mathematics

MDPI

Article

Gaussian Pseudorandom Number Generator Using Linear Feedback Shift Registers in Extended Fields

Guillermo Cotrina *, Alberto Peinado and Andrés Ortiz

Department Ingeniería de Comunicaciones, E.T.S. Ingeniería de Telecomunicación, Universidad de Málaga, Campus de Teatinos, 29071 Málaga, Spain; apeinado@ic.uma.es (A.P.); aortiz@ic.uma.es (A.O.)
* Correspondence: gcotrinacuenca@uma.es

Abstract: A new proposal to generate pseudorandom numbers with Gaussian distribution is presented. The generator is a generalization to the extended field $GF(2^n)$ of the one using cyclic rotations of linear feedback shift registers (LFSRs) originally defined in $GF(2)$. The rotations applied to LFSRs in the binary case are no longer needed in the extended field due to the implicit rotations found in the binary equivalent model of LFSRs in $GF(2^n)$. The new proposal is aligned with the current trend in cryptography of using extended fields as a way to speed up the bitrate of the pseudorandom generators. This proposal allows the use of LFSRs in cryptography to be taken further, from the generation of the classical uniformly distributed sequences to other areas, such as quantum key distribution schemes, in which sequences with Gaussian distribution are needed. The paper contains the statistical analysis of the numbers produced and a comparison with other Gaussian generators.

Keywords: LFSR; Gaussian distribution; extended fields; central limit theorem

Citation: Cotrina, G.; Peinado, A.; Ortiz, A Gaussian Pseudorandom Number Generator Using Linear Feedback Shift Registers in Extended Fields. *Mathematics* **2021**, *9*, 556. https://dx.doi.org/10.3390/math9050556

Academic Editor: Luis Hernández Encinas

Received: 19 January 2021
Accepted: 25 February 2021
Published: 6 March 2021

1. Introduction

Random number generators are of vital importance in many areas and, particularly, in cryptography. Most cryptographic algorithms and protocols make use of random or pseudorandom numbers. Encryption and authentication schemes in wireless and mobile communications, such as Bluetooth [1], IEEE 802.15.4, IEEE 802.11 WLAN [2], GSM [3] or LTE [4], employ pseudorandom numbers; radio frequency identification [5] standards define and recommend the utilization of true random numbers [6].A large part of the pseudo-random number generators (PRNGs) used in cryptography are based on linear feedback shift registers (LFSRs), mainly due to their simplicity, low cost of implementation, good statistical behavior and the possibility of using a mathematical model that allows the generator to be designed for an optimal performance [7].

In fact, the maximal sequence length generated by an LFSR of m cells is $2^m - 1$. However, those sequences suffer from a high predictability in such a way that the whole sequence can be reproduced if an eavesdropper gains access to $2m$ bits. Despite that, the LFSR is still an important part of the cryptographic generators because those sequences are used to derive more robust ones but keeping the original statistical properties. Two main methods are applied to fix that weakness: nonlinear combination and nonlinear filtering. The former is based on several LFSR, usually with different number of cells [3], and the latter on a unique LFSR whose sequence is processed (filtered) by a nonlinear function [4].

Another advantage of using LFSRs in cryptography is that the sequences generated have a uniform statistical distribution. For all these reasons, there is a lot of published works related to the LFSR, but only a few regarding its utilization to produce numbers with Gaussian distribution.

More precisely, in 2010, Kang [8] proposed a Gaussian PRNG, using a LFSR of length $N = 4M$ bits, to generate pseudorandom numbers of $(M + 4)$ bits. The numbers were

produced by means of an accumulator applied on decimated M-bits numbers, producing a sequence of length $(2^N - 1)/(8N)$. In order to fix this LFSR oversizing, Condo et al [9] proposed, in 2015, a Gaussian PRNG using permutations over the successive states of a unique LFSR, thus reducing the implementation cost. The main drawback is that not all the possible permutations yield numbers with the required Gaussian distribution and, consequently, a high computational effort must be performed to find the suitable permutations. Later, in 2020, Cotrina et al [10] presented an improvement of the Condo's PRNG focusing on a particular type of permutations of the LFSR state, the cyclic rotations. The authors concluded that more than 90% of the cycle rotations are usable for the PRNG and produce Gaussian distributed numbers.

These proposals are based on the central limit theorem (CLT) [11], trying to obtain a Gaussian distribution using sequences of uniformly distributed numbers. In this sense, the first proposal [12] employed several LFSRs to generate independent sequences of numbers. However, most of the proposals use a unique LFSR to generate all the sequences, such as those of Kang [8], Condo [9] and Cotrina [10].

Following the same approach, the present paper describes a Gaussian PRNG based on an LFSR operated and defined in an extension field $GF(2^n)$ instead of using the binary field $GF(2)$. An LFSR in $GF(2^n)$ can be represented as a combination of n LFSR in $GF(2^n)$. This fact, as it is described later, allows a particular implementation of the CLT. Furthermore, as it can be deduced from the equivalent model, the cyclic rotations proposed by Cotrina [10], as a particular case of the permutations proposed by Condo [9], are implicitly included in the operations of an LFSR in $GF(2^n)$. This proposal is also in line with the trend of using cryptographic algorithm and protocols in extended fields to take advantage of the word length of processors [13].

On the other hand, the proposed generator is a way to keep using the LFSR as a basic element to generate pseudorandom numbers in cryptographic areas where other than uniform distribution is required. An example of this is quantum key distribution (QKD) schemes [14]. This type of scheme, designed to establish keys between two endpoints, can be considered the most mature application of quantum communications. Currently, all developed countries have deployed QKD schemes, some experimentally (in controlled environments) and others in their current transit networks [15]. The first QKD schemes were based on the transmission of polarized photons using non-orthogonal states. These schemes, named discrete-variable QKD (DV-QKD) [16,17], require the utilization of specific components for single-photon detection. A different type of QKD scheme has been developed to carry information on some continuous properties of the light, such as the values of the quadrature components of a coherent state. The so-called continuous-variable QKD (CV-QKD) [18–21], currently deployed in several countries e.g., China, Japan, Spain and Italy [15] present a lower implementation cost due to the utilization of standard communications components. They use coherent detection techniques usually employed in classical optical communications. Furthermore, they employ Gaussian modulation to send random amplitude and phase values that must be generated following a Gaussian distribution [10,14,22].

Section 2 introduces the fundamentals of the LFSR in the binary and extended Galois fields. Section 3 describes the binary equivalent model of an LFSR defined in a $GF(2^n)$ and the proposal of a Gaussian PRNG based on this model. The statistical analysis of the numbers produced by the proposed PRNG is presented in Section 4 and compared to the Box–Muller [23,24] algorithm, a well-known algorithm not based on CLT. Finally, conclusions are presented in Section 5.

2. LFSR Fundamentals

In this section the basic properties of the linear feedback shift register (LFSR) (see Figure 1), and its generated sequences are described.

A linear feedback shift register (LSFR) is a shift register that takes a linear function of a previous state as an input. Most commonly. The LFSR [25] of length m consists of m

stages numbered $0, 1, 2, \cdots m-1$, each capable of storing one bit and having one input and one output and a clock which controls the movement of data.

Definition 1. *An LFSR of length m consists of: A linear feedback shift register (LFSR) of length m consists of m stages numbered* $0, 1, 2, \cdots m-1$*, each capable of storing one bit and having one input and one output; and a clock which controls the movement of data. During each unit of time the following operations are performed:*

- *The content of stage 0 is output and forms part of the output sequence.*
- *The content of stage i is moved to stage* $i-1$ *for each i where* $1 \leq i \leq m-1$.
- *The new content content of state* $m-1$ *is the feedback bit* a_j *which is calculated by adding together modulo* 2 *the previous contents of a fixed subset of stages* $0, 1, \cdots, m-1$.

The values q_i *are either* 0 *or* 1 *and the feedback bit* a_j *is the modulo* 2 *sum of the contents of those stages i,* $1 \leq i \leq m-1$*, for which* $q_{m-i} = 1$*. As a consequence, the output sequence of the LFSR is* $A = (a_0, a_1, a_2, \cdots)$ *and is uniquely determined by the following recursion:*

$$a_j = f(a_{j-m}, a_{j-m+1}, \cdots, a_{j-1}) = q_0 \cdot a_{j-m} + q_1 \cdot a_{j-m+1} + \cdots + q_m \cdot a_{j-1} \tag{1}$$

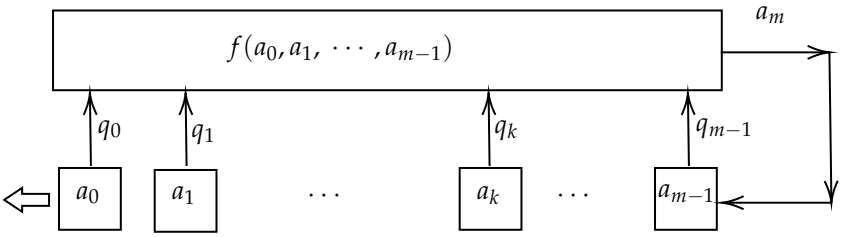

Figure 1. A Linear Feedback Shift Register of length m.

The evolution of the LFSR and their sequences generated can be performed by means of a polynomial whose coefficients are the values q_i that represents the stages used to compute the feedback bit a_j. For this reason, the LFSR is denoted $\langle m, p(x) \rangle$, where $p(x) = 1 + q_1 x + q_2 x^2 + \cdots + q_m x^m$ is the connection polynomial.

The LFSR is said to be nonsingular if the degree of $p(x)$ is m (that is, $q_m = 1$). If the initial content of stage i is $a_i \in \{0, 1\}$ for each i, $0 \leq i \leq, m-1$, then $[a_{m-1}, \cdots, a_1, a_0]$ is called the initial state or seed of the LFSR.

On the other hand, the state of the LFSR at the time t is denoted as

$$s^{(t)} = [a_{m-1+t}, \cdots, a_{t+1}, a_t] \tag{2}$$

which corresponds to the application of the recursion in the Equation (1) t consecutive times starting with the seed $s^{(0)} = [a_{m-1}, \ldots, a_1, a_0]$

Example 1. *Consider the LFSR* $\langle 4, 1 + x + x^4 \rangle$*. If the initial state of the LFSR is* $s^{(0)} = [0, 0, 0, 0]$*, the output sequence is the zero sequence* $A = (0, 0, \cdots)$*. For the initial state* $s^{(0)} = [0, 1, 1, 0]$*, the sequence has a length of* 15 *The following table shows the successive states* $s^{(t)}$*. Note that the rightmost bit of each state constitutes the output sequence* $A = (0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, \cdots)$ *as we can see in Table 1.*

Table 1. Values of the Linear Feedback Shift Register (LFSR) $\langle 4, 1 + x + x^4 \rangle$ whose initial state is $[0, 1, 1, 0]$.

t	$s^{(t)}$				t	$s^{(t)}$			
0	0	1	1	0	8	1	1	1	0
1	0	0	1	1	9	1	1	1	1
2	1	0	0	1	10	0	1	1	1
3	0	1	0	0	11	1	0	1	1
4	0	0	1	0	12	0	1	0	1
5	0	0	0	1	13	1	0	1	0
6	1	0	0	0	14	1	1	0	1
7	1	1	0	0	15	0	1	1	0

Definition 2 (cf. [25]). *An output sequence $\boldsymbol{A} = (a_0, a_1, \cdots)$ generated by an LFSR $\langle m, p(x) \rangle$, is said to be periodic if there exits $j_0 \in \mathbb{N}$ such that $a_i = a_{i+j_0}$ $\forall i \in \mathbb{N}$. Such j_0 is called period of the sequence.*

From this definition, the sequence of the example is a periodic sequence with period $L = 15$.

One of the advantages of LFSR is the mathematical model that allows one to predict the length of the sequences generated. The following definition and theorem states how and when the maximal length is reached by the sequences.

Definition 3 (cf. [25]). *If $p(x) \in \mathbb{Z}_2[x]$ is a connection polynomial of degree m, then $\langle m, p(x) \rangle$ is called a maximum length LFSR if the output sequence, with non-zero initial state, has period $2^m - 1$. This sequence is called m-sequence.*

Definition 4. *A polynomial $p(x)$ in $GF(2)[x]$ is irreducible if it cannot be factored into a product of lower-degree polynomials in $GF(2)[x]$. An irreducible polynomial $p(x) \in GF(2)[x]$ of degree $m \in \mathbb{N}$ is said to be primitive if $min_{n \in \mathbb{N}}\{n : p(x) \mid (x - 1)\} = 2^m - 1$.*

Theorem 1 (cf. [7]). *An output sequence $\boldsymbol{A}$ generated by an LFSR $\langle m, p(x) \rangle$ is an m-sequence if and only if the connection polynomial $p(x)$ is a primitive polynomial. The sequence length is independent of the initial state.*

Consequently, a primitive polynomial of degree m will generate a sequence of length $2^m - 1$ and the LFSR will run through 2^{m-1} different nonzero states, that is, all possible nonzero states. Hence, if we consider each state as an m-bit pseudorandom number, we can say that LFSR produce numbers with uniform distribution.

Besides its maximal length, the m-sequences have many desirable statistical properties that can be summarized in the three Golomb's postulates [7]. Given a periodic binary sequence $A = (a_i)_{i \in \mathbb{N}}$ with period length $L = 2^m - 1$, it is said to be pseudoradom if the following postulates hold.

1. Balance property. In every period, the number of zeros is nearly equal to the number of ones (the disparity does not exceed 1, or $\mid \sum_{i=0}^{m-1} (-1)^{a_i} \mid \leq 1$.
2. In every period, half of the run have length 1, one fourth have length 2, one eighth have length 3, and so on. For each of these lengths there are the same number of runs of 0's and runs of 1's.
3. Two level autocorrelation. The autocorrelation function $c(\tau)$ is two-valued given by

$$c(\tau) = \begin{cases} m & \text{if} \quad \tau = 0 \bmod m \\ k & \text{if} \quad \tau \neq 0 \bmod m \end{cases} \tag{3}$$

where k is a constant. If $k = -1$ for m odd, or $k = 0$ for m even, we say that the sequence has the ideal two level autocorrelation function.

4. The ideal $k-$ tuple distribution. In every period of A , if each nonzero $k-$ tuple $(a_1, a_2, a_3, \cdots, a_k)$ occurs q times and the zero $k-$tuple occurs $q^{n-k} - 1$ times, then we say that the sequence satisfies the ideal k-tuple distribution.

LFSR over $GF(2^n)$

Definition 5. *Let's suppose* $(F, +, *)$ *be a field with operations* $+$ *and* $*$*, we will say that this field* F *is a Galois field if the cardinal of field* F *is finite. If the cardinal of the finite field* F *is* p*, then* F *will be represented as* $GF(p)$*.*

It is possible to extend the prime field $GF(p)$ to a field of p^m elements, represented by $GF(p^m)$, which is called an extension field of $GF(p)$. In our case we shall work for $p = 2$.

To build up the field extension of $GF(2)$, $GF(2^n)$, let's consider $p(x)$ a primitive polynomial over $GF(2)$ of degree n, once this polynomial is defined, let's consider a root $alpha$ of this primitive polynomial, therefore $p(\alpha) = 0$. Let's consider $GF(2^n) = \left\{0, 1, \alpha, \alpha^2, \cdots, \alpha^{2^n-1}\right\}$. It can be proven that all the elements of $GF(2^n)$ can be represented by $2^n - 1$ distinct non-zero polynomials of α over $GF(2)$ with degree $m - 1$ or less. The $0 \in GF(2^n)$ can be represented as the zero polynomial. Then the set $GF(2^n)$ with the usual operations is a Galois field with 2^n elements.

Example 2. *Let's build the finite field* $GF(2^4)$ *generated by the primitive polynomial* $p(x) = 1 + x + x^4$*. First of all, we consider* α *to be a root of the primitive polynomial* $p(x)$*, then we determine all multiplicative powers of* α*,* $\alpha^0 = 1, \alpha^1, \cdots$ *until we obtain the multiplicative identity* 1 *on this field. For any Galois field we will have three representations as shown in Table 2.*

This Table 2 shows how all the elements of the extended Field are obtained. The equivalence has been obtained using the vector notation as a function of one of the roots of the primitive polynomial that generates the extended field. In the same way, it can be observed that the body is generated cyclically and that no equal values are obtained except when it has been cycled and all its elements have been obtained periodically.

Table 2. Representation of the Galois Field $GF(2^4)$ over $GF(2)$ with primitive polynomial $p(x) = 1 + x + x^4$ over $GF(2)$.

Power Representation	Polynomial Representation	Vector Representation
0	0	$(0, 0, 0, 0)$
1	1	$(1, 0, 0, 0)$
α	α	$(0, 1, 0, 0)$
α^2	α^2	$(0, 0, 1, 0)$
α^3	α^3	$(0, 0, 0, 1)$
α^4	$1 + \alpha$	$(1, 1, 0, 0)$
α^5	$\alpha + \alpha^2$	$(0, 1, 1, 0)$
α^6	$\alpha^2 + \alpha^3$	$(0, 0, 1, 1)$
α^7	$1 + \alpha + \alpha^3$	$(1, 1, 0, 1)$
α^8	$1 + \alpha^2$	$(1, 0, 1, 0)$
α^9	$\alpha + \alpha^3$	$(0, 1, 0, 1)$
α^{10}	$1 + \alpha + \alpha^2$	$(1, 1, 1, 0)$
α^{11}	$\alpha + \alpha^2 + \alpha^3$	$(0, 1, 1, 1)$
α^{12}	$1 + \alpha + \alpha^2 + \alpha^3$	$(1, 1, 1, 1)$
α^{13}	$1 + \alpha^2 + \alpha^3$	$(1, 0, 1, 1)$
α^{14}	$1 + \alpha^3$	$(1, 0, 0, 1)$

According to the Example 2, form this point on, we shall represent the elements of the extended fields using the vector notation, where for example the element $1 + \alpha^2 + \alpha^3$ will be represented as $(1, 0, 1, 1)$.

To generate an LFSR on a $GF(2^n)$ we will start by determining a primitive polynomial p over $GF(2)$ that will be used to generate the extended field. Once this polynomial and this field have been set, we choose a primitive polynomial g over $GF(2^n)$ and a seed or non-zero initial state value β_0 on which the primitive polynomial g is applied. In this way, a value β_1, is obtained by $g(\beta_1)$ on which the primitive polynomial g will be applied and thus the process will be repeated.

Example 3. *We consider the $GF(2^6)$ that has been built using the primitive polynomial $q(x) = 1 + x + x^6$. Let's consider the primitive polynomial $p(x)$ over $GF(2^6)$ whose vector representation is $p(x) = \{\{1,0,1,1,0,1\},\{1,0,0,1,0,1\},\{1,1,0,0,0,1\},\{1,1,0,1,1,0\},\{1,0,0,1,0,1\}\}$. The initial seed will be*

$$\{\{1,0,1,1,0,1\},\{1,0,0,1,0,1\},\{1,1,0,0,0,1\},\{1,1,0,1,1,0\},\{1,0,0,1,0,1\}\} \quad (4)$$

and taking into consideration these conditions the output sequence will be

$$A = (\{1,0,1,1,1,0\},\{0,1,0,1,0,1\},\{1,0,1,1,1,0\},\{0,1,1,0,0,0\},\cdots) \quad (5)$$

As in $GF(2)$, the LFSR in $GF(2^n)$ produces an m-sequence if and only of the connection polynomial is primitive. The $m-$ sequence has a period of $2^{m \cdot n} - 1$ that corresponds to all nonzero states, where m is the degree of the connection polynomial.

3. Gaussian Number Generator over $GF(2^n)$

To apply the CLT, Gaussian generators based on a single LFSR [8–10] try to obtain different sequences of pseudo-random numbers with uniform distribution from the same sequence generated by the LFSR. To do this, they apply permutations—generic in some cases, rotations in others—on each state of the LFSR. In this way, with each number generated by the LFSR, other numbers are also being generated (as many as permutations are applied) and therefore different uniform sequences suitable to be combined in a sequence of numbers are being simultaneously constructed. LFSRs defined on an extended field can be analyzed as the combination of a series of binary LFSRs. For this reason, each state of an LFSR in an extended field is related to the states of a series of binary LFSRs. This relation is the one used to propose a PRNG with Gaussian distribution. Next, the equivalent model of an LFSR defined in an extended field that allows the definition of a Gaussian PRNG with excellent performance is described.

3.1. Binary Equivalent Model of an LFSR in $GF(2^n)$

As it is known [26], there is a relationship between the $m-$ sequences generated in $GF(2^n)$ and those generated in $GF(2)$, in such a way that the former can be obtained from the latter. The relationship is established from the feedback polynomials that define each LFSR. Specifically, if $q(x)$ is the primitive polynomial of degree m that defines the LFSR in $GF(2^n)$ and therefore generates an $m-$sequence in $GF(2^n)$, the decimated sequences obtained by taking the $j-$th bit of each element in the $m-$sequence are generated by a binary LFSR with primitive feedback polynomial $h(x)$ of degree $m \cdot n$, where $h(x)$ divides $q(x)$. This allows to develop an equivalent model of the LFSR defined in $GF(2^n)$ using n LFSRs in $GF(2)$, as shown in Figure 2 where the $j-$th bit of each element is generated by the $j-$th binary LFSR. Note that all the binary LFSRs used in the model have the same polynomial $h(x)$ although initialized in different seeds. Therefore, the sequences generated by the binary LFSRs are $m-$sequences in $GF(2)$, and consequently, shifted versions of the same sequence. In this way, the generation of an $m-$sequence in $GF(2^n)$ implies the generation of n binary $m-$ sequences, one for each bit, which can be used as sources with a uniform distribution to later apply CLT.

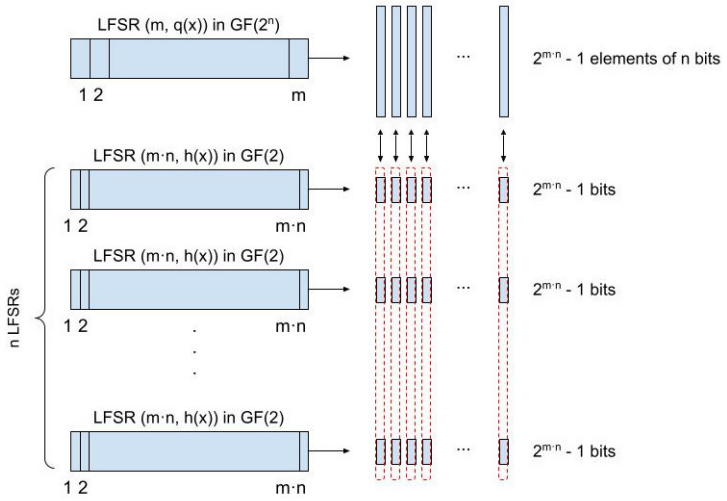

Figure 2. Binary equivalent model of an LFSR in $GF(2^n)$.

Definition 6. *Let $\langle m, q(x)\rangle$ be a LFSR over $GF(2^n)$. Let $s(t) = [a_{m-1+t}, a_{m-2+t}, \cdots, a_t]$ be the state of the LFSR at time t, with $a_i \in GF(2^n)$ that can be represented as the vector $[a_{i,n-1}, \ldots, a_{i,1}, a_{i,0}]$ with $a_{i,j} \in GF(2)\ \forall\ 0 \leq j \leq n-1$. Then, Π_k is defined as the projection over the $k-th$ component of an element. Hence, $\Pi_k(a_i) = a_{i,k}$ and $\Pi_k(s(t)) = s(t)_k = [a_{m-1+t,k}, a_{m-2+t,k}, \cdots, a_{t,k}]$.*

This definition allows us to represent the decimated sequences $\Pi_k(A)$ of a sequence $A = (a_0, a_1, a_2, \cdots)$ as:

$$\Pi_k(A) = (\Pi_k(a_0), \Pi_k(a_1), \Pi_k(a_2), \cdots) \tag{6}$$

Theorem 2. *Let $\langle m, q(x)\rangle$ be an LFSR over $GF(2^n)$ and let $\langle r_k, f_k(x)\rangle$ be the set of all its decimated sequences $\forall k \in \{1, 2, \cdots, n\}$ then, $q(x)$ is primitive over $GF(2^n)$ if and only if $\forall k \in \{1, 2, \cdots, n\}$ f_k is primitive over $GF(2)$. If these conditions hold, $f_j(x) = f_k(x) = h(x)\ \forall j, k \in \{1, 2, \cdots, n\}$ and $m \mid r_k$*

Therefore, if we have an LFSR $\langle m, q(x)\rangle$ over $GF(2^n)$ where $q(x)$ is primitive, then we can consider that we have n's $m-$sequences over $GF(2)$.

In other words, this equivalent model represents the interleaving process that generates the sequence in $GF(2^n)$; that is, the sequence generated by the LFSR in $GF(2^n)$ can be expressed as an interleaved sequence, in the sense describe by Gong in [27], composed by n component sequences, corresponding with the decimated sequences. More precisely, it is a primitive interleaved sequence as all the component sequences are generated by the same primitive polynomial in $GF(2)$.

On the other hand, pseudorandom sequences must be difficult to reproduce. Linear complexity (or linear span) of a sequence is defined as the degree of the minimal polynomial that generates it, or equivalently the length of the shortest LFSR that generates it. Consequently, the linear complexity of a sequence generated by a primitive LFSR is the length of that LFSR. Hence, considering the sequence produced by an LFSR of m cells in $GF(2^n)$ as an interleaved sequence, its linear complexity LC_{ext} is n times the linear complexity LC_{bin} of its primitive components; that is,

$$LC_{ext} = n \cdot LC_{bin} = m \cdot n^2 \tag{7}$$

3.2. The Proposed Generator

Taking in mind that one of the potential applications of a Gaussian PRNG based in LFSR could be a QKD scheme, it is important to note the following requirements:

- The PRNG should allow a discrete set of values to be generated large enough to approximate the continuous probability distribution.

- The set of values generated must have a Gaussian probability distribution.
- The security of the system must allow the generation of a set of values with a sufficiently large cardinal.
- The generation of obtained values should be done as fast as possible and within the lowest implementation cost. In addition, for the system to be effective, the possibility of a hardware implementation must be considered.
- The system must allow the generation of the pseudo-random values with Gaussian distribution to be different for each of its different executions.

In order to fulfil all the requirements, we propose a PRNG based on an LFSR in $GF(2^n)$. The PRNG is composed by two units: the control unit and the processing unit (see Figure 3).The control unit consists in an LFSR with m cells, defined by a primitive polynomial $q(x)$ over $GF(2^n)$. This *unit* is responsible for the generation of the basic $m-$sequence from which all the sequences with uniform distribution are obtained for the subsequent application of CLT. As it is described in the previous sections, $if q(x)$ is primitive the LFSR generates an $m-$sequence, i.e., a sequence of maximal period $2^{n \cdot m} - 1$.

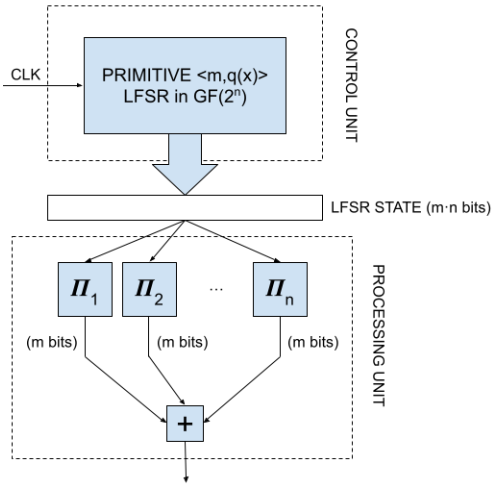

Figure 3. Gaussian number generator proposed.

The processing unit has been design with a two-fold objective. On the one hand, like many other cryptosystems, it applies a nonlinear filtering to the sequence produced by the LFSR in the control unit to increase the difficulty that an eavesdropper reproduces the whole sequence. On the other hand, this unit implements the operations that transform the statistical distribution from uniform to Gaussian, i.e., implements the CLT. To do that, the operator Π_k is applied on each LFSR state, thus producing n strings of m bits that corresponds with segments of the n binary sequences in the equivalent model described in Section 3.1. Hence, applying the CLT, the pseudorandom number B is obtained by means of the integer sum of those n bitstrings as:

$$B = D(s_1^t) + D(s_2^t) + \cdots + D(s_n^t) \tag{8}$$

where D is the function that maps an $m-$bit vector $x = (x_0, x_1, \cdots, x_{n-1})$ into a decimal value,

$$D(x) = \sum_{i=1}^{n-1} 2^i \cdot x_i \tag{9}$$

The period, range and accuracy of the proposed PRNG can be configured using two main parameters: m and n. The sequence of pseudorandom numbers has a period of $2^{mn} - 1$ because the feedback polynomial in the LFSR is a degree m primitive polynomial in $GF(2^n)$. The range of the numbers is mainly determined by m since each state of the LFSR contains $n \cdot m$ bits that split in n strings of m bits later summed to produce the

pseudorandom number. The parameter n increases slightly, until $m + log_2(n)$, the number of bits of the generated numbers due to the carry of the integer sum. Thus, for $m = 5$ and $n = 8$, the PRNG generates values with a range of $5 + 3 = 8$ bits.

Finally, when the CLT is applied to obtain a Gaussian distribution, its accuracy is related to the numbers of uniform distributed numbers that are summed. In this case, each pseudorandom number is obtained summing n values. Hence, generating $16-$bit pseudorandom numbers requires the utilization of an LFSR with 16 cells. If the LFSR operates in $GF(2^4)$, the PRNG will present a period of $2^{4 \cdot 16} - 1 = 2^{64} - 1$, obtained from the addition of 4 uniformly distributed sequences. The accuracy may increase using 8 values instead of 4. In that case, the LFSR would work in $GF(2^8)$ what would also increase the period to $2^{16 \cdot 8} - 1 = 2^{128} - 1$.

Regarding the speed of generation, although, generally, the use of LFSRs in $GF(2^n)$ is motivated by the speed increase that is achieved by generating n bits instead of 1 in each iteration, in this case, the use of the LFSR in $GF(2^n)$ pursues a second objective: to take advantage of the implicit relationship with n different binary sequences. This makes it easy to apply the CLT by using a single LFSR that is equivalent to n binary LFSRs. Therefore, the final rate of generating numbers with Gaussian distribution is the same as the rate at which a binary LFSR generates a bit. However, taking into account that the generated numbers are m bits long, the generation speed in bits per second is m times higher.

4. Statistical Analysis

In this section, the distribution of the numbers generated by the proposed PRNG is analyzed. Several normality tests have been applied to identify the configurations that generates numbers with Gaussian distribution.

4.1. Distribution Fit Test

Goodness-of-fit tests are used to evaluate how well a proposed model fits or predicts a particular data set. Usually, test statistics compute deviations between the observed data and predictions from the model. The value of a test statistic is said to be statistically significant if it is found to be within the rejection area of the distribution of the test statistic under the assumption that the model is true. The rejection area is often the upper its significance level α of 5% or 10% of the distribution frequencies. In our work we have set this acceptance minimum level to 10%. Therefore a distribution fit test performs a goodness of fit hypothesis test with null hypothesis H_0 that data was drawn from a population with a specific distribution of values, in this case the Normal distribution, and alternative hypothesis that it was not.

Usually, a statistical hypothesis test returns a value called p or the p-value. This value is used to reject or fail to reject the null hypothesis. This is done by comparing the p-value to a threshold value chosen beforehand called the significance level α. When the p-value is less than α, the default hypothesis can be rejected. In the same way, the confidence level of the test is $1 - \alpha$. If we set the significance level to 5% and the p-value is greater than 95%, we would conclude that the null hypothesis affirming that the data is distributed according to the Normal Distribution would not be rejected at the 5 percent significance level. In the present context, the higher the p-value, the better the data fits the normal distribution.

According to the CLT [21], if we consider $\{X_1, \ldots, X_n\}$ a random sample of size n that is, a sequence of independent and identically distributed random variables drawn from a distribution of expected value given by μ and finite variance given by σ^2. Suppose we are interested in the sample average $S_n = \frac{X_1 + \cdots + X_n}{n}$ of these random variables. By the law of large numbers, the sample averages converge in probability and almost surely to the expected value μ as $n \longrightarrow \infty$. The classical central limit theorem describes the size and the distributional form of the stochastic fluctuations around the deterministic number μ during this convergence. More precisely, it states that as n gets larger, the distribution of the difference between the sample average S_n and its limit μ, when multiplied by the factor $\sqrt{n}$ (that is $\sqrt{n}(S_n - \mu)$), approximates the normal distribution with mean 0 and

variance σ^2. For large n, the distribution of s_n is close to the normal distribution with mean μ and variance σ^2/n. The usefulness of the theorem is that the distribution of $\sqrt{n}(S_n - \mu)$ approaches normality regardless of the shape of the distribution of the individual Xi.

There exist different methods to distinguish whether or not the range of values in a distribution follows a Normal distribution. Due to the large number of values that we have obtained in all our tests, we have decided to use the Chi Square Test that will be described in the next Section 4.1.1.

4.1.1. Chi Square Test

The chi-square test [22,28] is used to test if a sample of data came from a population with a specific distribution. In this case we shall focus on this test to check if the distribution of numbers fits the normal distribution.

The χ^2 goodness-of-fit test examines the discrepancy between observed values and the values expected under some particular distribution of a random variable A.

The null hypothesis H_0: The random variable A follows the normal distribution.

The alternative hypothesis H_1: The random variable A does not follow the normal distribution.

Let $a_1, a_2, \cdots, a_m$ be the observed values of a variable A. We shall follow:

- Categorize the observations into k categories.
- Calculate the frequencies f_i where $k = \{1, 2, \cdots, n\}$ and f_i is the observed frequency of the category i.
- Let p_i be the probability, that under null hypothesis, the random variable A belongs to the category i. Calculate the expected frequencies $E_i = npi$ of the observations in category i.
- Now, under the null hypothesis, the random variables $f_1, f_2, \cdots, f_k$ follow multinomial distribution with parameters $n, p_1, p_2, \cdots, p_k$.
- We shall continue working out the test statistic

$$\chi_g^2 = \sum_{i=1}^{k} \frac{(f_i - E_i)^2}{E_i} \tag{10}$$

- If n is large, then under the null hypothesis, the test statistic χ_g^2 approximately follows $\chi_g^2(k - 1 - e)$ where e is the number of estimated parameters.
- The expected value of the test statistic, under the null hypothesis, is $k - 1 - e$.
- Large and small values of the test statistic (compared to the expected value) suggest that the null hypothesis H_0 does not hold.
- If the p-value is small enough, the null hypothesis H_0 is rejected.

4.1.2. Measures of Central Tendency, Dispersion, Kurtosis and Skewness

To check if these measures make the values fit in a feacient way with the data of a Gaussian distribution, we have normalized the results applying the elementary transformation

$$Z = \frac{X - \mu}{\sigma} \tag{11}$$

From here, we have determined the measures of central tendency of the variable, the measures of dispersion and the measures of skewness and kurtosis. The first thing that we have should verify is that if the values of the degree of the feedback polynomial are increased and the cardinal of the field is increased, the obtained values fit better to the normal distribution, obtaining in each case values closer to standard values of the normal distribution.

If the numerical data have been normalized, in the sense that we have applied the typification (11), then we can set the various control parameters to verify whether or not the data follow a normal distribution.

The expected values for the normal distribution are as follows:

1. Quartiles Q_1 and Q_3 to be $Q_1 = -0.67448, Q_3 = 0.67448$
2. About 95% of the observations are within 2 times the standard deviation of the mean. 95% of the values will be within 1.96 times the standard deviation from the mean (between -1.96 and 1.96). Approximately 68% of the observations are within one standard deviation of the mean (-1 to $+1$), and around 99.7% of the observations would be within three standard deviations of the mean (-3 to 3).
3. The standard deviation $\sigma = 1$ and the mean $\mu = 0$.
4. The skewness for a normal distribution is zero, and any symmetric data should have a skewness near zero. Negative values for the skewness indicate data that are skewed left and positive values for the skewness indicate data that are skewed right. By skewed left, we mean that the left tail is long relative to the right tail. Similarly, skewed right means that the right tail is long relative to the left tail. If the data are multi-modal, then this may affect the sign of the skewness.
5. The kurtosis for a standard normal distribution is 3.

4.2. Results

In this section we will go on to show the results that have been obtained for the generation of numbers with a Gaussian distribution.

To illustrate the results obtained in the generation of numbers with a Gaussian distribution, the following polynomials $\{q_i\}_{i\in\mathbb{N}}$ have been taken into account for the generation of the base Fields and the following polynomials as polynomials of connection polynomials $\{p_j\}_{j\in\mathbb{N}}$.

To generate all the extended fields, different primitive polynomials have been used. The degrees of these polynomials have ranged from 4 to 16. The list of primitive polynomials that have been used are represented in Table 3.

Table 3. List of primitive polynomial used for the extended field generation.

Degree (q_i)	**Polynomial** $q_i(x)$
4	$q_4 = x^4 + x + 1$
5	$q_5 = x^5 + x^2 + 1$
6	$q_6 = x^6 + x + 1$
7	$q_7 = x^7 + x + 1$
8	$q_8 = x^8 + x^4 + x^3 + x^2 + 1$
16	$q_{16} = x^{16} + x^{12} + x^3 + x^1 + 1$

We will continue describing the primitive polynomials that we have used as the connection polynomial for each of extended field. Note that due to the great length of each of the terms of each polynomial, the conversion to hex has been carried out.

From now on we will denote κ_{ij} to be the LFSR $\langle j, p_j \rangle$ so $j \in \{4,5,6,7,8,16\}$ represents the degree of the primtive polynomial over the extended Field $GF(2^i)$. Therefore, according to the Table 4, the LFSR $\kappa_{45}(x)$ will be the LFSR generated by $p(x) = \{\{1,0,0,0\},\{0,1,0,1\},\{0,0,1,1\},\{1,1,0,1\},\{1,0,0,1\},\{0,0,0,1\}\}$ over $GF(2^4)$.

First, we will determine the arithmetic mean, the standard deviation and the quartiles of the obtained numerical values. According to the Table 5, it can be seen that all the values obtained are within the range of expected values so that they fit to the values of a Gaussian distribution.

Table 4. List of primitive polynomial used for the extended field generated by each $q_i(x)$.

Polynomial $q_i(x)$	**Degree (p_i)**	$p_i(x)$
q_4	4	$\{a, 5, 9, 6\}$
q_4	5	$\{6, 5, a, 8, 9\}$
q_4	6	$\{6, 3, c, a, 9, b\}$
q_4	7	$\{6, 3, c, a, 2, 9, b\}$
q_4	8	$\{6, 3, c, a, d, 2, 9, b\}$
q_5	4	$\{15, a, 11, 16\}$
q_5	5	$\{15, a, 6, 17, 13\}$
q_5	6	$\{15, a, 17, 6, 17, 13\}$
q_5	7	$\{15, a, 17, 16, 6, 17, 13\}$
q_5	8	$\{15, a, 17, 14, 16, 6, 17, 13\}$
q_6	4	$\{2d, 25, 31, 36\}$
q_6	5	$\{2d, 25, 31, 36, 25\}$
q_6	6	$\{2d, 23, 25, 31, 36, 25\}$
q_6	7	$\{2d, 23, 25, 31, 14, 36, 25\}$
q_6	8	$\{2d, 23, 25, 31, 14, 36, 26, 25\}$
q_7	4	$\{45, 65, 55, 76\}$
q_7	5	$\{45, 65, 55, 65, 76\}$
q_7	6	$\{45, 65, 55, 55, 65, 57\}$
q_7	7	$\{45, 65, 55, 55, 61, 65, 57\}$
q_7	8	$\{45, 65, 63, 66, 55, 61, 65, 57\}$
q_8	4	$\{95, e5, 55, ed\}$
q_8	5	$\{aa, 55, 93, 65, 99\}$
q_8	6	$\{aa, 55, 93, b1, 65, 99\}$
q_8	7	$\{aa, 55, 74, 93, b1, 65, 99\}$
q_8	8	$\{80, aa, 0, 0, 0, 1\}$
q_{16}	24	$\{1002d, 10039, 1003f, 10053, 100bd, 100d7, 1012f, 1013d, 1014f, 1015d, 10197, 101a1, 101ad, 101bf, 101c7, 10215, 10219, 10225, 1022f, 1025d, 66157, 10285, 10291, 102a1\}$

In the same way we have generated all the quantiles and we have compared them with the theoretical values of those of the Gaussian distribution. In each of the cases it can be verified that the values fit the Gaussian model. In the Figure 4 we plot the obtained quantiles list against the quantiles list of a normal distribution for some cases.

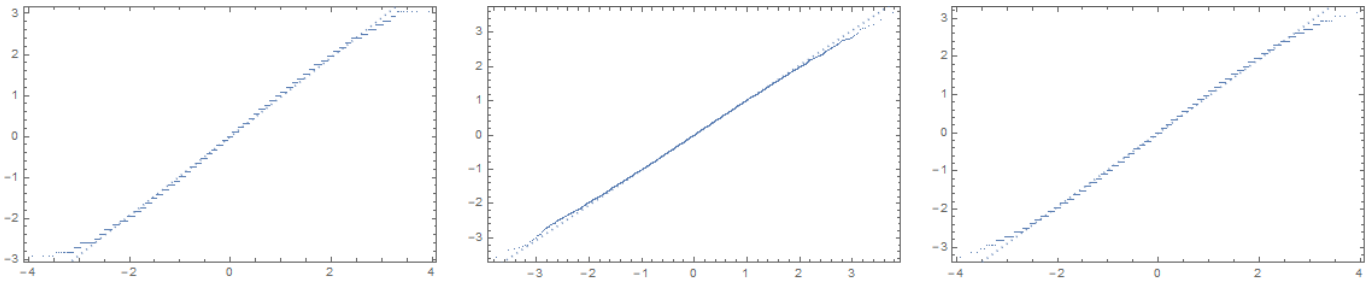

Figure 4. Quantiles plot against normal distribution for κ_{44}, κ_{58} and κ_{74} respectively.

In order to better support the results presented, we have represented the cumulative distribution function (CDF) and the results obtained and compared them with those expected in the normal distribution. In the Figure 5 we confront the CDFs of the normal distribution against those obtained resutls.

In this regard, it should also be noticed that the histograms of the results obtained have been analyzed and these have been compared within the corresponding histograms of the normal distribution. Continuity correction has been applied since a discrete set of values are being used and we have also been able to verify that the data fit a normal

distribution of values. In the Figure 6 we can see how the histograms tend to the normal ditribution CDFs curve.

Table 5. Mean and standard deviation for the different polynomial over extended fields.

LFSR generation	Mean μ	St. Deviation σ	$\{Q_1, Q_2, Q_3\}$
κ_{44}	$1.28113 \cdot 10^{-17}$	1.	$\{-0.652405, -0.000815332, 0.650774\}$
κ_{45}	$7.92835 \cdot 10^{-18}$	1.	$\{-0.652215, -0.000012122, 0.650214\}$
κ_{46}	$7.21716 \cdot 10^{-18}$	1.	$\{-0.6754, -0.0100635, 0.699629\}$
κ_{47}	$4.02607 \cdot 10^{-18}$	1.	$\{-0.699102, 0.0137925, 0.696134\}$
κ_{48}	$2.62951 \cdot 10^{-17}$	1.	$\{-0.696669, -0.000285917, 0.696097\}$
κ_{54}	$-2.60767 \cdot 10^{-17}$	1.	$\{-0.660901, -0.00813484, 0.644631\}$
κ_{55}	$-3.87947 \cdot 10^{-17}$	1.	$\{-0.671761, -0.0087515, 0.676358\}$
κ_{56}	$4.76993 \cdot 10^{-17}$	1.	$\{-0.671761, -0.0087515, 0.676358\}$
κ_{57}	$1.74443 \cdot 10^{-17}$	1.	$\{-0.680723, 0.00183247, 0.684388\}$
κ_{58}	$3.47529 \cdot 10^{-17}$	1.	$\{-0.687545, -0.0026389, 0.691847\}$
κ_{64}	$3.90608 \cdot 10^{-17}$	1.	$\{-0.659401, 0.000866493, 0.661134\}$
κ_{65}	$3.20183 \cdot 10^{-17}$	1.	$\{-0.711603, -0.026917, 0.706675\}$
κ_{66}	$2.84824 \cdot 10^{-17}$	1.	$\{-0.685646, -0.000279343, 0.685087\}$
κ_{67}	$7.32401 \cdot 10^{-17}$	1.	$\{-0.688653, 0.0113709, 0.680959\}$
κ_{68}	$2.24457 \cdot 10^{-17}$	1.	$-0.67203, -0.00582178, 0.68299$
κ_{74}	$1.26656 \cdot 10^{-17}$	1.	$\{-0.65119, 0.000928774, 0.653048\}$
κ_{75}	$8.49908 \cdot 10^{-18}$	1.	$\{-0.699352, -0.0176537, 0.712738\}$
κ_{76}	$8.49908 \cdot 10^{-18}$	1.	$\{-0.673098, -0.00289631, 0.667306\}$
κ_{77}	$1.39497 \cdot 10^{-18}$	1.	$\{-0.694148, 0.00888319, 0.701576\}$
κ_{78}	$-4.53049 \cdot 10^{-18}$	1.	$\{-0.693528, 0.0000646841, 0.703291\}$
κ_{84}	$5.43506 \cdot 10^{-17}$	1.	$\{-0.649619, 0.0029409, 0.655501\}$
κ_{85}	$3.3536 \cdot 10^{-18}$	1.	$\{-0.698031, -0.0244844, 0.697172\}$
κ_{86}	$1.42056 \cdot 10^{-17}$	1.	$\{-0.693034, -0.00748555, 0.700178\}$
κ_{87}	$-3.13666 \cdot 10^{-18}$	1.	$\{-0.685563, -0.00883622, 0.69865\}$
κ_{88}	$-3.93139 \cdot 10^{-19}$	1.	$\{-0.687156, -0.0022108, 0.682735\}$
κ_{1624}	$1.90913 \cdot 10^{-17}$	1.	$\{-0.690607, -0.005302, 0.672002\}$

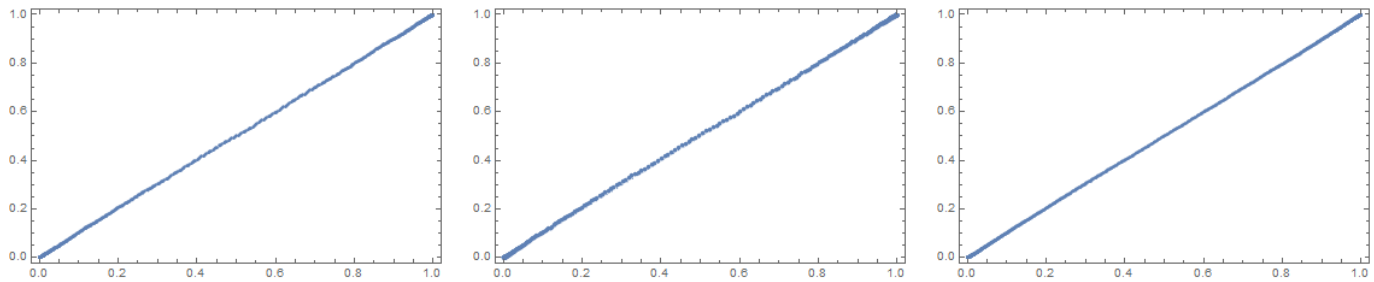

Figure 5. Plots of the cumulative distribution function (CDF) of list against the CDF of a normal distribution for κ_{67}, κ_{86} and κ_{88} respectively.

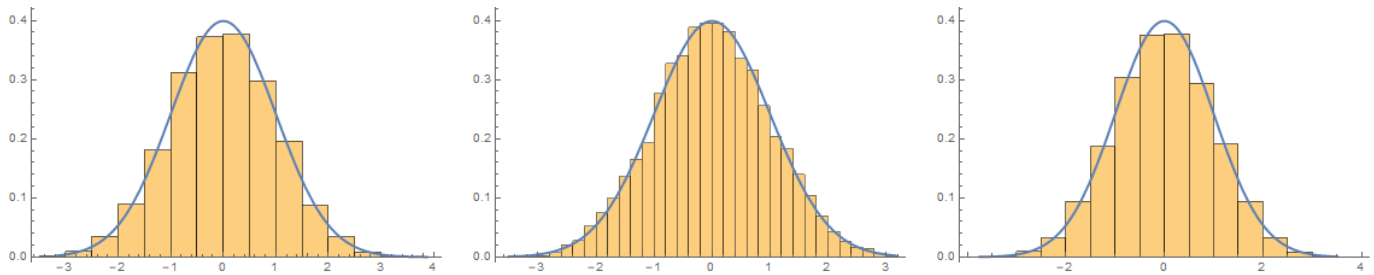

Figure 6. Plots of the histogrmas of the obtained results against their corresponding in a normal distribution for κ_{57}, κ_{68} and κ_{88} respectively.

Although different normality tests were originally used to check if the data fit the normal distribution, as we can see in Table 6, due to the large number of data, we have opted for the method of the Chi-Square Test, which allows us to check whether or not a model or theory follows an approximately normal distribution.

Table 6. Distribution fit tests for κ_{44}.

Distribution Fit Tests	Statistic	*p*-Value
Anderson–Darling	0.16486	0.956321
Cramér–von Mises	0.0248859	0.924378
Jarque–Bera ALM	0.214813	0.894745
Mardia Combined	0.214813	0.894745
Mardia Kurtosis	0.00564325	0.995497
Mardia Skewness	0.205783	0.650093
Pearson χ^2	13.4135	0.966447
Shapiro–Wilk	0.997609	0.907292

These described tests have been used to check whether the statistical variables are distributed according to a Normal distribution. A minimum level of confidence has been set to 90% therefore the significance level is set to 10% and according to that level of confidence the sequence obtained has been screened. That is, given a set of values, the Normality tests have been applied to verify whether the data followed a normal distribution or not. The output of these mentioned tests is a p-test. If the p-test value obtained is greater than 90%, the sequence obtained is considered ***valid*** and otherwise has been discarded.

The proposed generator has been tested for all possible polynomials. All primitive polynomial combinations have been tested using the Mathematica environment. The tests have been performed using Chi sqaure, the Anderson–Darling and Shapiro methods. The results are exemplified in Table 6.

The proposed PRNG [24] has also been compared with the Box–Muller algorithm that was designed as a pseudo-random number sampling method for generating pairs of independent, standard, normally distributed (zero expectation, unit variance) random numbers, given a source of uniformly distributed random numbers. If U_1 and U_2 are independent samples chosen from the uniform distribution on the unit interval $(0, 1)$, then the variables defined as:

$$Z_0 = Rcos(\Theta) = \sqrt{-2ln(U_1)} \cdot cos(2\pi U_2) \tag{12}$$

$$Z_1 = Rsin(\Theta) = \sqrt{-2ln(U_1)} \cdot sin(2\pi U_2) \tag{13}$$

are independent random variables with a standard normal distribution.

After having executed the Box–Muller algorithm we have found the following disadvantages.

- According to the results presented in Table 7, we can see that the values of the p-test are better our LFSR model, than in the Box–Muller algorithm.
- The computational cost required to implement the algorithm is much higher.

Table 7. Evolution of the p-test values obtained, after applying the Chi Square goodness of fit test method to a set of values, for Box–Muller and for our LFSR model.

LFSR Model	Box–Muller	Pol. Degree	Pol. Degree Ext Field	Number of Values
0.951123	0.87231	4	4	65,535
0.90900	0.761121	5	4	1,048,575
0.90101	0.755181	6	5	1,073,741,823
0.90012	0.699876	6	6	68,719,476,735

Another way to improve the accuracy of the Gaussian distribution is to modify the way the m-bit strings are generated. If all the states of the LFSR are used, each cell is used in the generation of n consecutive pseudo-random numbers. Although the fit tests reveal very good results (see Section 4.2), decreasing or removing the amount of numbers affected by the same cell would help to improve the accuracy. Therefore, we propose, as an alternative, to use one out of every m states. More formally, we propose to decimate by m the LFSR output. In this way, the period would be $(2^{mn}-1)/gcd((2^{mn}-1),m)$ giving rise to select m,n such that $gcd((2^{mn}-1),m)=1$ in order to reach the same period $2^{mn}-1$.

In any case, it is important to note that the period is much greater than the range, i.e., $2^{mn}-1 \gg 2^{m+log_2(n)}$, giving rise to a probability about 0.005 of generating the same number in $10,000$ values generated.

5. Conclusions

A new Gaussian PRNG has been proposed in this paper. It is based on a unique LFSR, using the same approach than the previous proposals [8–10], in order to generate a certain number of sequences of uniformly distributed numbers, needed to apply the CLT. Unlike the previous proposals, no explicit permutations or rotations have been applied to the successive LFSR states. Instead, the PRNG is operated in $GF(2^n)$ to take advantage of the relationship between the states and sequences generated in $GF(2^n)$ and $GF(2)$, that allows to represent the $m-$sequences in $GF(2^n)$ as primitive interleaved sequences composed by $m-$sequences in $GF(2)$.

The PRNG, presented in Section 3.2, allows to configure it by means of two main parameters, m (the number of cells in the LFSR) and n (the dimension of $GF(2^n)$), determining the period as $2^{m \cdot n}-1$, and the range $2^{m+log_2(n)}$. The statistical analysis reveals an excellent behaviour when the fit tests are applied.

Finally, this PRNG is a way to keep using LFSR in cryptographic applications where a uniform distribution is not required. Furthermore, as in other applications, the use of LFSRs in $GF(2^n)$ is motivated by the speed increase that is achieved by generating n bits instead of 1 in each iteration; in this case, the use of the LFSR in $GF(2^n)$ pursues a second objective: to take advantage of the implicit relationship with n different binary sequences looking for an easier implementation of the CLT. Therefore, the final rate of generating numbers with Gaussian distribution is the same as the rate at which a binary LFSR generates a bit. However, taking into account that the generated numbers are $m+log_2(n)$ bits long, the generation speed in bits per second is $m+log_2(n)$ times higher.

Author Contributions: Writing of original draft and writing—review and editing, G.C., A.P. and A.O. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by Ministerio de Economia Industria y Competitividad (MINECO), Agencia Estatal de Investigacion (AEI) and FEDER-UE under projects TIN2017-84844-C2-1-R and PGC2018-098813-B-C32.

Institutional Review Board Statement: Not Applicable.

Informed Consent Statement: Not Applicable.

Data Availability Statement: Not Applicable.

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

References

1. Padgette, J.; Bahr, J.; Batra, M.; Holtmann, M.; Smithbey, R.; Lily, C.; Scarfone, K. *Guide to Bluetooth Security*; NIST: Gaithersburg, MD, USA, 2017.
2. Jindal, P.; Singh, B. RC4 Encryption-A Literature Survey. *Procedia Comput. Sci.* **2015**, *46*, 697–705. [CrossRef]
3. Biham, E.; Dunkelman, O. Cryptanalysis of the A5/1 GSM stream cipher. In Proceedings of the International Conference on Cryptology in India, Calcutta, India, 10–13 December 2000; pp. 43–51.

4. ETSI/SAGE. *Specification of the 3GPP, Confidentiality and Integrity algorithm UEA2 and UIA2*; Document 2: SNOW 3G Specification; ETSI: Sophia Antipolis, France, 2006.
5. Finkenzeller, K. *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification*, 2nd ed.; John Wiley and Sons: Hoboken, NJ, USA, 2003.
6. Peinado, A.; Munilla, J.; Fúster-Sabater, A. Generation of pseudorandom binary sequences by means of linear feedback shift registers (LFSRs) with dynamic feedback. *Math. Comput. Model* **2013**, *57*, 2596–2604. [CrossRef]
7. Golomb, S.W. *Shift Register Sequences*, 3rd revised ed.; Aegean Park Press: Laguna Hills, CA, USA, 2017; ISBN 978-0-89412-048-0.
8. Kang, M. FPGA implementation of Gaussian-distributed pseudorandom number generator. In Proceedings of the 6th International Conference on Digital Content, Multimedia Technology and its Applications, Seoul, Korea, 16–18 August 2010; pp. 11–13.
9. Condo, C.; Gross, W.J. Pseudo-random Gaussian distribution through optimised LFSR permutations. *Electron. Lett.* **2015**, *51*, 2098–2100. [CrossRef]
10. Cotrina, G.; Peinado, A.; Ortiz, A. Gaussian Pseudorandom Number Generator Based on Cyclic Rotations of Linear Feedback Shift Registers. *Sensors* **2020**, *20*, 2103. [CrossRef] [PubMed]
11. Thomas, D.B.; Luk, W.; Leong, P.H.; Villasenor, J.D. Gaussian random number generators. *ACM Comput. Surv.* **2007**, *39*, 11-es. [CrossRef]
12. Thomas, D. FPGA gaussian random number generators with guaranteed statistical accuracy. In Proceedings of the 2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines, Boston, MA, USA, 11–13 May 2014; pp. 149–156.
13. Jouguet, P.; Elkouss, D.; Kunz-Jacques, S. High-bit-rate continuous-variable quantum key distribution. *Phys. Rev. A* **2014**, *90*, 042329. [CrossRef]
14. Gehring, T.; Händchen, V.; Duhme, J.; Furrer, F.; Franz, T.; Pacher, C.; Werner, R.F.; Schnabel, R. Implementation of continuous-variable quantum key distribution with composable and one-sided-device-independent security against coherent attacks. *Nat. Commun.* **2015**, *6*, 8795. [CrossRef]
15. Travagnin, M.; Lewis, A.M. *Quantum Key Distribution In-Field Implementations: Technology Assessment of QKD Deployments*; EUR 29865 EN; Publications Office of the European Union: Luxembourg, 2019; ISBN 978-92-76-11444-4. [CrossRef]
16. Bennett, C.H.; Brassard, G. Quantumcryptography: Public key distribution and coin tossing. In Proceedings of the IEEE International Conference on Computers, Systems and Signal Processing, New York, NY, USA, 8–11 October 1984.
17. Hu, Y.; Wu, Y.; Chen, Y.; Wan, G.C.; Tong, M.S. Gaussian Random Number Generator: Implemented in FPGA for Quantum Key Distribution. *Int. J. Numer. Model. Electron. Netw. Devices Fields* **2019**, *32*, e2554. [CrossRef]
18. Laudenbach, F.; Pacher, C.; Fung, C.F. Continuous-Variable Quantum Key Distribution with Gaussian Modulation—The Theory of Practical Implementations. *Adv. Quantum Technol.* **2018**, *1*, 1800011. [CrossRef]
19. Diamanti, E.; Leverrier, A. Distributing Secret Keys with Quantum Continuous Variables: Principle, Security and Implementations. *Entropy* **2015**, *17*, 6072–6092. [CrossRef]
20. Bai, D.; Huang, P.; Zhu, Y.; Ma, H.; Xiao, T.; Wang, T.; Zeng, G. Unidimensional continuous-variable measurement device-independent quantum key distribution. *Quantum Inf. Process.* **2020**, *19*, 53. [CrossRef]
21. Weisstein, E.W. Central Limit Theorem. From MathWorld—A Wolfram Web Resource. Available online: http://mathworld.wolfram.com/CentralLimitTheorem.html (accessed on 15 January 2021).
22. Stephens, M.A. EDF Statistics for Goodness of Fit and Some Comparisons. *J. Am. Stat. Assoc.* **1974**, *69*, 730–737. [CrossRef]
23. Malik, J.; Malik, J.; Hemani, A.; Gohar, N. An efficient hardware implementation of high quality AWGN generator using Box-Muller method. In Proceedings of the 2011 11th International Symposium on Communications & Information Technologies (ISCIT), Hangzhou, China, 12–14 October 2011; pp. 449–454.
24. Wang, Y.; Bie, Z. A novel hardware Gaussian noise generator using Box-Muller and CORDIC. In Proceedings of the 2014 Sixth International Conference on Wireless Communications and Signal Processing (WCSP), Hefei, China, 23–25 October 2014; pp 1–6.
25. Menezes, A.J.; van Oorschot, P.C.; Vanstone, S.A. *Handbook of Applied Crytography*; CRC Press: Boca Raton, FL, USA; Massachusetts Institute of Technology: Cambridge, MA, USA, 2001.
26. Park, W.J.; Komo, J.J. Relationships Between m-Sequences over $GF(q)$ and $GF(q^m)$. *IEEE Trans. Inf Theory* **1989**, *35*, 183–186. [CrossRef]
27. Gong, G. Theory and Applications of q-ary Interleaved Sequences. *IEEE Trans. Inform. Theory* **1995**, *41*, 400–411. [CrossRef]
28. Knuth, D. *The Art of Computer Programming*; Addison-Wesley: Boston, MA, USA, 1998.