.N., Senthil, J.: Security protocols for IoT. In: Jeyanthi, N., Abraham, A., Mcheick, H. (eds.) Ubiquitous Computing and Computing Security of IoT. SBD, vol. 47, pp. 1–28. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-01566-4_1
6. Mavromoustakis, C.X., Mastorakis, G., Batalla, J.M. (eds.): Internet of Things (IoT) in 5G Mobile Technologies. MOST, vol. 8. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30913-2
7. Nist lightweight cryptography project. https://csrc.nist.gov/Projects/Lightweight-Cryptography. Accessed 30 Mar 2020
8. McGinthy, J.M.: Solutions for internet of things security challenges: trust and authentication. Ph.D. thesis, Virginia Tech (2019)
9. Massey, J.: Shift-register synthesis and BCH decoding. IEEE Trans. Inf. Theory **15**(1), 122–127 (1969)
10. Cardell, S.D., Fúster-Sabater, A.: Binomial representation of cryptographic binary sequences and its relation to cellular automata. Complexity (2019)
11. Chang, K.Y., et al.: Method and apparatus for generating keystream. US Patent 7,587,046, 8 September 2009
12. Kang, Y.-S., Kim, H.-W., Chung, K.-I.: Apparatus and method for protecting RFID data. US Patent 8,386,794, 26 February 2013
13. Golomb, S.W., et al.: Shift Register Sequences. Aegean Park Press, Laguna Hills (1967)
14. Cardell, S.D., Fúster-Sabater, A.: The modified self-shrinking generator? In: Shi, Y., et al. (eds.) ICCS 2018. LNCS, vol. 10860, pp. 653–663. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93698-7_50
15. Díaz Cardell, S., Fúster-Sabater, A.: Cryptography with Shrinking Generators. SM. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-12850-0

16. Coppersmith, D., Krawczyk, H., Mansour, Y.: The shrinking generator. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 22–39. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48329-2_3
17. Meier, W., Staffelbach, O.: The self-shrinking generator. In: Communications and Cryptography, pp. 287–295. Springer (1994). https://doi.org/10.1007/978-1-4615-2694-0_28
18. Hu, Y., Xiao, G.: Generalized self-shrinking generator. IEEE Trans. Inf. Theory **50**(4), 714–719 (2004)
19. Fúster-Sabater, A.: Generation of cryptographic sequences by means of difference equations. Appl. Math. Inf. Sci. **8**(2), 475 (2014)
20. Cusick, T.W., Stanica, P.: Cryptographic Boolean Functions and Applications. Academic Press, Cambridge (2017)