cryptography

Article

Computational Analysis of Interleaving PN-Sequences with Different Polynomials

Sara D. Cardell [1], **Verónica Requena** [2] and **Amparo Fúster-Sabater** [3,*]

[1] Centro de Matemática, Computação e Cognição, Universidade Federal do ABC (UFABC), Santo André 09210-580, Brazil; s.cardell@ufabc.edu.br
[2] Departament de Matemàtiques, Universitat d'Alacant, 03690 Alacant, Spain; vrequena@ua.es
[3] Instituto de Tecnologías Físicas y de la Información (ITEFI), C.S.I.C., 28006 Madrid, Spain
* Correspondence: amparo@iec.csic.es

Abstract: Binary PN-sequences generated by LFSRs exhibit good statistical properties; however, due to their intrinsic linearity, they are not suitable for cryptographic applications. In order to break such a linearity, several approaches can be implemented. For example, one can interleave several PN-sequences to increase the linear complexity. In this work, we present a deep randomness study of the resultant sequences of interleaving binary PN-sequences coming from different characteristic polynomials with the same degree. We analyze the period and the linear complexity, as well as many other important cryptographic properties of such sequences.

Keywords: PN-sequence; interleaved sequence; linear complexity; randomness

check for updates

Citation: Cardell, S.D.; Requena, V.; Fúster-Sabater, A. Computational Analysis of Interleaving PN-Sequences with Different Polynomials. *Cryptography* **2022**, *6*, 21. https://doi.org/10.3390/cryptography6020021

Academic Editor: Josef Pieprzyk

Received: 21 March 2022
Accepted: 21 April 2022
Published: 26 April 2022

1. Introduction

The rapid development and evolution of the internet have made possible the connectivity among many devices of daily use and, consequently, the irruption of the so-called Internet of Things (IoT). Moreover, many critical services as e-banking, e-govern, e-health or e-commerce are based on IoT infrastructures. As nowadays, the presence of such services grows exponentially, so do all risks associated with their security [1]. On the one hand, the IoT devices are currently characterized by their constrains in what processing power, size, memory and energy consumption are concerned [2]. On the other hand, they are also characterized by their minimum or non-existent security [3], since the vast majority of IoT devices have been designed without safety in mind. Combining the inherent lack of security of IoT infrastructures with their network dependability, the final effect is that IoT devices are a suitable target to compromise the whole network. This is the reason why 5G communications [4] or specific calls such as that of NIST for cryptography primitives [5] are addressing this essential topic. In this context, lightweight cryptography in general and stream ciphers in particular are the key stones on which certain communication protocols are being designed to guarantee security.

Stream ciphers are related with the idea of pseudo-randomness. In fact, the purpose of Pseudo-Random Numbers Generators (PRNGs) is to produce sequences of numbers that seem to behave as if they were generated randomly from a specified probability distribution. These numbers are sometimes called pseudo-random numbers to underline the fact that they are not truly random. The PRNGs must be fast and easy to be implemented in a computer, displaying small memory requirements and good statistical properties. The bit-wise Exclusive-OR logic operation between the original message and a pseudo-random bit sequence (key-stream sequence) preserves the confidentiality of the message in the traditional procedure of stream cipher. Other important security features, such as the integrity or authentication of the message, require additional mechanisms such as an MAC (Message Authentication Code) function to guarantee that the message is authentic and consequently its integrity checked. In brief, they are two different algorithms (confidentiality and authentication–integrity) that sometimes can be unified in the same scheme; see the requirements of the NIST call [5] for lightweight primitives. For this reason, the application of pseudo-random number generators for IoT is increasingly being studied [6,7]. In this work, we focus exclusively on the key-stream sequence and, consequently, on the confidentiality of the message.

Traditionally, the pseudo-random bit sequences with application in cryptography are generated by means of maximal-length Linear Feedback Shift Registers (LFSR) [8]. Their output sequences are the PN-sequences that exhibit good statistical properties. However, their linearity, i.e., their predictability, makes them vulnerable against cryptanalytic attacks. One common way to break this linearity is through irregular decimation, which has given rise to a wide family of decimation-based sequence generators. A representative element of this family is the shrinking generator, which decimates one PN-sequence according to the positions of the ones in another PN-sequence [9]. In [10], the authors proved that the output sequence of this generator, the so-called shrunken sequence, is made up of interleaving shifted versions of a single PN-sequence. Moreover, the shifts of the corresponding interleaved sequences can be easily deduced from the characteristic polynomials of the LFSRs, and this fact can be advantageously used to implement cryptanalytic attacks [11].

In [12], the authors proposed the interleaving of shifted versions of one single PN-sequence considering these shifts (different from the ones used in the shrunken sequence) as part of the key. This idea makes even more difficult the cryptanalysis of such sequences. However, depending on the initial state of the LFSR, some of the resultant sequences showed a high predictability, i.e., a low linear complexity.

A natural way to deal with the vulnerabilities of interleaving shifted versions of the same PN-sequence is to interleave different PN-sequences coming from different LFSRs. In this work, we propose a similar analysis to the one developed in [12] but considering the interleaving of different PN-sequences instead. The sequences here analyzed present the same pseudo-randomness properties as those of [12]; however, their linear complexity is quite higher. Furthermore, given several maximal-length LFSRs with the same length, the linear complexity of the resultant interleaved PN-sequences is fixed regardless of the initial states considered. We also perform a randomness analysis on the resultant sequences that shows that our sequences are better than the sequences obtained interleaving PN-sequences from the same LFSR, that is, interleaving shifted versions of the same PN-sequence.

This paper is organized as follows. In Section 2, we recall some basic concepts related to binary sequences, which are needed to understand the rest of the paper. In Section 3, we study the linear complexity and the characteristic polynomial of the sequences obtained interleaving PN-sequences from different LFSRs. Furthermore, in Section 4, we compare our sequences with the ones obtained from other sequence generators with similar parameters. In Section 5, we perform a deep randomness analysis of the obtained sequences. Finally, the paper ends in Section 6 with some conclusions and future work.

2. Preliminaries

Let $\mathbb{F}_2 = \{0,1\}$ be the Galois field of two elements, i.e., the binary field. Let $\{u_i\}_{i\geq 0} = \{u_0, u_1, u_2, \ldots\}$ be a binary sequence, that is, each term satisfies that $u_i \in \mathbb{F}_2$, for all $i \geq 0$. The sequence $\{u_i\}_{i\geq 0}$ (or simply $\{u_i\}$) is said to be periodic if there exists a positive integer T such that $u_{i+T} = u_i$, for all $i \geq 0$. This number T is known as the period of the sequence.

Let L be a positive integer and $a_0, a_1, \ldots, a_{L-1}$ elements of $\mathbb{F}_2$. The sequence $\{u_i\}$ is a binary L-th order linear recurring sequence if it satisfies

$$u_{i+L} = a_{L-1}u_{i+L-1} + a_{L-2}u_{i+L-2} + \cdots + a_1 u_{i+1} + a_0 u_i, \qquad i \geq 0 \tag{1}$$

The expression in Equation (1) is known as an L-th order linear recurrence relationship. The polynomial of degree L given by

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{L-1}x^{L-1} + x^L \in \mathbb{F}_2[x],$$

is called the characteristic polynomial of the linear recurrence relationship as well as the characteristic polynomial of $\{u_i\}$.

The generation of these linear recurrence sequences can be implemented by Linear Feedback Shift Registers (LFSRs) [8]. An LFSR of length L is a generator of binary sequence with L cell or stages interconnected. The terms $\{a_0, a_1, a_2, \ldots, a_{L-1}\}$ are binary coefficients assigned to the corresponding stages. The initial state (stage contents at round zero) is the seed, and since the register operates in a deterministic form, the resultant sequence is completely determined by the initial state. At each clock pulse, the binary content of each stage shifts one position to the left, and one bit is output from the register. The input of each round is a bit resultant from applying a linear transformation function to a previous state (see Figure 1). If the characteristic polynomial $p(x)$ is primitive, then the LFSR is said to be a maximal-length LFSR, and the resultant sequence, called a PN-sequence (or m-sequence), has period $T = 2^L - 1$ (with 2^{L-1} ones and $2^{L-1} - 1$ zeros) [8].

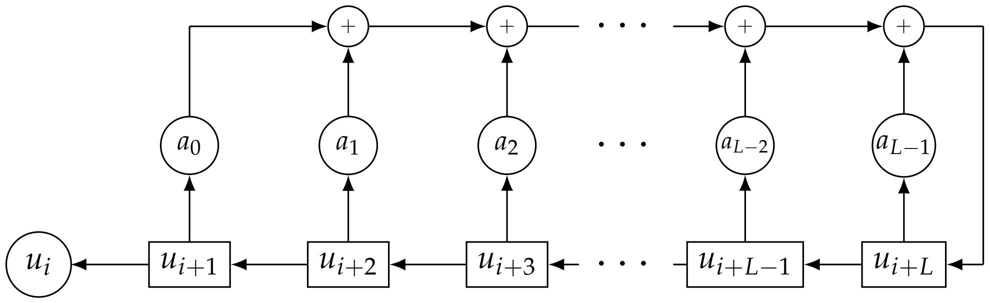

Figure 1. LFSR of length L (or LFSR with L stages).

The linear complexity of a sequence, denoted by LC, is defined as the length of the shortest LFSR that generates such a sequence, i.e., the degree of its characteristic polynomial. In cryptography, LC must be as large as possible. The expected value is approximately half the period $LC \simeq T/2$ (see [13]). Nowadays, values of T in the range $T \geq 2^{128}$, i.e., $LC \simeq 2^{64}$, seem to be enough for cryptographic purposes (see specifications of the candidates in the call of NIST for lightweight cryptography primitives [5]). Notice that all examples included in this work are merely illustrative, since they do not achieve the required values for cryptographic applications. PN-sequences produced by maximal-length LFSRs have a large period, but their LC is very low. This is due to the inherent linearity of these sequences; thus, we need to do something to break it. One possible approach is implementing irregular decimation on the PN-sequences.

2.1. Shrinking Generator

First, we need to recall the concept of decimation. The decimation of the sequence $\{s_i\}$ by (distance) δ is the new sequence $\{u_i\} = \{s_{\delta \cdot i}\}$, which is obtained by taking every δ-th term of such a sequence [14].

The binary sequence generator known as the *Shrinking Generator* (SG) [9] is made up of two maximal-length LFSRs, R_1 and R_2, with lengths L_1 and L_2, respectively, satisfying $\gcd(L_1, L_2) = 1$. Denote by $p_k \in \mathbb{F}_2[x]$, with degree L_k, the characteristic polynomial of R_k, and $T_k = 2^{L_k} - 1$, the period of the corresponding PN-sequence, for $k = 1, 2$. The PN-sequence $\{a_i\}$ generated by R_1 decimates the PN-sequence $\{b_i\}$ produced by the other register R_2. The decimation rule satisfies the following: given a_i and b_i, $i = 0, 1, 2, \ldots$, the output sequence $\{s_j\}$ is obtained as

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

The sequence $\{s_j\}$ is known as the *shrunken sequence* whose period is $T = (2^{L_2} - 1)2^{L_1 - 1}$. Its linear complexity [10] satisfies the inequality $L_2 2^{L_1 - 2} < LC \leq L_2 2^{L_1 - 1}$, and its characteristic polynomial has the form $p(x)^m$, where $2^{L_1-2} < m \leq 2^{L_1-1}$ and $p(x)$ is a primitive polynomial of degree L_2 [15]. Notice that here, $p(x)^m$ denotes the power of the polynomial $p(x)$ with coefficients modulo 2.

The shrunken sequence is almost balanced with $2^{L_1+L_2-2}$ ones in its first period. This binary generator is suitable for applications in stream ciphers, since it is easy to implement and has nice cryptographic properties. Notice that the shrunken sequence is obtained by the irregular decimation of a PN-sequence according to the ones of another PN-sequence.

Example 1. *Consider R_1 and R_2, LFSRs with characteristic polynomials $p_1(x) = 1 + x + x^2$ and $p_2(x) = 1 + x^2 + x^3$, and initial states $\{11\}$ and $\{111\}$, respectively. The shrunken sequence can be computed as*

$$\begin{array}{rccccccccccccccccccccc}
R_1: & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
R_2: & 1 & 1 & \cancel{1} & 0 & 1 & \cancel{0} & 0 & 1 & \cancel{1} & 1 & 0 & \cancel{1} & 0 & 0 & \cancel{1} & 1 & 1 & \cancel{0} & 1 & 0 & \cancel{0} \\
\{s_j\}: & \mathbf{1} & \mathbf{1} & & \mathbf{0} & \mathbf{1} & & \mathbf{0} & \mathbf{1} & & \mathbf{1} & \mathbf{0} & & \mathbf{0} & \mathbf{0} & & \mathbf{1} & \mathbf{1} & & \mathbf{1} & \mathbf{0} &
\end{array}$$

The generated sequence has period 14, and it is easy to check that its characteristic polynomial is $p(x)^2 = (1 + x + x^3)^2$, i.e., the linear complexity is $LC = 6$.

Let $\mathbb{F}_{2^{L_2}}$ denote the extension field of $\mathbb{F}_2$, where α root of $p_2(x)$, is a primitive element [16]. The next results state that the shrunken sequence can be obtained interleaving shifted versions of one single PN-sequence.

Theorem 1 (**[10], Theorem 3.1**)**.** *The sequences obtained decimating by 2^{L_1-1}, the shrunken sequence, are PN-sequences with period T_2. We call these sequences the interleaved PN-sequences of the shrunken sequence.*

Theorem 2 (**[10], Theorem 3.3**)**.** *The primitive polynomial $p(x)$ that generates the interleaved PN-sequences of the shrunken sequence can be computed as*

$$p(x) = (x + \alpha^{T_1})(x + \alpha^{2T_1})(x + \alpha^{4T_1}) \cdots (x + \alpha^{2^{L_2-1}T_1}),$$

where $\alpha \in \mathbb{F}_{2^{L_2}}$ is a root of $p_2(x)$.

Corollary 1 (**[10], Corollary 1**)**.** *If $L_2 = L_1 + 1$, then the polynomial $p(x)$ is the reciprocal polynomial of $p_2(x)$.*

In order to illustrate the previous results, we consider now another example with larger parameters.

Example 2. *Let R_1 and R_2 be two LFSRs with characteristic polynomials $p_1(x) = 1 + x^2 + x^3$ and $p_2(x) = 1 + x^3 + x^4$, with $L_1 = 3$ and $L_2 = 4$, and initial states $\{111\}$ and $\{1111\}$, respectively. The corresponding PN-sequences have periods $T_1 = 7$ and $T_2 = 15$, respectively. The shrunken sequence is given by*

$$\{s_j\} = \{111011010111011000111010000101011001101101001100001011111000\}.$$

It has period $T = (2^{L_2} - 1)2^{L_1-1} = 60$ and characteristic polynomial $p(x)^{16} = (1 + x + x^4)^4$, i.e., the linear complexity is $LC = 16$. If we decimate the shrunken sequence by $\delta = 4$, then we obtain the following four PN-sequences:

$$\begin{array}{rl}
\{s_{4\cdot j}\}: & \{110001001101011\} \\
\{s_{4\cdot j+1}\}: & \{111100010011010\} \\
\{s_{4\cdot j+2}\}: & \{101111000100110\} \\
\{s_{4\cdot j+3}\}: & \{011010111100010\}
\end{array}$$

The characteristic polynomial of these four interleaving PN-sequences is

$$p(x) = \left(x + \alpha^7\right)\left(x + \alpha^{14}\right)\left(x + \alpha^{28}\right)\left(x + \alpha^{56}\right) = 1 + x + x^4$$

where $\alpha \in \mathbb{F}_{2^{L_2}}$ is a root of $p_2(x)$ and $p(x)$ is the reciprocal polynomial of $p_2(x)$. Notice that the four PN-sequences are shifted versions of the same PN-sequence.

The polynomial $p(x)$ depends on L_1 (the degree of $p_1(x)$) and $p_2(x)$. Thus, every primitive polynomial with degree L_1 produces the same polynomial $p(x)$, once the polynomial $p_2(x)$ is fixed.

Notice that if $p(x)$ generates the interleaved PN-sequences of the shrunken sequence, then $p(x)^{2^{L_1-1}}$ generates such a sequence. Nonetheless, although $p(x)^{2^{L_1-1}}$ always generates the shrunken sequence, it might not be the characteristic polynomial. Sometimes, the characteristic polynomial has the form $p(x)^m$, with $2^{L_1-2} < m < 2^{L_1-1}$.

2.2. Shifted Versions of the Same PN-Sequence

In Section 2.1, we saw that the shrunken sequence can be generated interleaving shifted versions of the same PN-sequence, and the characteristic polynomial of these PN-sequences is obtained from the input polynomials of the shrinking generator. The shifts of the shifted versions can be also obtained via the input LFSRs (see [10,11]), and this fact is used to attack the SG [11]. One way to deal with this liability is to consider random shifts.

In this section, we briefly comment on the results obtained in [12]. First, we need to introduce the concept of *t-interleaved sequence*. We say that the sequence $\{s_j\}$ is obtained interleaving the sequences $\{u_i^{(1)}\}$, $\{u_i^{(2)}\}$, ..., $\{u_i^{(t)}\}$, all of them with period T, if it has the following form

$$\{s_j\} = \left\{u_0^{(1)}, u_0^{(2)}, \ldots, u_0^{(t)}, u_1^{(1)}, u_1^{(2)}, \ldots, u_1^{(t)}, \ldots, u_{T-1}^{(1)}, u_{T-1}^{(2)}, \ldots, u_{T-1}^{(t)}\right\}.$$

We call this sequence a *t-interleaved sequence*.

In [12], the authors consider that these t sequences $\{u_i^{(j)}\}$ for $j = 1, 2, \ldots, t$, are PN-sequences obtained from the same primitive polynomial, that is, shifted versions of the same PN-sequence. If the corresponding LFSR has length L, then the resultant t-interleaved sequence is almost balanced, and its number of 1s is $t \cdot 2^{(L-1)}$.

The linear complexity for this sequence satisfies $LC \leq t \cdot L$ and its period $T \leq t \cdot (2^L - 1)$. For a fixed value of t, almost 90% of the t-interleaved sequences (running over all possible shifted versions) achieve the maximal LC and period. In [12], the authors study more deeply the cases where $t = 2^l$, and they perform a preliminary analysis on the randomness of these sequences. They also provide some tools to identify the cases where the LC is low and the sequences are not suitable for cryptographic purposes. More information about these sequences and some comparison with the sequences constructed in this work can be found in Section 4.

In this work, we consider t-interleaved sequences obtained interleaving PN-sequences from different primitive polynomials with the same degree. Note that these t-interleaved sequences can be seen as the output sequences of a keystream generator where, at each clock pulse, we obtain at the same time the output of t different LFSRs. That is, at each instant t_i, the output bits are $\{u_{t_i}^{(1)}, u_{t_i}^{(2)}, \ldots, u_{t_i}^{(t)}\}$. Therefore, the interleaving method, in this case, could be considered as the concatenation of the output of t LFSRs at each instant of time. On the other hand, this interleaving method is very similar to the generation method of a DLFSR. A DLFSR (Dynamic Linear Feedback Shift Register) is a type of LFSR in which the characteristic polynomial changes at certain clock pulse [17,18]. In Figure 2, we represent a DLFSR that consists of a main LFSR and an additional control module. This module manages the characteristic polynomial used at each instant of time. The sequences generated by a DLFSR can be considered as the concatenation of segments of different PN-sequences. The purpose of a DLFSR is to generate sequences with larger periods and higher linear complexity than the ones produced by a single LFSR [19,20]. To carry out this task, the control module modifies different feedback parameters to generate a different sequence. Our interleaving method can be seen as a DLFSR where the characteristic polynomial changes depending on the counter module, i.e., at each clock pulse, we consider a different primitive polynomial. In Figure 3, we can check the generation of a four-interleaved sequence. At each clock pulse, one bit is generated from the corresponding LFSR in that instant, and then, we jump from the actual polynomial to the next one. Thus, we obtain our interleaved sequence concatenating the individual outputs of each one of the LFSRs at each instant of time.

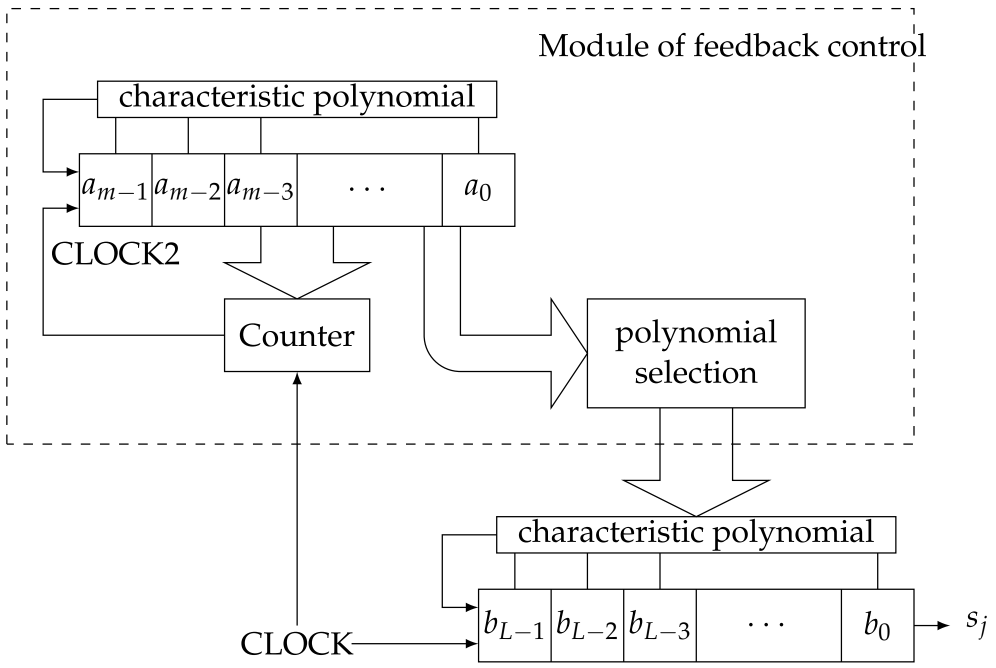
Module of feedback control
characteristic polynomial
a_{m-1} a_{m-2} a_{m-3} ··· a_0
CLOCK2
Counter
polynomial selection
characteristic polynomial
b_{L-1} b_{L-2} b_{L-3} ··· b_0
s_j
CLOCK

Figure 2. DLFSR.

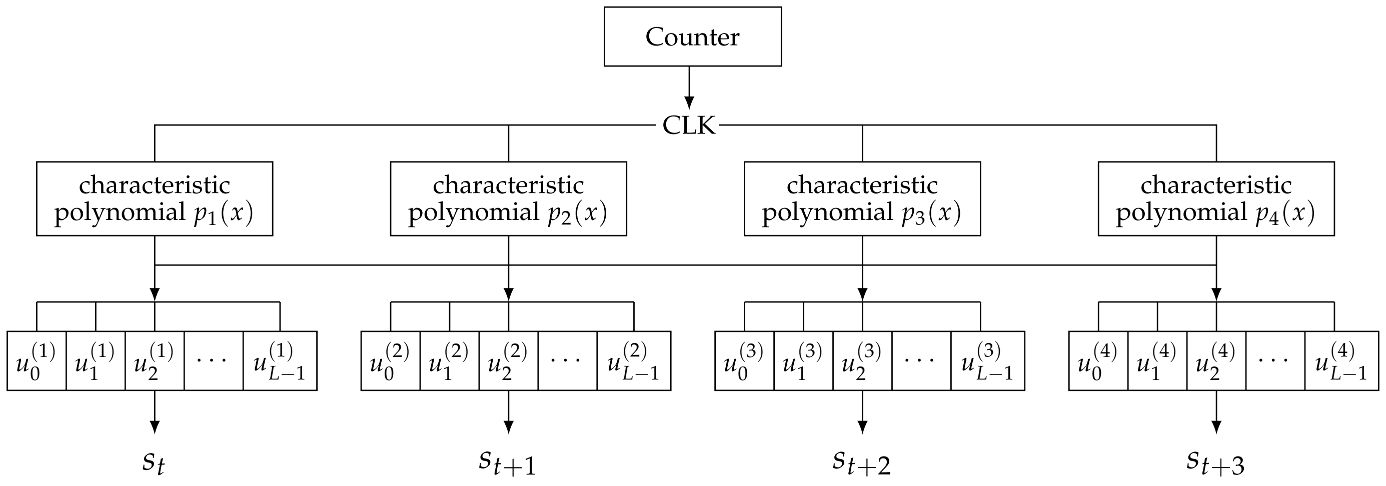
Counter
CLK
characteristic polynomial $p_1(x)$
characteristic polynomial $p_2(x)$
characteristic polynomial $p_3(x)$
characteristic polynomial $p_4(x)$
$u_0^{(1)}$ $u_1^{(1)}$ $u_2^{(1)}$ ··· $u_{L-1}^{(1)}$
$u_0^{(2)}$ $u_1^{(2)}$ $u_2^{(2)}$ ··· $u_{L-1}^{(2)}$
$u_0^{(3)}$ $u_1^{(3)}$ $u_2^{(3)}$ ··· $u_{L-1}^{(3)}$
$u_0^{(4)}$ $u_1^{(4)}$ $u_2^{(4)}$ ··· $u_{L-1}^{(4)}$
s_t
s_{t+1}
s_{t+2}
s_{t+3}

Figure 3. The generation of a four-interleaving sequence as a DFLSR.

3. Interleaving PN-Sequences with Different Characteristic Polynomials

In this section, we analyze the interleaving of PN-sequences obtained from different polynomials with the same degree.

Consider t maximal-length LFSRs, notated R_1, R_2,. . ., R_t, with primitive characteristic polynomials $p_1(x), p_2(x), \dots, p_t(x)$, respectively, and all of them with degree L. Given the

PN-sequence $\{a_i^{(k)}\}$, generated by R_k, for $k = 1, 2, \ldots, t$, the corresponding t-interleaved sequence $\{s_j\}$ is obtained as follows

$$\{s_j\} = \{a_0^{(1)}, a_0^{(2)}, \ldots, a_0^{(t)}, a_1^{(1)}, a_1^{(2)}, \ldots, a_1^{(t)}, \ldots, a_{L-1}^{(1)}, a_{L-1}^{(2)}, \ldots, a_{L-1}^{(t)}, \ldots\}.$$

From now on, we only consider t-interleaved sequences obtained with different polynomials of the same degree.

The following result provides the value of the LC for the t-interleaved sequences. Moreover, it allows us to obtain their characteristic polynomials.

Theorem 3 (**[21], Theorem 1**)**.** *The linear complexity of the sequence generated interleaving t PN-sequences produced by different primitive polynomials $p_1(x), \ldots, p_t(x)$ of degree L is $LC = t^2 L$. Furthermore, the characteristic polynomial is*

$$p(x) = \prod_{i=1}^{t} p_i(x^t).$$

It is worth noticing that the LC and period are not affected by the initial states.

Example 3. *Consider 3 registers with primitive polynomials $p_1(x) = 1 + x^2 + x^5$, $p_2(x) = 1 + x + x^2 + x^4 + x^5$ and $p_3(x) = 1 + x + x^2 + x^3 + x^5$. We take the initial states $\{11101\}$, $\{10001\}$ and $\{10101\}$, respectively. The corresponding PN-sequences are*

$$\begin{aligned} \{a_i^{(1)}\} &: \{1110101000010010110011111000110\} \\ \{a_i^{(2)}\} &: \{1000100101011000011100110111110\} \\ \{a_i^{(3)}\} &: \{1010100011101111100100110000101\}. \end{aligned}$$

If we interleave these three PN-sequences, we obtain a sequence with period $T = 93$ and $LC = 45$:

$$\begin{gathered} \{111100101000111000100010001011001110011001101 \\ 001101110010011100100111111100010010010111110001\} \end{gathered}$$

Using the Berlekamp–Massey algorithm [22], it is possible to check that the characteristic polynomial of this sequence is

$$\begin{aligned} p(x) &= 1 + x^9 + x^{24} + x^{27} + x^{39} + x^{42} + x^{45} \\ &= p_1(x^3) \cdot p_2(x^3) \cdot p_3(x^3) \end{aligned}$$

with

$$\begin{aligned} p_1(x^3) &= 1 + x^6 + x^{15} = (x^5 + x^4 + x^3 + x + 1)(x^{10} + x^9 + x^7 + x^5 + x^2 + x + 1) \\ p_2(x^3) &= 1 + x^3 + x^6 + x^{12} + x^{15} = (x^5 + x^4 + x^3 + x^2 + 1)(x^{10} + x^9 + x^7 + x^2 + 1) \\ p_3(x^3) &= 1 + x^3 + x^6 + x^9 + x^{15} = (x^5 + x^3 + 1)(x^{10} + x^8 + x^6 + x^5 + 1) \end{aligned}$$

where all three polynomials of degree 5 are primitive and those of degree 10 are irreducible.

The next result is a particular case of Theorem 3 for the case in which t is a power of 2.

Corollary 2. *Let t be a power of two. Then, the characteristic polynomial of a t-interleaved sequence produced by t different primitive polynomials $p_1(x), \ldots, p_t(x)$ of degree L is*

$$p(x) = [p_1(x) \cdot p_2(x) \cdots p_{t-1}(x) \cdot p_t(x)]^t.$$

Proof. Let $t = 2^r$ for r be a positive integer. The result is an immediate consequence of the fact that $p_i(x^{2^r}) = p_i(x)^{2^r}$ in $\mathbb{F}_2$. $\square$

Next, we show different examples of the generation of t-interleaved sequences. We analyze their LC and their characteristic polynomials depending on the choice of the initial primitive polynomials.

In the following example, we obtain a $four$-interleaved sequence corresponding to two primitive polynomials and their corresponding reciprocal polynomials.

Example 4. *Consider $four$ registers with primitive polynomials $p_1(x) = 1 + x^2 + x^5$, $p_2(x) = 1 + x^3 + x^5$, $p_3(x) = 1 + x^2 + x^3 + x^4 + x^5$ and $p_4(x) = 1 + x + x^2 + x^3 + x^5$. We observe that $p_2(x)$ and $p_4(x)$ are the reciprocal polynomials of $p_1(x)$ and $p_3(x)$, respectively. We take the initial states $\{01101\}$, $\{10001\}$, $\{10001\}$, and $\{00110\}$, respectively. The corresponding PN-sequences are*

$$\begin{aligned}
\{a_i^{(1)}\} &: \{0110111010100001001011001111100\} \\
\{a_i^{(2)}\} &: \{1000111110011010010000101011101\} \\
\{a_i^{(3)}\} &: \{1000101011101000011001001111011\} \\
\{a_i^{(4)}\} &: \{0011000010110101000111011111001\}.
\end{aligned}$$

If we interleave these four PN-sequences, we obtain a sequence with period $T = 124$ and $LC = 80$:

$$\begin{aligned}
\{&0110100010010001111011001110010011110010100101110100000101001 0 \\
&01001001101000000110111001010000111111101111111111110000100111\}
\end{aligned}$$

Using the Berlekamp–Massey algorithm [22], it is easy to check that the characteristic polynomial of this sequence is

$$\begin{aligned}
p(x) &= [p_1(x) \cdot p_2(x) \cdot p_3(x) \cdot p_4(x)]^4 \\
&= (1 + x^2 + x^5)^4 (1 + x^3 + x^5)^4 (1 + x^2 + x^3 + x^4 + x^5)^4 (1 + x + x^2 + x^3 + x^5)^4 \\
&= 1 + x^4 + x^8 + x^{12} + x^{20} + x^{32} + x^{36} + x^{40} + x^{44} + x^{48} + x^{60} + x^{68} + x^{72} + x^{76} + x^{80}.
\end{aligned}$$

In this example, the LC does not depend on the initial states; its value is always 80. *Moreover, if we consider different primitive polynomials of degree* 5, *the value of LC remains the same.*

The next example shows a case where there are no reciprocal polynomials.

Example 5. *Consider now four registers with primitive polynomials $p_1(x) = 1 + x + x^7$, $p_2(x) = 1 + x^3 + x^7$, $p_3(x) = 1 + x + x^2 + x^3 + x^7$ and $p_4(x) = 1 + x^2 + x^3 + x^4 + x^7$, with initial states $\{1010001\}, \{0111011\}, \{1101001\}, and \{1100101\}$, respectively. If we interleave the four PN-sequences generated by the previous polynomials, we obtain a sequence with period 508 (the same as that of the SG with polynomials of degree 3 and 7) and $LC = 112$, which is four times higher than that of the SG:*

$$\begin{aligned}
\{&10110111110001100001010011111110110010100001010010000110010100111010011010 \\
&10100001110011100110111010001110010111101100110010111110101011110111000001 1 \\
&00000110100001001011111111010010101000001101101000101111110001100111000100 1 \\
&00111101111000011000011011011000011010011110101110010110110011100001001101 0 \\
&11100110001011000011001101000010110101001100100011100101011100010010010111 1 \\
&01011001111110010010000110101000101101100011001011101110011111011100010101 0 \\
&1100110000001001000000101010001010110110111010101001100100010\}.
\end{aligned}$$

The characteristic polynomial of the sequence is given by

$$\begin{aligned} p(x) &= [p_1(x) \cdot p_2(x) \cdot p_3(x) \cdot p_4(x)]^4 \\ &= [(1+x+x^7)(1+x^3+x^7)(1+x+x^2+x^3+x^7)(1+x^2+x^3+x^4+x^7)]^4 \\ &= (1+x^2+x^5+x^7+x^8+x^9+x^{11}+x^{12}+x^{15}+x^{16}+x^{17}+x^{18}+x^{24}+x^{25}+x^{28})^4. \end{aligned}$$

The next example shows that the polynomials must be all different to achieve the maximal complexity.

Example 6. *Consider the primitive polynomials* $p_1(x) = p_2(x) = 1 + x^2 + x^5$, $p_3(x) = 1 + x + x^2 + x^3 + x^5$ *and* $p_4(x) = 1 + x^2 + x^3 + x^4 + x^5$ *and consider the initial states* $\{10111\}$, $\{11010\}$, $\{00011\}$, *and* $\{10110\}$, *respectively. The corresponding* $four$*-interleaved sequence has the following form:*

$$\{1101010010011111101001111000001010100010011100110000110000 1101 \\ 0011000011101111010101010111001111100110111010010001100001 1110\}$$

This sequence has period $T = 124$ *and* $LC = 60$*, which is not the maximal vale (80) for this parameter. The characteristic polynomial is given by:*

$$p(x) = [p_1(x)p_3(x)p_4(x)]^4 = 1 + x^4 + x^8 + x^{16} + x^{20} + x^{24} + x^{36} + x^{56} + x^{60}$$

Notice that in this case, the primitive polynomial is not the product of all four polynomials; this is due to the fact that $p_1(x) = p_2(x)$.

In Table 1, we can check the values of the LC of t-interleaved sequences using PN-sequences from different polynomials of degree L. It is worth recalling that there are only six primitive polynomials of degree 5 and six of degree 6. It means that when we construct seven-interleaved sequences or eight-interleaved sequences, we have to consider at least one repeated polynomial. Therefore, the values in red in Table 1 are just upper bounds, since, as we saw in Example 6, when the polynomials are not different, we risk having a sequence without maximal LC.

Table 1. LC of the interleaving sequences of t primitive polynomials of degree L.

t \ L	5	6	7	8	9
4	80	96	112	128	144
5	125	150	175	200	225
6	180	216	252	288	324
7	245	294	343	392	441
8	320	384	448	512	576

4. Comparison with Other Sequence Generators

In this section, we analyze briefly the advantages of our t-interleaved sequences compared with the sequences obtained from generators with similar parameters.

1. *Shrinking generator*
   Given two primitive polynomials of degree L_1 and L_2, the linear complexity of the shrunken sequence satisfies: $2^{L_1-2} < LC \leq L_2 \cdot 2^{L_1-1}$ and $T = (2^{L_2} - 1)2^{L_1-1}$. In this case, the sequence is obtained interleaving 2^{L_1-1} shifted versions of the same PN-sequence.
   If we interleave 2^{L_1-1} PN-sequences generated by different primitive polynomials of degree L_2, the linear complexity of the resultant sequence is $LC = L_2 \cdot 2^{2(L_1-1)}$, which

is much higher than that of the SG. Notice that the period and the number of ones remain the same.
In the following example, we compare the shrunken sequence and the corresponding t-interleaved with similar parameters. We see that the LC of the t-interleaved sequence is greater.

Example 7. *Consider the SG composed of two registers of length* $L_1 = 3$ *and* $L_2 = 5$*. In this case, the shrunken sequence is made up by interleaving four shifted versions of the same PN-sequence generated by a primitive polynomial of degree 5. The period of the shrunken sequences in this case is* $T = 124$ *and* $LC \leq 20$*.*
If we consider again Example 4, we interleave four PN-sequences produced by primitive polynomials of degree 5. The resultant sequence has period 124 (the same as that of the SG with polynomials of degree 3 and 5) and $LC = 80$*, which is four times higher than that of the SG.*

2. *t-interleaved sequences with the same polynomial*
In [12], the authors analyze the t-interleaved sequences obtained interleaving shifted versions of the same PN-sequence produced by a primitive polynomial $p(x)$ of degree L. They determine the period and the linear complexity of the t-interleaved sequences for some particular cases of t. They also study an upper bound for the LC and the period of t-interleaved PN-sequences. In [21], the authors study different cases of interleaving sequences, analyzing the LC and the characteristic polynomials of the resultant sequences. The next theorem is a consequence of Theorem 2 in [21] and the results obtained in [12].

Theorem 4. *[12,21] Consider a primitive polynomial* $p(x)$ *of degree* L*. If we interleave* t*-shifted versions of the same PN-sequence of period* $T = 2^L - 1$*, then the resultant* t*-interleaved sequence has an* $LC \leq t \cdot L$ *and period* $T \leq t \cdot (2^L - 1)$*.*

In the following example, we compare a t-interleaved sequence obtained using one primitive polynomial $p(x)$ of degree L with a corresponding t-interleaved sequence, with similar parameters, obtained using t different primitive polynomials $p_i(x)$ of degree L, $i = 1, \ldots, t$. We see that the LC of the t-interleaved sequence with different polynomials is greater.

Example 8. *Consider any* $four$*-interleaved sequence obtained with a primitive polynomial of degree* 5 *and* $four$ *shifted versions of the corresponding PN-sequence. In this case, the period of the sequences is* $T \leq 124$ *and* $LC \leq 20$*.*
Consider the primitive polynomials of degree 5 given in Example 4 and interleave $four$ *different PN-sequences produced by these polynomials. The resultant sequences have period* 124 *and* $LC = 80$*. If we compare both types of* $four$*-interleaved sequence, we have that using different polynomials for the construction provides higher values for the LC, in this case, four times larger.*

In Table 2, we have a comparison between the values of LC and T for our t-interleaved sequences and the values for the sequences obtained in [12] (using shifted versions of the same PN-sequence, that is, with the same characteristic polynomial). First of all, notice that the values for the same polynomial are upper bounds (it depends on the initial state), while the values for our sequences are exact (regardless the initial state). Note that values of LC are higher in our sequences.

In order to complete this comparison, in the next section, we perform a statistical study on the randomness of our t-interleaved sequences, and we compare these results with the ones obtained for t-interleaved sequences obtained with shifted versions of the same PN-sequence (which includes the shrunken sequence), that is, using always the same characteristic polynomial.

Table 2. Values for the LC and the period T of t-interleaved sequences obtained from one single primitive polynomial of degree L compared with the values taking different polynomials.

	t Different Polynomials		**1 Polynomial**	
(t, L)	**LC**	**T**	**LC**	**T**
(8,16)	1024	524,280	128	524,280
(8,17)	1088	1,048,568	136	1,048,568
(8,18)	1152	2,097,144	144	2,097,144
(8,19)	1216	4,194,296	152	4,194,296
(8,20)	1280	8,388,600	160	8,388,600

5. Statistical Analysis of T-Interleaved Sequences

RNGs should be designed and selected based on a solid theoretical analysis of their mathematical structure. In our algorithm, we interleave PN-sequences to hide and delete their linearity. Once our generator is designed and implemented, the next step is to submit it to empirical statistical tests in order to detect statistical deficiencies. In the study of RNGs, different quality criteria can be used. However, three basic properties of a random bit sequence $\{s_i\}$ should be achieved:

1. *Unpredictability*: Having k consecutive elements of $\{s_i\}$ should not give any information about the next element $k+1$ of the sequence.
2. *Uniformity*: Given any subsequence of $\{s_i\}$, there should be nearly equal number of 1's and 0's.
3. *Independence*: Each element of $\{s_i\}$ is independent from other elements.

There is no mathematical proof that ensures the randomness of a bit sequence; however, there exists a huge number of empirical tests to determine if a sequence is random enough and secure to be used in cryptography [23]. If the sequences produced by a particular generator pass the statistical tests, then this could be accepted as a generator of random sequences. Otherwise, if any of the tests fail, then it means the generator is not good and must be rejected.

- *Golomb's Randomness Postulates*

Golomb's postulates constitute a base for randomness tests, since they were one of the first attempts to establish some necessary conditions for a periodic pseudo-random sequence to look random. Sequences satisfying the three properties are called PN-sequences. The sequences produced by LFSRs are PN-sequences in these terms. At present, these conditions are far from being sufficient for such sequences to be considered random. However, there are diverse ways and tools that allow us to analyze the randomness of the sequences.

>From now on, we consider $\{s_i\}$ a binary sequence of period T. A run of $\{s_i\}$ is defined as a maximal subsequence of consecutive bits of either all ones or all zeros. A run of zeroes is called a gap, and a run of ones is called a block. Golomb's postulates are defined as follows:

(R1) In a period of $\{s_i\}$, the number of ones should differ from the number of zeros by at most 1. In other words, the sequence should be balanced.

(R2) In a period of $\{s_i\}$, at least $\frac{1}{2}$ of the all runs of zeroes or ones should have length one, at least $\frac{1}{4}$ should have length 2, at least $\frac{1}{8}$ should have length 3, and so on. Moreover, for each one of these lengths, there should be (almost) equally many gaps and blocks.

(R3) The autocorrelation function $C(\tau)$ should be two valued. That is, for some integer k and for all $\tau = 0, 1, 2, \ldots, T-1$

$$C(\tau) = \frac{1}{T}\sum_{i=0}^{T-1}(-1)^{(s_i+s_{i+\tau})} = \begin{cases} 1 & \text{if } \tau = 0, \\ \frac{k}{T} & \text{if } 1 \le \tau \le T-1. \end{cases} \quad (2)$$

Any of Golomb's randomness postulates are analyzed through the statistical tests package FIPS 140-2 [24], as we study in Section 5.2.

In this section, we include diverse ways to analyze the randomness of our sequences. On the one hand, in Section 5.1, we present some visual results where, through different graphs, we could understand the behavior of the generated sequences. On the other hand, in Section 5.2, we evaluate various batteries of statistical tests, which help us to determine if our generator could be considered random. The generator and the battery of tests were implemented with Matlab R2020b in a Windows 10 environment in a 64 bits PC with CPU Intel Core i7, at 3 GHz. We check a great quantity of t-interleaved sequences, with $3 \leq t \leq 8$ and with polynomials of degree up to 27.

5.1. Simple Visual Analysis

In this subsection, we examine our random number generator creating a visualization of the sequences it produces. We study the autocorrelation, the return map, the chaos game, and the Lyapunov exponent. This type of approach should not be considered as an exhaustive or formal analysis. However, it is an interesting and easy way to get a rough impression of the performance of the generator.

- *Autocorrelation*

The autocorrelation function, defined in expression (2), measures the amount of similarity between the sequence $\{s_i\}$ and its shifted version by τ positions. If $\{s_i\}$ is a random periodic sequence of period T, then $|T \cdot C(\tau)|$ can be expected to be quite small for all values of τ with $0 < \tau < T$.

This function is a mathematical tool very useful for finding repeated patterns. It analyzes different sections of a message and compares them to find similarities. Moreover, it allows measuring the linear relationship between random variables of processes separated a certain distance. The first autocorrelation coefficient is always equal to 1, and the other coefficients must have the smallest amplitude possible, so that the sequence can be considered random.

In Figure 4, we compare the autocorrelation values for two 8-interleaved sequences. In Figure 4a, we show the results for an eight-interleaved sequence with eight different primitive polynomials of degree $L = 16$. We observe that the values are almost zero except for the first value which is 1, as would be expected for a random sequence. Obtaining these results provides an indication about the randomness of the sequence but not the certainty. That is, this does not guarantee that it was indeed produced by a random bit generator, but it means that we can continue checking it. However, in Figure 4b, we represent the results for a eight-interleaved sequence with the same polynomial of degree $L = 16$. We can observe in the graph how the values increase for some shifts of the sequence with itself. This allows us to deduce the existence of certain autocorrelation in this sequence.

- *Chaos Game*

Chaos game [25–27] is a method that converts a one-dimensional sequence into a sequence in two dimensions providing a very provocative visual representation, which reveals some of the statistical properties of the sequence under study. >From this graphical tool, we can visually look for patterns in the sequences generated by a random number generator.

Figure 5 shows the Chaos maps of two eight-interleaved sequences with polynomials of degree 16. In Figure 5b, we have the Chaos map of an eight-interleaved sequence generated with one single polynomial. We can observe the lack of randomness in this sequence, since it presents a clear pattern. However, in Figure 5a, we have the Chaos map of an eight-interleaved sequence using different polynomials where we observe a disordered cloud, without patterns. It means that there is an indication of a Chaos map but not certainty. That is, it does not assure the randomness of our sequence, but we can continue with the analysis of this generator.

A practical method of determining whether a system is chaotic or not is the calculation of the Lyapunov exponent, which we study in the next section.

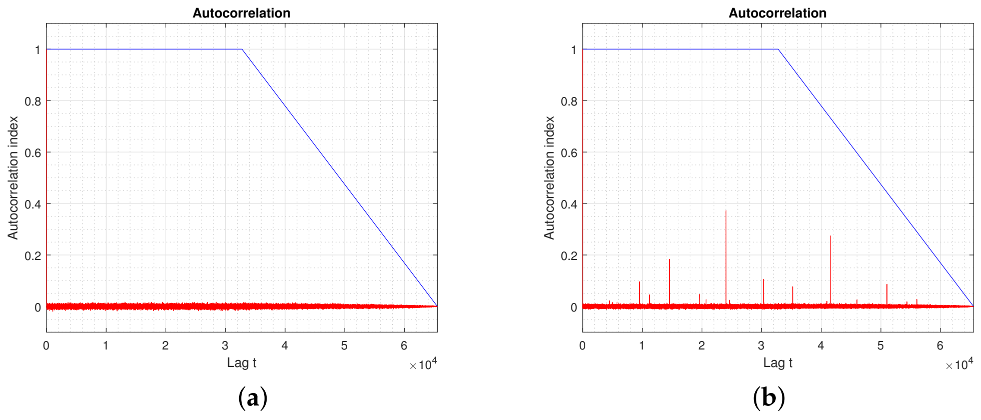

Figure 4. Autocorrelation for eight-interleaved sequences with polynomials of degree 16. (**a**) Eight-interleaved sequence with different polynomials; (**b**) Eight-interleaved sequence with the same polynomial.

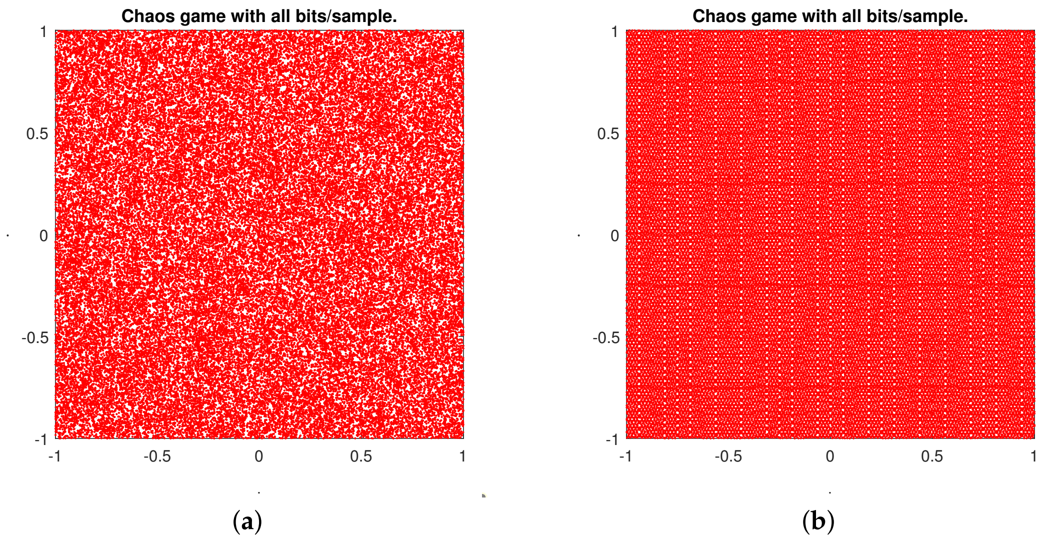

Figure 5. Chaos map for eight-interleaved sequences with polynomials of degree 16. (**a**) Eight-interleaved sequence with different polynomials; (**b**) Eight-interleaved sequence with the same polynomial.

- *Lyapunov exponent*

Lyapunov exponent is an essential tool and a useful analytical metric to characterize the chaos. An important property of chaos is its very sensitive dependence on initial condition. Lyapunov exponent is used as a quantitative measure for this dependence.

Lyapunov exponent of a dynamical system is a quantity that characterizes the rate of separation of infinitesimally close trajectories $Z(t)$ and $Z_0(t)$ in phase space

$$|Z(t) - Z_0(t)| \approx |Z(0) - Z_0(0)|e^{\lambda t}.$$

The exponent λ measured for a long period of time (ideally $t \to \infty$) is the Lyapunov exponent.

Next, we consider the definition of Lyapunov exponent given for sequences in [28]. Let d_0 be the measure of the initial distance between two sequences and d_t be the distance between the same sequences but after t iterations. We define *Lyapunov exponent (LE)* as:

$$LE = \frac{1}{t} \ln\left(\left|\frac{d_t}{d_0}\right|\right).$$

It is desirable that two very close initial conditions provide very different trajectories (sequences). If LE is greater than zero, the distance between two close initial conditions rapidly increases in the time, which means there exists an exponential divergence of the trajectories of a chaotic system. This value gives an idea of how different the sequences are generated by similar seeds, which is a very important feature to avoid attacks on the key of the generator. However, if $LE = 0$, the sequences decrease their distance, and they tend to join and be confused in one. The system converges, and it is not at all random.

We can use the Hamming distance (which indicates the number of bit positions in which both sequences differ) instead of the logarithm of the Euclidean distance in the Lyapunov exponent, and it is called the Lyapunov Hamming Exponent (LHE). If two numbers are identical, then its LHE value will be 0. Nevertheless, if all the bits of both numbers are different, then its LHE will be $LHE = \log_2\ m = \log_2 2^n = n$, where n is the number of bits with which the numbers are encoded.

Obtaining the Lyapunov Hamming exponent for the chosen sequence is done by calculating the average of the LHE between every two consecutive numbers of the sequence. The best value will be $n/2$.

For this case, we take $n = 8$, so the best value is 4. Next, we show the value obtained for a eight-interleaved sequence with polynomials of degree $L = 20$

Lyapunov Hamming exponent, ideal	=	4
Lyapunov Hamming exponent, real	=	4.0001
Absolute desviation from ideal	=	2.2889×10^{-5}

Hence, the proposed generator passes this test.

All the t-interleaved sequences with different polynomials analyzed have passed this test.

- *Return map*

In Information Theory, the entropy of a sequence is a measure of the amount of information of a process in bits; or it is a measure of the diversity of the elements in the sequence. It is computed from the frequencies of each element of the alphabet in the sequence.

The return map is useful to visually measure the entropy of the sequence above defined; that is, it allows us to detect the existence of some useful information about the parameters used in the design of pseudo-random generators [29].

The return application consists of drawing a two-dimensional graph of the points of the sequence $\{s_i\}$ as a function of $\{s_{i-1}\}$. The result should be a distribution of points where you can guess no trend, no shape, no line, no symmetry, and no pattern, as happens in the Chaos map.

In Figure 6, we represent the return maps of two eight-interleaved sequences with polynomials of degree 16. In Figure 6a, we have the return map of an eight-interleaved sequence using different polynomials. We observe a disordered cloud, without patterns, which, in principle, does not provide any useful information for the cryptanalysis of the sequence. It does not mean that our sequence is random, simply that it is not rejected in the randomness analysis. However, in Figure 6b, we represent the return application of an eight-interleaved sequence generated with one single polynomial. We can check that this graph presents a pattern of defined curves, which are repeated. It indicates non-randomness in the sequence.

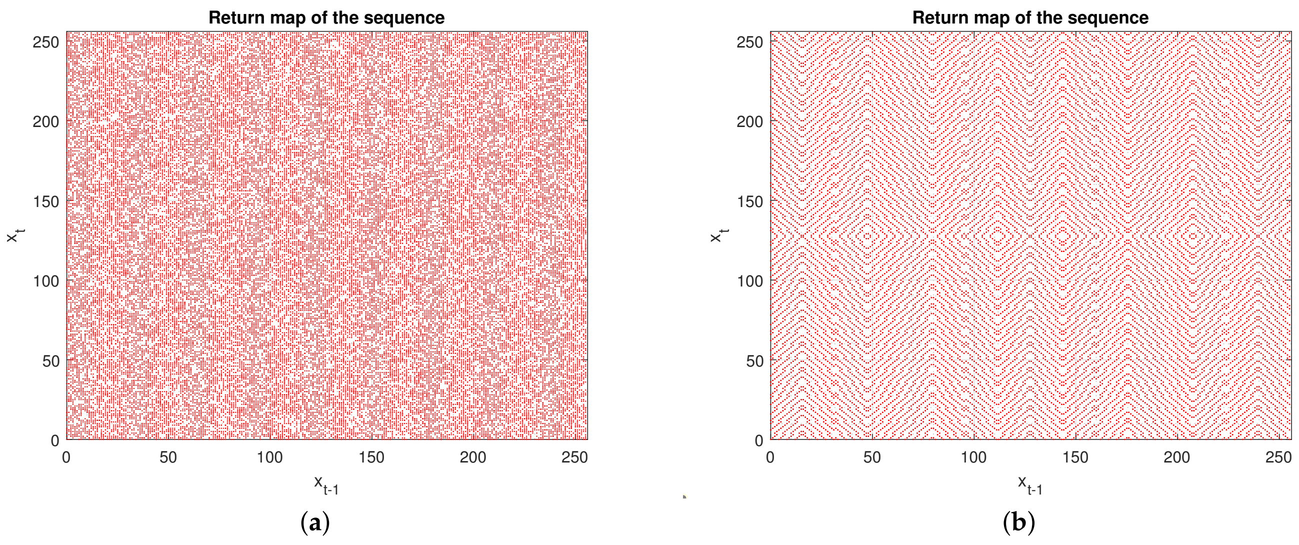

Figure 6. Return map for eight-interleaved sequences with polynomials of degree 16. (**a**) Eight-interleaved sequence with different polynomials; (**b**) Eight-interleaved sequence with the same polynomial.

5.2. Battery of Statistical Tests

Next, we present two of the most important batteries of statistical tests used to evaluate the randomness of the sequences generated by pseudo-random number generators, Diehard and NIST.

- NIST
  The Statistical Test Suite developed by NIST [30] is an excellent and exhaustive document looking at various aspects of randomness in a long sequence of bits. The NIST has documented 15 statistical tests, where FIPS 140-2 package, Maurer's Universal Test [31] and Lempel–Ziv Compression Test are among them.
  The Frequency Test, Maurer's Universal Test and Lempel-Ziv Compression have the standard normal as reference distribution. The rest of the tests have the chi-square χ^2 as reference distribution. The chi-square and the normal variation is converted into a p-value. If the computed p-value is <0.01, then we conclude that the sequence is non-random. Otherwise, we conclude that the sequence is random.
- *FIPS 140-2*
  FIPS 140-2 is an U.S. government computer security standard used to approve cryptographic modules issued by the National Institute of Standards and Technology (NIST). It consists of four statistical random number generator tests: the Monobit Test, The Poker Test, The Runs Test and The Long Runs Test. Moreover, we add the Frequency Test within a block, which can be categorized as a frequency test. If a sequence passes all five tests, there is no guarantee that it was indeed produced by a random bit generator. However, if one of the algorithms fails any of these tests, then the other tests are not even applied, and we can not consider our sequence sufficiently random; and, therefore, our generator is not secure in cryptographic terms.
  Each test needs a binary sequence of 10^6 bits. All our sequences have the required length for this analysis.
  1. FREQUENCY (MONOBIT) TEST: The focus of the test is the proportion of zeros and ones along the whole sequence. The purpose of this test is to determine whether the number of ones and zeros in a sequence are approximately the same as would be expected for a truly random sequence. The test assesses the closeness of the fraction of ones to $\frac{1}{2}$; that is, the number of ones and zeros in a sequence should be about the same. All subsequent tests depend on the approval of this test.

2. POKER (SERIAL) TEST: Let m be an integer number such that $\lfloor \frac{T}{m} \rfloor \geq 5 \cdot 2^m$ and let $k = \lfloor \frac{T}{m} \rfloor$. The sequence $\{s_i\}$ is divided into k non-overlapping parts each one of length m, and let m_i be the number of occurrences of the i-th type of sequence of length m, for $1 \leq i \leq 2^m$. The Poker Test determines if each stream of length m appears approximately the same number of times in $\{s_i\}$, as would be expected for a random sequence. Note that for $m = 1$, the Poker Test is equivalent to the Frequency Test.
3. RUNS TEST: The incidences of runs (for both consecutive zeros and consecutive ones) of all lengths ($\geq$1) in the sample stream should be counted and stored. The purpose of the Runs Test is to determine if the number of runs of different lengths in the sequence $\{s_i\}$ is as expected for a random sequence. In particular, this test determines whether the oscillation between zeros and ones is too fast or too slow.
4. LONG RUNS TEST: A long run is defined to be a run of length 26 or more (of either zeros or ones). The focus of this test is the longest run of ones within M-bit blocks. Its purpose is to determine whether the length of the longest run of ones within the sequence is consistent with the length of the longest run of ones that would be expected in a random sequence. Note that one irregularity in the expected length of the longest run of ones implies that there is also an irregularity in the expected length of the longest run of zeros. Therefore, only a test for ones is necessary.
5. FREQUENCY TEST WITHIN A BLOCK: The focus of this test is the proportion of ones within M-bit blocks. The purpose is to determine whether the frequency of ones in an M-bit block is approximately $\frac{M}{2}$, as would be expected under an assumption of randomness. For the block size $M = 1$, this test degenerates to the Frequency (Monobit) test.

Frequency Test is defined to check the first postulate of Golomb. The second postulate of Golomb, about the number of runs in sequences, is analyzed in the Runs Tests. Finally, the third postulate gives information about similarities between the sequence and shifted versions of it. If $\{s_i\}$ is a random sequence, the autocorrelation should be constant.

- *Maurer's Universal Test*
  The focus of this test is the number of bits between matching patterns (a measure that is related to the length of a compressed sequence). The purpose of the test is to detect whether or not the sequence can be significantly compressed without loss of information. A significantly compressible sequence is considered to be non-random.
- *Lempel–Ziv Compression Test*
  The focus of this test is the number of cumulatively distinct patterns (words) in the sequence. The purpose is to determine how far the tested sequence can be compressed; it is considered to be non-random if it can be significantly compressed. A random sequence will have a characteristic number of distinct patterns.
  This test works by reading a sequence of symbols, grouping the symbols into strings, and converting the strings into codes. We get compression because the codes take up less space than the strings they replace. No data are lost when compressing.
  In Table 3, we present a small sample of the results obtained in the NIST tests here presented. All these values are the average of the results obtained for any sample of t-interleaved sequences studied.

Table 3. p-values of some statistical tests of NIST for t-interleaved sequences with different characteristic polynomials of degree 20.

Tests \ t-Interleaved	3	4	5	6	7	8
Monobit	0.4952	0.4740	0.0493	0.6213	0.5592	0.9745
Poker	0.9083	0.2742	0.5086	0.0488	0.9960	0.4902
Runs	0.5402	0.5160	0.4629	0.5946	0.6000	0.9840
Long Runs	0.5715	0.5823	0.6785	0.8791	0.1204	0.9719
Frequency block	0.4068	0.2167	0.3672	0.7633	0.3473	0.5089
Maurer's	0.6623	0.4129	0.2069	0.8374	0.5539	0.4262
Lempel-Ziv	0.0931	0.9159	0.0531	0.2314	0.9069	0.9719

- *Diehard*

  Diehard battery of tests [32] is a reliable standard for evaluating the randomness of sequences of pseudo-random number generators. This tool is a powerful instrument for the practical evaluation process of cryptographic primitives. It cannot guarantee if your generator can be considered perfectly random, but if it does not pass the test suite, then it is not suitable for cryptographic applications.
  Diehard battery [32] consists of 15 different independent statistical tests, some of them repeated but with different parameters:

  1. BIRTHDAY SPACINGS TEST: Choose random points on a large interval. The spacings between the points should be asymptotically exponentially distributed.
  2. OPERM5 TEST: Analyze sequences of five consecutive random numbers. The 120 possible orderings should occur with statistically equal probability.
  3. BINARY RANK TEST FOR 31×31 MATRICES: The leftmost 31 bits of 31 random integers from the test sequence are used to form a 31×31 binary matrix over the field $\{0,1\}$. The rank is determined. That rank can be from 0 to 31, but ranks less than 28 are rare, and their counts are pooled with those for rank 28. Ranks are found for 40,000 of such random matrices, and a chi-square test is performed on counts for ranks $31, 30, 29$ and ≤ 28.
  4. BINARY RANK TEST FOR 32×32 MATRICES: The rank of a random 32×32 matrix is identified. Ranks less than 29 are rare. Chi-square tests are performed on the ranks $32, 31, 30$ and less than or equal to 29. This is repeated 40,000 times.
  5. BINARY RANK TEST FOR 6×8 MATRICES: The rank of a random 6×8 matrix is identified. Ranks less than 4 are rare. Chi-square tests are performed on the ranks $6, 5$ and less than or equal to 4. This is repeated 100,000 times.
  6. BITSTREAM TEST: Consider each bit as a single letter $(0$ or $1)$. In a rolling group of 20 bits, count the number of 20-bit permutations out of 2^{21} 20-bit groups. As there are 2^{20} possible 20-bit permutations, count how many are missing, which should be normally distributed. This test is repeated 20 times.
  7. OPSO, OQSO and DNA TESTS:
     (a) OPSO TEST: This is the overlapping-pairs-sparse-occupancy test. Each set of 5 bits is considered a 'letter'; thus, there are 1024 letters in the 'alphabet'. Two-letter words are taken from each 32-bit integer and are counted. As there are 2^{21} possible two-letter words, the missing words are identified and should be normally distributed.
     (b) OQSO TEST: A variant that uses four-letter words.
     (c) DNA TEST: A variant where there are only four letters in the alphabet, and each letter is two bits.
  8. COUNT-THE-1s TEST: A specific byte from each integer is chosen to represent a letter. There are five possible letters, each chosen by counting the number of

1's in the byte: $0, 1, 2 = A; 3 = B; 4 = C; 5 = D; 6, 7, 8 = E$. The five probabilities are therefore $37, 56, 70, 56$ and 37 over 256, respectively. Five integer sequences are selected on a rolling basis, and counts are made on word frequencies. A covariance matrix is formed.

9. PARKING LOT TEST: Randomly place unit circles in a 100×100 square. A circle is successfully parked if it does not overlap an existing successfully parked one. After 12,000 tries, the number of successfully parked circles should follow a certain normal distribution.
10. MINIMUM DISTANCE TEST: In a square of size 10,000 $\times$ 10,000, randomly select 8000 points. Find the minimum distance between the pairs. The square of this distance should be exponentially distributed with a mean close to 0.995. This is repeated for 100 random selections of 8000 points.
11. RANDOM SPHERES TEST: Randomly choose 4000 points in a cube of edge 1000. Center a sphere on each point, whose radius is the minimum distance to another point. The smallest sphere's volume should be exponentially distributed with a certain mean.
12. SQUEEZE TEST: Multiply 2^{31} by random floats on $(0, 1)$ until you reach 1. Repeat this 100,000 times. The number of floats needed to reach 1 should follow a chi-square distribution.
13. OVERLAPPING SUMS TEST: Generate a long sequence of random floats on $(0, 1)$. Add sequences of 100 consecutive floats. The sums should be normally distributed with characteristic mean and variance.
14. RUNS TEST: Generate a long sequence of random floats on a $[0, 1)$ distribution. Ascending and descending runs should follow a certain covariance matrix. This is repeated 10 times for sequences of length 10,000.
15. CRAPS TEST: Play 200,000 games of craps, counting the wins and the number of throws per game. Each count should follow a chi-square distribution.

Note that The Count-The-1s and the OPSO tests are both sometimes known as the Monkey Test. These statistical tests are designed to test the null hypothesis H_0, which states that the input sequence is randomly generated. If the hypothesis is not rejected in all the tests, then it is implied that the input sequences are random. Most of the tests in DIEHARD return a p-value or the KS p-value (given by the Kolmogorov–Smirnov test), which should be uniform on $[0, 1)$ if the input file contains truly independent random bits. It is considered that a bit stream really fails when it obtains p-values of 0 or 1 to six or more places.
Testing Diehard battery of tests for a hundred eight-interleaved sequences with different polynomials of degree 24, we say that Diehard does not show any weakness. >From the results of Table 4 of a particular sequence, we can check that all the values are in the appropriate range.

Table 4. Diehard battery of tests results for an eight-interleaved sequence with different characteristic polynomials of degree 24.

Test Name	p-Value	Result	Test Name	p-Value	Result
Birthday spacing	0.770936	Pass	OQSO	0.8197	Pass
	0.747460			0.1329	
	0.989202			0.5293	
	0.774785			0.6284	
	0.576802			0.7687	
	0.176450			0.0969	
	0.874796			0.7288	

Table 4. *Cont.*

Test Name	p-Value	Result	Test Name	p-Value	Result
	0.139735			0.9149	
	0.514557			0.9812	
Overlapping	0.974948	Pass	OQSO	0.7603	Pass
permutations	0.759794			0.6207	
Binary ranks 31 × 31	0.752307	Pass		0.8554	
Binary ranks 32 × 32	0.934338	Pass		0.3293	
Binary ranks 6 × 8	0.445734	Pass		0.0179	
Bit stream (Monkey tests)	0.76389	Pass		0.7859	
	0.13337			0.4336	
	0.67455			0.1403	
	0.49876			0.7540	
	0.88496			0.3442	
	0.96748			0.1236	
	0.07041			0.1888	
	0.08609			0.8394	
	0.67958			0.6233	
	0.61726			0.1351	
	0.78081			0.4005	
	0.61369			0.4097	
	0.80996			0.4941	
	0.88405			0.8206	
	0.35224		DNA	0.5756	Pass
	0.62968			0.7611	
	0.53228			0.5149	
	0.17966			0.8418	
	0.02605			0.9799	
	0.16593			0.2000	
OPSO	0.8834	Pass		0.6843	
	0.7423			0.8916	
	0.2625			0.2560	
	0.5394			0.2569	
	0.5394			0.0096	
	0.6175			0.2598	
	0.2614			0.1103	
	0.6739			0.2117	
	0.7986			0.5963	
	0.6588			0.3547	

Table 4. *Cont.*

Test Name	*p*-Value	Result	Test Name	*p*-Value	Result
	0.7102			0.4503	
	0.4069			0.5184	
	0.8906			0.9202	
OPSO	0.4968	Pass	DNA	0.0457	Pass
	0.1266			0.8440	
	0.1259			0.9479	
	0.8229			0.6468	
	0.4243			0.3536	
	0.3429			0.6446	
	0.6911			0.0831	
	0.1838			0.7538	
	0.2961			0.7575	
	0.2145			0.9951	
Count-the-1′s (stream of bytes)	0.476036	Pass		0.5849	
	0.572657			0.2852	
Count-the-1′s (specific bytes)	0.	Pass	Parking lot	0.407931	Pass
	0.453489		Minimum distance	0.752286	Pass
	0.531694		3D Spheres	0.947691	Pass
	0.476337		Squeeze	0.990622	Pass
	0.115181		Overlapping sums	0.276467	Pass
	0.238283		Runs	0.276783	Pass
	0.248038			0.893007	
	0.170200			0.908305	
	0.595302			0.913183	
	0.167417		Craps	0.995956	Pass
	0.574701			105,661	
	0.384873				
	0.944743				
	0.955924				
	0.210026				
	0.142320				
	0.717744				
	0.191102				
	0.728247				
	0.297792				
	0.971290				
	0.323464				
	0.408101				
	0.013264				
	0.859849				

6. Conclusions

Interleaving sequences is a way to increase the linear complexity of such sequences and to break the linearity just in case of working with PN-sequences. In this paper, we analyze the randomness of the sequences obtained by interleaving PN-sequences generated by different characteristic polynomials with the same degree. According to the obtained results, these sequences achieve the maximal possible linear complexity and, in terms of randomness, they are better than the sequences obtained interleaving PN-sequences with the same polynomial. Therefore, they seem to be suitable for applications in cryptography. As future work, we would like to apply more batteries of tests to our sequences and study what happens if we interleave PN-sequences with different periods. In this last case, we are not sure how the different periods can affect the resultant sequence. We need to perform a deep study in order to achieve some conclusions.

Author Contributions: All authors contributed equally. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Spanish State Research Agency (AEI) of the Ministry of Science and Innovation (MICINN), project P2QProMeTe (PID2020-112586RB-I00/AEI/10.13039/501100011033). It was also supported by Comunidad de Madrid (Spain) under project CYNAMON (P2018/TCS-4566), co-funded by FSE and European Union FEDER funds. The work of the second author was partially supported by Spanish grant VIGROB-287 of the University of Alicante.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank Fausto Montoya and Amalia B. Orúe for kindly sharing some of their visual programs, which were applied in the frame of this work to check the randomness of the sequences generated. They would also like to thank Miguel Beltrá for the help provided during the computational calculations.

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.

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
PRNG	Pseudo-Random Number Generator
LFSR	Linear Feedback Shift Register
LC	Linear Complexity
PN-sequence	Pseudo Noise-sequence
SG	Shrinking Generator
MAC	Message Authentication Code

References

1. Gallegos-Segovia, P.; Bravo-Torres, J.; Argudo-Parra, J. 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.
2. Biryukov, A.; Perrin, L. State of the Art in Lightweight Symmetric Cryptography. Cryptology ePrint Archive, Report 2017/511, 2017. Available online: https://ia.cr/2017/511 (accessed on 3 April 2022).
3. Chin, W.; Li, W.; Chen, H. Energy big data security threats in IoT-based smart grid communications. *IEEE Commun. Mag.* **2017**, *55*, 70–75. [CrossRef]
4. Mavromoustakis, C.; Mastorakis, G.; Batalla, J. *Internet of Things (IoT) in 5G Mobile Technologies*; Springer: Berlin/Heidelberg, Germany, 2016.

5. National Institute of Standards and Technology (NIST). NIST Lightweight Cryptography Project. Technology Administration. 2022. Available online: https://csrc.nist.gov/Projects/Lightweight-Cryptography (accessed on 3 April 2022).
6. Zia, U.; McCartney, M.; Scotney, B.; Martinez, J.; Sajjad, A. A novel pseudo-random number generator for IoT based on a coupled map lattice system using the generalised symmetric map. *SN Appl. Sci.* **2022**, *4*, 48. [CrossRef]
7. Kietzmann, P.; Schmidt, T.C.; Wählisch, M. A Guideline on Pseudorandom Number Generation (PRNG) in the IoT. *ACM Comput. Surv.* **2021**, *54*, *1–38* . [CrossRef]
8. Golomb, S.W. *Shift Register-Sequences*; Aegean Park Press: Laguna Hill, CA, USA, 1982.
9. Coppersmith, D.; Krawczyk, H.; Mansour, Y. The shrinking generator. In *Advances in Cryptology—CRYPTO'93*; Stinson, D., Ed.; Springer: Berlin/Heidelberg, Germany, 1994; Volume 773, pp. 22–39. [CrossRef]
10. Cardell, S.D.; Fúster-Sabater, A. Modelling the shrinking generator in terms of linear CA. *Adv. Math. Commun.* **2016**, *10*, 797–809. [CrossRef]
11. Cardell, S.D.; Climent, J.J.; Fúster-Sabater, A.; Requena, V. Representations of Generalized Self-Shrunken Sequences. *Mathematics* **2020**, *8*, 1006. [CrossRef]
12. Cardell, S.D.; Fúster-Sabater, A.; Requena, V. Interleaving Shifted Versions of a PN-Sequence. *Mathematics* **2021**, *9*, 687. [CrossRef]
13. Pichler, F. (Ed.) Linear Complexity and Random Sequences. In *Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 1986; Volume 219.
14. Duvall, P.F.; Mortick, J.C. Decimation of Periodic Sequences. *SIAM J. Appl. Math.* **1971**, *21*, 367–372. [CrossRef]
15. Fúster-Sabater, A.; Caballero-Gil, P. Linear solutions for cryptographic nonlinear sequence generators. *Phys. Lett. A* **2007**, *369*, 432–437. [CrossRef]
16. Lidl, R.; Niederreiter, H. *Introduction to Finite Fields and Their Applications*; Cambridge University Press: New York, NY, USA, 1986.
17. Mita, R.; Palumbo, G.; Pennisi, S.; Poli, M. Pseudorandom bit generator based on dynamic linear feedback topology. *Electron. Lett.* **2002**, *28*, 1097–1098. [CrossRef]
18. Ali Eljadi, F.M.; Taha Al Shaikhli, I.F. Dynamic linear feedback shift registers: A review. In Proceedings of the 5th International Conference on Information and Communication Technology for The Muslim World (ICT4M), Kuching, Malaysia, 17–18 November 2014; pp. 1–5. [CrossRef]
19. Peinado, A.; Munilla, J.; Fúster-Sabater, A. Improving the Period and Linear Span of the Sequences Generated by DLFSRs. In Proceedings of the International Joint Conference SOCO'14-CISIS'14-ICEUTE'14, Advances in Intelligent Systems and Computing, Bilbao, Spain, 25–27 June 2014; de la Puerta, J.G., Ferreira, I.G., Bringas, P.G., Klett, F., Abraham, A., de Carvalho, A.C., Herrero, Á., Baruque, B., Quintián, H., Corchado, E., Eds.; Springer International Publishing: Cham, Switzerland, 2014; Volume 299, pp. 397–406. [CrossRef]
20. Stępień, R.; Walczak, J. Comparative analysis of pseudo random signals of the LFSR and DLFSR generators. In Proceedings of the 20th International Conference Mixed Design of Integrated Circuits and Systems—MIXDES 2013, Gdynia, Poland, 20–22 June 2013; pp. 598–602.
21. Xiong, H.; Qu, L.; Li, C.; Fu, S. Linear complexity of binary sequences with interleaved structure. *IET Commun.* **2013**, *7*, 1688–1696. [CrossRef]
22. Massey, J.L. Shift-register synthesis and BCH decoding. *IEEE Trans. Inf. Theory* **1969**, *15*, 122–127. [CrossRef]
23. Golomb, S.W.; Parker, M.; Pott, A.; Winterhof, A. In Proceedings of the Sequences and Their Applications—SETA 2008, Lexington, KY, USA, 14–18 September 2008; Volume 5203.
24. National Institute of Standards and Technology. *FIPS 140-2: Security Requirements for Cryptographic Module. Federal Information Processing Standards Publication*; U.S. Department of Commerce: Washington, DC, USA, 2001. Available online: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf (accessed on 3 April 2022).
25. Barnsley, M. *Fractals Everywhere*, 2nd ed.; Academic Press: Cambridge, MA, USA, 1988.
26. Peitgen, H.; Jurgens, H.; Saupe, D. *Chaos and Fractals: New Frontiers of Science*; Springer: Berlin/Heidelberg, Germany, 2004.
27. Orúe, A.; Fúster-Sabater, A.; Fernández, V.; Montoya, F.; Hernández, L.; Martín, A. Actas de la XIV Reunión Española sobre Criptología y Seguridad de la Información, RECSI XIV. Available online: https://alarcos.esi.uclm.es/DocumentosWeb/2016-RECSI-Moreno.pdf (accessed on 3 April 2022).
28. Romera, M. Técnica de Los Sistemas Dinámicos Discretos. *Textos Univ. CSIC* **1997**,*27*, 50–58.
29. Álvarez, G.; Montoya, F.; Romera, M.; Pastor, G. Cryptanalyzing an improved security modulated chaotic encryption scheme using ciphertext absolute value. *Chaos Solitons Fractals* **2005**, *23*, 1749–1756. [CrossRef]
30. National Institute of Standards and Technology (NIST). A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. 2010. Available online: http://csrc.nist.gov/publications/nistpubs800/-22rec1/SP800-22red1.pdf (accessed on 3 April 2022).
31. Maurer, U. A universal statistical test for random bit generators. *J. Cryptol.* **1992**, *5*, 89–105. [CrossRef]
32. Marsaglia, G. The Marsaglia Random Number CDROM Including the Diehard Battery of Tests of Randomness. 1995. Available online: https://web.archive.org/web/20160125103112/http://stat.fsu.edu/pub/diehard/ (accessed on 3 April 2022).