Journal of Engineering Technology and Applied Sciences, 2021 *e-ISSN:* 2548-0391
Received 10 August 2020, Accepted 28 March 2021 *Vol. 6, No. 1, 45-68*
Published online 30 April 2021, Research Article *doi: 10.30931/jetas.778792*

Citation: Naim, M., Ali-Pacha, H., Ali-Pacha, A., Hadj-Said, N., "Lengthening the Period of a Linear Feedback Shift Register". Journal of Engineering Technology and Applied Sciences 6 (1) 2021 : 45-68.

LENGTHENING THE PERIOD OF A LINEAR FEEDBACK SHIFT REGISTER

Mohammed Naim, Hana Ali-Pacha, Adda Ali-Pacha*, Naima Hadj-Said

Laboratory of Coding and Security of Information
University of Sciences and Technology of Oran Mohamed Boudiaf, Algeria USTO,
PoBox 1505 El M'Naouer Oran 31000 Algeria
mkn_1993@hotmail.com, hana.alipacha@univ-usto.dz,
*a.alipacha@gmail.com (*corresponding author), naima.hadjsaid@univ-usto.dz*

Abstract

A linear feedback shift register (LFSR) is the basic element of the pseudo-random generators used to generate a sequence of pseudo-random values for a stream cipher. It consists of several cells; each cell is a flip-flop and a feedback function. The feedback function is a linear polynomial function; this function has a degree equal to the number of cells in the register. The basic elements of the register are connected to each other in two different ways, either in Fibonacci mode or in Galois mode.

In the best case, the length of an LFSR is equal to two to the power of the number of cells of this register minus one, which is very low for cryptographic applications. To increase this length, one must look for primitive polynomials of great degree or to use adequate methods to lengthen LFSR with a reduced number of cells and, this is the objective of this work. Our method of lengthening of period of a LFSR is based on the logistics map.

Keywords: LFSR, cryptography, stream cipher, pseudo-random generation, Fibonacci mode, Galois mode, logistic map

1. Introduction

The stream cipher is becoming increasingly important in our daily lives [1] [2]. Some of these systems use Linear Feedback Shift Register (LFSR) to produce a random sequence, which must be XORed with plaintext massages to give encrypted messages. LFSR is an electronic device (flip-flops connected in series) [3] that can be seen as software that produces a sequence of bits that can be seen as a recurring sequence on the Galois filed F_2 of 2 elements (0 and 1).

An LFSR is characterized by its feedback function, which connects its cells to each other in two different modes: Fibonacci and Galois. The most natural mode is the so-called Fibonacci mode. This is called because the Fibonacci sequence is represented in this model. It updates the first cell of the register, which is on the left then operates by shift for the other cells. In 2002, Goresky and Klapper introduced a completely different mode called the Galois mode [4]. The basic idea of the Galois mode is that the contents of the output cell are re-injected into the input cell and added to the contents of the other cells of the register, and then all the cells are shifted to the output.

The critical difference between the two modes is how the feedback polynomial is interpreted. For a given feedback polynomial, the two modes produce different output sequences. Another difference is their implementation, if you implement an LFSR in a CPU or an FPGA [5], the Galois structure that is more computerized and its cells are updated simultaneously is probably faster and has less latency than Fibonacci mode.

In the best case, the length of an LFSR is equal to two to the power of the number of cells of this register minus one, which is very low for cryptographic applications. To increase this length, one must look for primitive polynomials of great degree or to use adequate methods to lengthen LFSR with a reduced number of cells and, this is the objective of this work.

Our method of lengthening the period of an LFSR is essentially based on the initiation of the flip-flops of the LFSR register when the cycle of a counter is reached. In parallel with the LFSR generator activity, a non-linear discrete dynamic system is activated. When the value of the counter is reached, the value of the discrete non-linear dynamic system which corresponds to it, it is used to initialize the flip-flops of the LFSR register.

We know, by definition, that the output of such a non-linear dynamic system can seem almost random and, this is what lengthens the period of our LFSR, and that we can use this assembly in cryptographic applications.

In this paper and for the reason of simplicity, we have chosen the logistics map as a non-linear dynamic system because the system is chaotic for a simple deterministic equation but, with a parameter chosen judiciously.

2. Linear feedback shift register

A linear feedback shift register is an electronic device that produces a sequence of random bits [6]. It consists of several cells each cell presented by a flip-flop, the particularity of the connection between cells called feedback function, looks like a linear function, so we can apply a mathematical structure. The LFSRs (Figure 1) have the following characteristics:

- ✓ The output is the contents of the last cell on the right.
- ✓ The period is the sequence length produced before it begins to repeat itself.
- ✓ The feedback function is the sum of "exclusive" operations of some of the bits of the register, the list of these bits is called "the derivation sequence". It is obtained from a chosen polynomial.
- ✓ They are easy to make in hardware [7].

2.1 How an LFSR works

LFSRs are used as pseudo-random number generators [3] [6]. When properly configured, they reach periods of maximum length; each state will be reached only once until all states are reached. Once each state has been reached, the period will be repeated [7].

In general, LFSRs are built with D flip-flops and, XOR operations. The initial value of the shift register and the feedback function determines the order of output [6].

An LFSR of length L is composed of a shift register containing a sequence of L bits (S_i... S_{i+L-1}) and a linear feedback function, as well as a clock controlling the movement data [6].
At each clock tick, the content of the rightmost cell is the output of the register and, the contents of the other cells are shifted to the right, a new bit is calculated by the feedback function and will be placed in the leftmost cell of the register:

$$S_{t+L} = C_1 S_{t+L-1} + C_2 S_{t+L-2} + \cdots + C_{L-1} S_{t+1} + C_L S_t \qquad (1)$$

The coefficients C_i are binary.

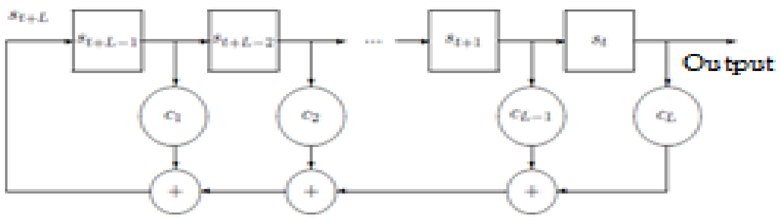

Figure 1. LFSR of length L.

Definition 2.1 ([4]) An LFSR whose feedback function is given by the relation:

$$S_{t+L} = C_1 S_{t+L-1} + C_2 S_{t+L-2} + \cdots + C_{L-1} S_{t+1} + C_L S_t \qquad (2)$$

Its feedback polynomial f is the following $F_2[X]$ polynomial:

$$f(x) = 1 + C_1 x + C_2 x^2 + \cdots + C_{L-1} x^{L-1} + C_L x^L \qquad (3)$$

According to [4]: “The sequence $(S_n)_{n \geq n_0}$ is produced by an LFSR whose feedback polynomial is f (x) If and only if it’s formal serial development:

$$S(x) = \sum_{n \geq 0} S_n x^n \qquad (4)$$

Is written:

$$S(x) = \frac{g(x)}{f(x)} \qquad (5)$$

Where $\boldsymbol{g}$ is a polynomial of F_2[X] such that deg (g) < deg (f), and gcd (g_0,f_0) =1. In addition, the polynomial $\boldsymbol{g}$ is determined by the initial state of the register:

$$g(x) = \sum_{i=0}^{L-1} x^i \sum_{j=0}^{i} C_{i-j} S_j \qquad (6)$$

It may be noted that such a sequence is ultimately periodic, that is to say, that there exists a pre-period n_0 such that the sequence $((S_n)_{n\geq n_0}$ is periodic of period T $\leq 2^L - 1$ (there exists an integer i_0 such that $S_i = S_i$+T for all i ≥ i_0)".

The LFSRs have been studied since 1930 in their purely theoretical aspect and are mostly built on a finite field. From 1948 to 1969, LFSRs are used as generators of pseudo-random sequences in cryptosystems since they can generate binary sequences of the maximum period. These sequences are called the m-sequences (maximum length sequences). The search for sequences for a very long period becomes a crucial problem in the 1950s.

To ensure better security, we must respect three characteristics, in addition to having a maximum period, the m-sequences must verify all the random postulates that give them good random quality. A pseudo-random generator used by cryptography must be able to [1]:

- Generate sequences of bits that must satisfy the statistical characteristics of truly random sequences.
- To guarantee that if an attacker knows all, or part of the sequence encrypting $S_0, S_1, \ldots, S_i$ it is difficult (from a computational point of view) to find the seed.

2.2 Presentation of LFSR modes: Fibonacci & Galois

2.2.1 Fibonacci mode

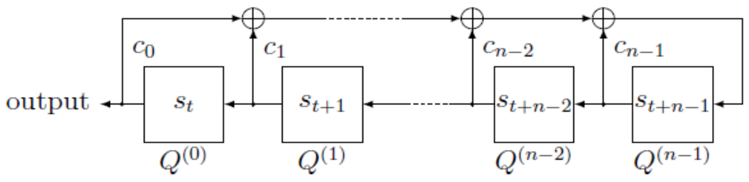

Figure 2. LFSR Fibonacci mode

The Fibonacci model LFSR or just Fibonacci LFSR, named after the 12th-century Italian mathematician Leonardo Fibonacci, is the most common model of LFSR [5] [6] used in systems that require to be generated. It is these qualities along with its lightweight hardware implementation that make it so popular in cryptographic systems. A Fibonacci LFSR consists of several memory cells that shift their values with each clock interval and a feedback function that feeds new values into the first cell, see Figure 2. The memory cells whose values are evaluated by the feedback function are known as the taps or tapped positions.
The output sequence S of the LFSR of Figure 2 satisfies the linear recurrence:

$$S_{t+n} = C_{n-1}S_{t+n-1} + C_{n-2}S_{t+n-2} + +C_1S_{t+1} + C_0S_t \tag{7}$$

The feedback polynomial of the LFSR is equivalent to the inverse of the characteristic polynomial of the linear recurrence sequence above:

$$f(x) = 1 + C_{n-1}x + \cdots + C_1x^{n-1} + C_0x^n \tag{8}$$

A Fibonacci LFSR of length n [6], with a feedback polynomial f (x) that updates at each clock interval of t in equation (9).

$$Q_{t+1}{}^{(i)} = \begin{cases} C_{n-1}Q_t{}^{(n-1)} + +C_1Q_t{}^{(1)} + C_0Q_t{}^{(0)} \dots \ if\ i = n-1 \\ Q_t{}^{(i-1)}\ Else \end{cases} \tag{9}$$

An example of a Fibonacci mode LFSR with a feedback polynomial [8] $f(x) = \boldsymbol{X^8 + X^6 + X^5 + X^3 + 1}$, is shown in Figure 2.1.

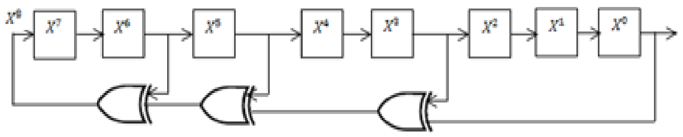

Figure 2.1. LFSR Fibonacci $X^8 + X^6 + X^5 + X^3 + 1$

LFSR Fibonacci is the most common model of LFSR used in systems that require the generation of a pseudo-random sequence. It is these qualities with its simple hardware implementation that make it very popular in cryptographic systems.

2.2.2 Galois mode

The Galois model, named after the 19th-century French mathematician Evariste Galois [4] [6]. The particularity of an LFSR in Galois mode resides in how the feedback polynomial is interpreted as well as, it uses the XOR operator between the cells of the register (**modular internal XORs**): the output bit is XORed with the contains of a cell and the result is stored in the next cell.

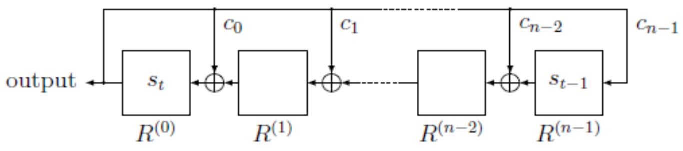

Figure 3. LSFR Galois Mode

In the Galois configuration when the output bit is zero (the input bit becomes zero) and all the bits in the register shift to the right unchanged. When the output bit is one (the input bit becomes 1), we reverse the contents of a cell (if they are 0, they become 1, and if they are 1, they become 0) before storing it in the next cell.

The feedback polynomial of the LFSR Galois is defined:

$$f(x) = x^n + C_{n-1}x^{n-1} + \cdots + C_1x + C_0, \tag{10}$$

A Galois LFSR of length n [4], with feedback polynomial ***f(x)*** that updates at each clock interval of ***t*** in equation 11.

$$R_{t+1}{}^{(i)} = \begin{cases} C_{n-1}R_t{}^{(0)}, & if\ i = n-1 \\ R_t{}^{(i+1)} + C_iR_t{}^{(0)} & else \end{cases} \tag{11}$$

An example of a Galois mode LFSR with a feedback polynomial [8] $f(x) = \boldsymbol{X^8 + X^6 + X^5 + X^3 + 1}$, is shown in Figure 3.1.

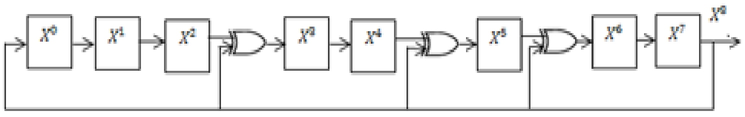

Figure 3.1. LFSR Galois $X^8 + X^6 + X^5 + X^3 + 1$

3. Logistic map

Logistics map [9] [10] is a well-known dynamic in non-linear systems theory, defined by equation (12):

$$yk + 1 = \mu\, xk\,(1 - xk) \qquad (12)$$

It gives a perfect explanation of dynamic system behavior. This system was developed by Prof. Pierre François Verhulst (1845) to measure the evolution of a population in a limited environment, later used in 1976 by the biologist Robert May to study the evolution of insect population:

- $yk + 1$: Generation in the future that is proportional to x_k.
- xk: Previous generation.
- μ: Positive constant incorporates all factors related to reproductive, successful overwintering eggs for example, etc.

To study this dynamic system and some asymptotic individuals' models, the first thing to do is to draw the parabolic graph $y= \mu.x\,(1\text{-}x)$, and the diagonal $y=x$.

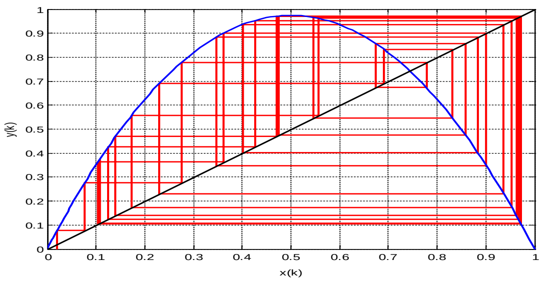

Figure 4a. Evolution of yk in the function of xk

The operation that we will follow to draw the iterative form $yk + 1$ according to x_k is simply summarized as follows:

- Starting from an initial value $x0$ of the x-axis, we reach the function with a vertical; the function takes the value $y1 = \mu.x0\,(1 - x0)$,
- From horizontal $y1 = r.x0\,(1 - x0)$ of the previous point, we join the line $y = x$;
- We represent the abscissa of the intersection with the vertical line $x = x0$; we have $y1 = x1$

- From the *x₁* value of the x-axis, we reach the function with a vertical; the function takes the value $y2 = \mu.x1\ (1 - x1)$; and so on.

We take $\mu = 3.9$ and, $x0 = 0.01$ for the logistics map, the previous operations for 100 iterations are represented in Figure 4.
Figure 4 shows two signals yk generated from the logistic map in chaotic mode ($\mu = 3.9$), one with an initial condition $x0 = 0.1$ and the other with $x0 = 0.100000000000001$ very close to 0.1.

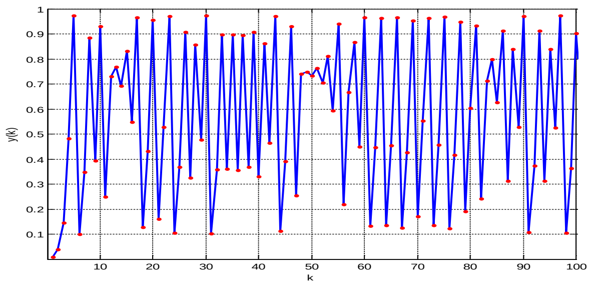

Figure 4b. The chaotic regime in the function of k

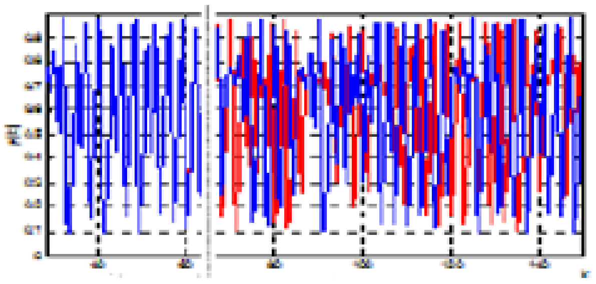

Figure 4c. Sensitivity to initial conditions

We note that a very small error on the knowledge of the initial state x_0 in the phase space will be rapidly amplified and gives us two widely different signals. Quantitatively, the growth of error is locally exponential for highly chaotic systems (sensitivity to initial conditions).
It should be noted that the initial condition error in this case is 10^{-15} and this is the smallest value because Matlab works with only 52 bits but the system can be sensitive to smaller values than 10^{-15} depending on the work environment.

4. Proposed system: Lengthening of the period

In the best case, the length of an LFSR is equal to two to the power of the number of cells of this register minus one (L= 2^N-1, N number of cells of an LFSR). To increase this length, one proposes to use the logistics map for our designing schemas figures 5a and 5b.

We will take the following values as initial conditions of the logistic map (eq.12), and this is valid throughout the paper:

1. $X0 = 0.1, \mu = 3.9999, F = 10^7$.

We multiply the result of the logistic sequence with F and take the integer part (by the function floor) and we do the modulo 2 which will give a result equal to either 1 or 0.

$T_n = Mod\ (F * \mu X_{n-1}(1 - X_{n-1}), 2)$
(13)

In addition, we will take the following assumptions:

2. The feedback function [8]: is : $\boldsymbol{f(x) = X^8 + X^6 + X^5 + X^3 + 1}$
3. The seed of the cells of the register is 10101010.
4. Our data to be encrypted are images of dimension (256x256 = 65,536) pixels, we take the image "cameraman" for the validation of our system.

Our designing schemes are following:

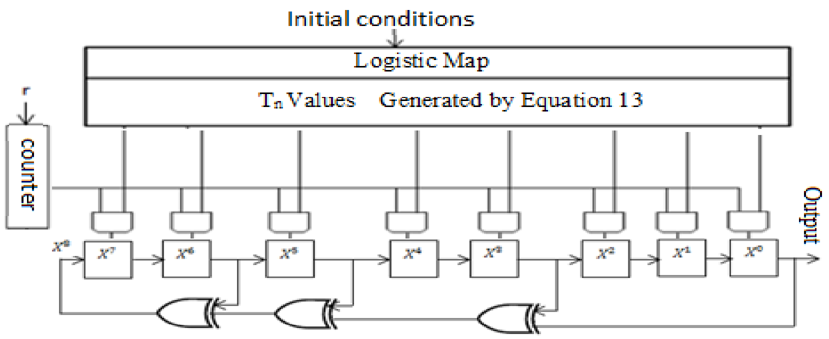

Figure 5a. LFSR Fibonacci $X^8 + X^6 + X^5 + X^3 + 1$

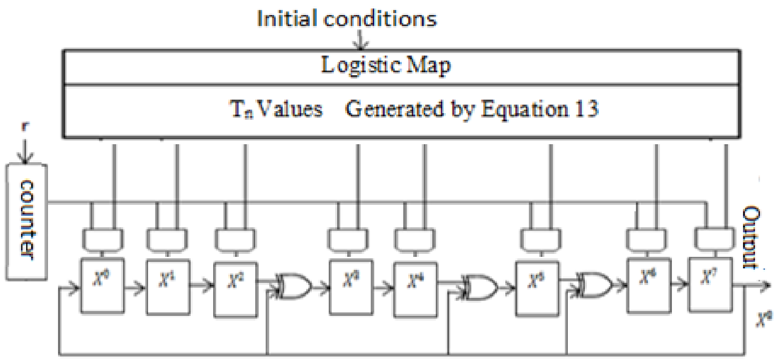

Figure 5b. LFSR Galois $X^8 + X^6 + X^5 + X^3 + 1$

After initial conditions (logistic and seed of LFSR) judiciously chosen, there is a discretization in modulo 256 of the real values of the logistic map. This discretization is an 8-bit binary value, where each bit is linked to an LFSR cell. We will allow that the internal state of the LFSR will take the discrete value of the logistic map each time there is an initialization of a counter that counts to a chosen value r.

5. Results and interpretations

Determining whether a generator is random or not is a tricky problem. Indeed there is no universal test that can say with certainty that a generator is random. The principle is to show that this generator is not biased by studying the properties of the numbers it generates. In practice, a random generator produces a sequence of numbers with properties of unpredictability and independence, and follows a certain distribution (uniform in cryptography, Gaussian in telecommunications, etc.). The evaluation of the random quality of a generator thus passes through the control of the properties of the sequence that it generates. This is achieved through statistical tests that compare the performance of the generator studied compared to theoretical ones.

We will present some tests used to evaluate the performance of our generator such as entropy test, average test or spectral test.

A. Characteristics of the Working Computer

The application was created from a PC HP pavilion 15 Notebook:
- ✓ Memory : 4096MB RAM
- ✓ Processor : Intel® Core™ i3-3120M CPU @ 2.50GHZ
- ✓ Operating System : Windows 7 Ultimate 32-bit Edition
- ✓ Graphics Card: Intel® HD Graphics 4000
- ✓ Total memory ≈ 2734 MB

For the implementation of our application, we used the C/C++ programming language.

B. Tests of Different Generator with Different Modes

B.1. Mean, Variance and autocorrelation factor Tests

We must test the distribution of the numbers produced in the sequence in its interval of operation, we have calculated the three operators of statistic test: the mean, the variance and the autocorrelation function of these numbers. In the ideal case [11], and for a random variable u which follows a uniform distribution over an interval [0; 1], the following three values must be found:

- **Mean of the numbers**
  $\bar{u} = \frac{1}{n}\sum_{i=1}^{n} u_i = \frac{1}{2} = 0.5$
  (14a)
- **The variance of the numbers**
  $v = \frac{1}{n}\sum_{i=1}^{n} u_i^2 - \bar{u}^2 = \frac{1}{12} = 0.0833$
  (14b)
- **Autocorrelation Factor**
  $E(u_i u_{i+1}) = \frac{1}{n}\sum_{i=1}^{n=1} u_i\, u_{i+1} = \frac{1}{4} = 0.25$
  (14c)

In our case, one pose: $\boldsymbol{u_i = \frac{x_i}{m}}$, $\forall$ i = 1... n

m: The largest value or the value of the modulo.

B.2 Frequency Test: Repetition of Number

The most natural test is that of the frequencies of occurrence of each digit, for a real random sequence, a particular number has no reason to be more or less represented than another. In other words, the frequencies of each digit must eventually come close to 10%.

B.3 Entropy Test

Shannon entropy is a mathematical function that intuitively corresponds to the amount of information contained or delivered by a source of information. For a source, which is a discrete random variable X comprising n symbols, each symbol x_i having a probability P_i to appear, the entropy H of the source X is defined as:

$H(x) = -\sum_{i=1}^{n} P_i . log_2 \, (P_i)$
(15)

We pose $P_i = \frac{k_i}{n}$
(16)

With i varying from 0 to 255, and n and the number of values generated in our case (n = 256 * 256 = 65536), ki corresponds to the occurrence frequency of each number i. A logarithm based on 2 is usually used because the entropy then has the bit/symbol units. On the other hand, consider a source that has an alphabet of 256 characters. If all these characters are equiprobable, the entropy associated with each character is **$\log_2(256)= \log_2(2^8)=8$ bits**, which means that it takes 8 bits to transmit a character thus, its entropy is equal to **8 bits**.

B.4. Spectral analysis

Knuth describes [11] the spectral test as the most discriminating of all. Indeed, no proven bad generator could succeed. Very simple, the method consists of studying the distribution of the values generated in a dimension k (2D or 3D) to check the quality. All generators suffer from a Marsaglia effect (this is because we do not generate all the real numbers, but only fractions are generated). In general, the spectral test makes it possible to determine the deviation d between two lines. At the most, this gap is small at most the generator is of good quality.

Dimension 2 (2D): Two consecutive values will be the coordinates of a point on the plane. One looks if the points are uniformly distributed in a square.

Dimension 3 (3D): Three consecutive values will be the coordinates of a point in space. One looks if; the points are distributed evenly in a cube. By turning the cube, one sees the undesirable effect: plans of Marsaglia.

5.1 Status of flip-flops of the register

For a seed =10101010, the LFSR with the polynomial $\boldsymbol{f(x) = X^8 + X^6 + X^5 + X^3 + 1}$ which is a primitive polynomial of degree 8 [8], has a maximum period equal to $\mathbf{255}$ $(= \mathbf{2^8 - 1})$ in both modes: Fibonacci and Galois. To read the states of the LFSR flip-flops, read the values line by line. We start with the first line and, from left to right, then and the second line always from left to right, and so on until the end of the last value in the last line of the matrix, we return to the first value of the first line.

Table 1a. Status of flip-flops register in Fibonacci mode

$$\begin{pmatrix}
170-85-42-21-138-197-98-49-24-140-198-227-241-248-252-254 \\
255-127-63-159-79-167-83-41-148-74-37-18-9-4-2-1 \\
128-64-160-208-232-244-122-189-222-111-55-27-13-6-3-129 \\
192-224-112-56-28-142-199-99-177-88-44-22-11-5-130-65 \\
32-144-72-36-146-73-164-210-233-116-58-29-14-135-195-97 \\
176-216-108-182-219-237-118-59-157-78-39-19-137-68-162-209 \\
104-180-218-109-54-155-77-166-211-105-52-154-205-230-115-185 \\
220-110-183-91-173-214-235-117-186-93-174-87-43-149-202-101 \\
178-217-236-246-123-61-158-207-231-243-249-124-190-95-175-215 \\
107-53-26-141-70-163-81-40-20-10-133-194-225-240-120-188 \\
94-47-151-203-229-242-121-60-30-143-71-35-17-136-196-226 \\
113-184-92-46-23-139-69-34-145-200-100-50-153-76-38-147 \\
201-228-114-57-156-206-103-179-89-172-86-171-213-106-181-90 \\
45-150-75-165-82-169-212-234-245-250-253-126-191-223-239-119 \\
187-221-238-247-251-125-62-31-15-7-131-193-96-48-152-204 \\
102-51-25-12-134-67-33-16-8-132-66-161-80-168-84
\end{pmatrix}$$

Table 1b. Status of flip-flops register in Galois mode

$$\begin{pmatrix}
170-85-188-94-47-129-214-107-163-199-245-236-118-59-139-211 \\
255-233-226-113-174-87-189-200-100-50-25-154-77-176-88-44 \\
22-11-147-223-249-234-117-172-86-43-131-215-253-232-116-58 \\
29-152-76-38-19-159-217-250-125-168-84-42-21-156-78-39 \\
133-212-106-53-140-70-35-135-213-252-126-63-137-210-105-162 \\
81-190-95-185-202-101-164-82-41-130-65-182-91-187-203-243 \\
239-225-230-115-175-193-246-123-171-195-247-237-224-112-56-28 \\
14-7-149-220-110-55-141-208-104-52-26-13-144-72-36-18 \\
9-146-73-178-89-186-93-184-92-46-23-157-216-108-54-27 \\
155-219-251-235-227-231-229-228-114-57-138-69-180-90-45-128 \\
64-32-16-8-4-2-1-150-75-179-207-241-238-119-173-192 \\
96-48-24-12-6-3-151-221-248-124-62-31-153-218-109-160 \\
80-40-20-10-5-148-74-37-132-66-33-134-67-183-205-240 \\
120-60-30-15-145-222-111-161-198-99-167-197-244-122-61-136 \\
68-34-17-158-79-177-206-103-165-196-98-49-142-71-181-204 \\
102-51-143-209-254-127-169-194-97-166-83-191-201-242-121
\end{pmatrix}$$

5.2 T_n vector of the logistic map

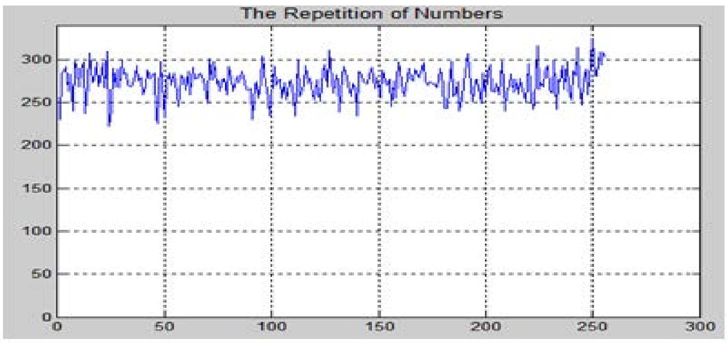

Figure 6a: Representation of T_n values

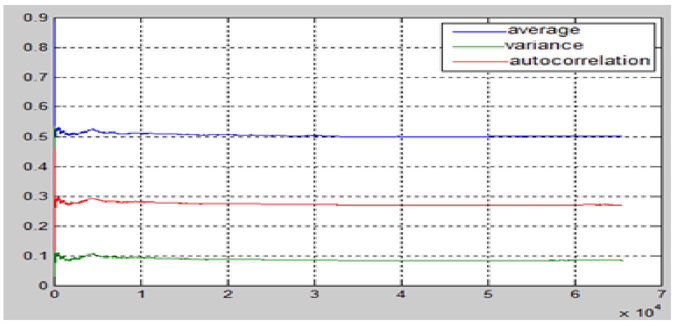

Figure 6b. statistics test of T_n values

FIG. 6 represents the first 70000 values of the logistic sequence T_n resulting from equation 13. The length of the period of T_n values are greater than 70000, and we have found an entropy equal to 7.996930 (it is a 99.96% of entropy of random sequence). The statistics test of T_n values are near a statistic test of random sequence values.

5.3 Lengthening of the period of LFSR in different modes

Table 2 shows the different lengths of the period of LFSR following different R-values of the counter for both modes.

Table 2. Period of the lengthening of the period of LFSR for different R-values of the counter

Counter	Length of the period in a mode of	
R	Fibonacci	Galois
3	>70000	>70000
5	>70000	10800
8	>70000	>70000
10	1111	2250
100	321	118
150	261	270
200	525	343
250	641	335
400	26363	12855
500	29513	22302
1000	>70000	>70000
2000	>70000	>70000
2500	>70000	>70000
20000	46649	1041

5.3.1 Lengthening of the period of LFSR in Fibonacci mode

We will present the results of the data generated by the generators of ***Figure 2.1*** and ***Figure 5a***, to see the contribution of the lengthening of the period of a register operating in Fibonacci mode, on the data generated.

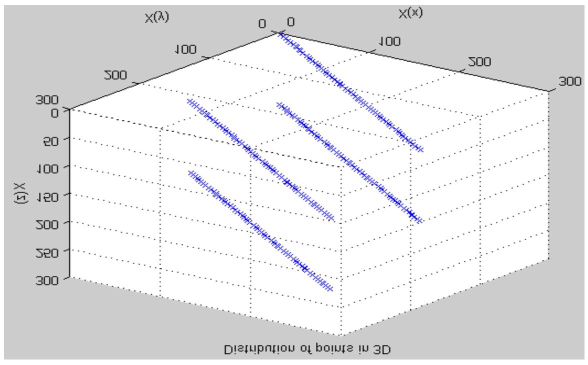

Figure 7a. Spectral analyses in 3D of figure 2.1

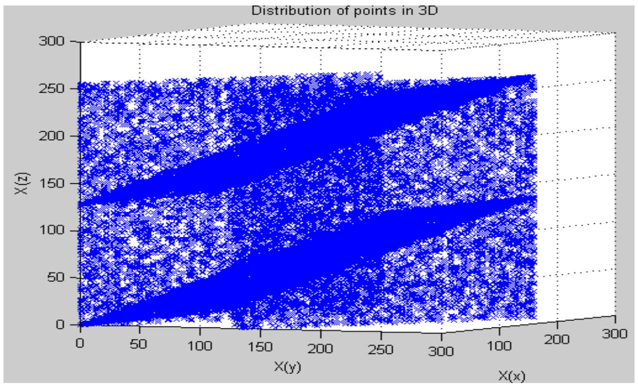

Figure 7b. Spectral analyses in 3D of figure 5

FIG 7 and FIG 8. The spectral test of the 2D and 3D of the generator schematized by figure 2.1, shows that the distribution of the values are on, the spaced lines, and are not distributed in all the surface of the plane 2D (of the sphere 3D). On the other hand, the generator was schematized by FIG. 5a shows a very good distribution of the values of the surface of the plane (of the sphere).

FIG 9. The statistics test (mean, variance, and autocorrelation) of generators schematized by figure 2.1 and by figure 5a are near a statistic test of random sequence values.

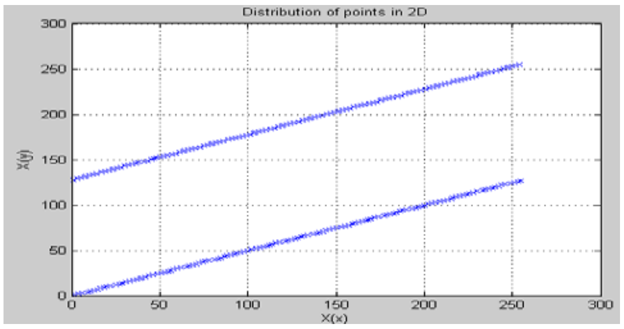

Figure 8a. Spectral analyses in 2D of figure 2.1

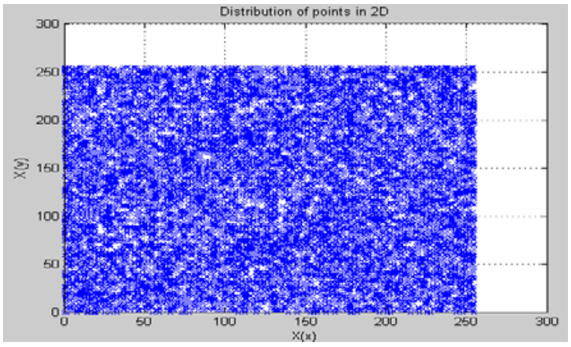

Figure 8b. Spectral analyses in 2D of figure 5a

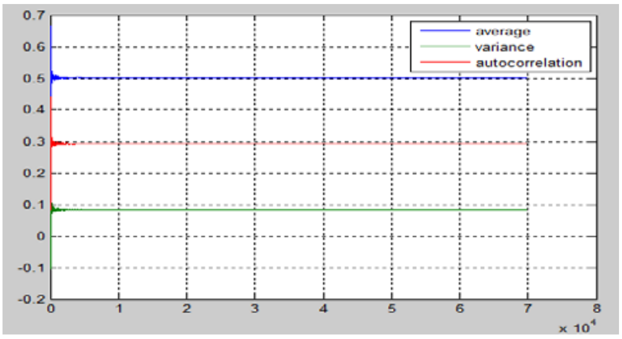

Figure 9a. Statistic test of figure 2.1

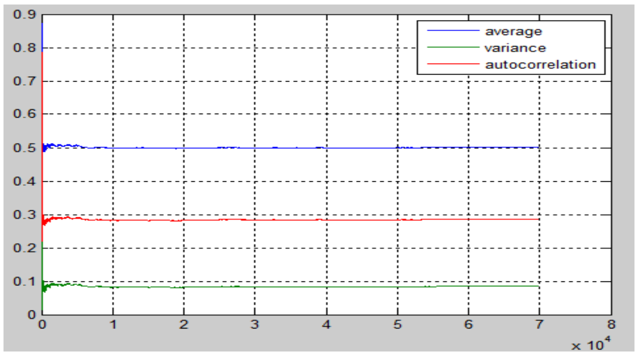

Figure 9b. Statistic test of figure 5a

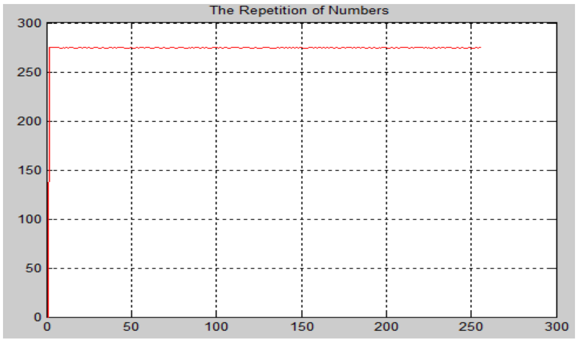

Figure 10a. Frequency test of figure 2.1

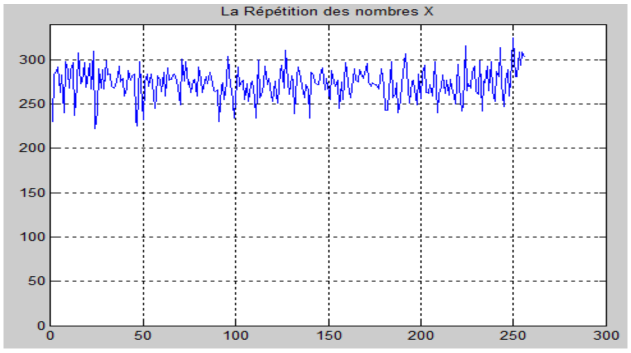

Figure 10b. Frequency test of figure 5a

The generator frequency test shown in Figure 5a shows that all the values from 0 to 255 are distributed with a normal distribution (same probability of occurrence), but those of the generator shown in Figure 2.1, follow an almost normal distribution.

Table.3 Entropy and period of LFSR in Fibonacci mode

Test	Generator based on the schema of	
	figure 2.1	*figure 5a*
Entropy	**7.994351**	**7.995870**
	99.93%	**99.95%**
	% in ideal entropy test of random sequence	
Period	**255**	**>70000**

Table 3 shows the positive contribution of the generator schematically in Figure 5a to generate random data with a very large repetition period of its data. It is appropriate to use it in encryption applications.

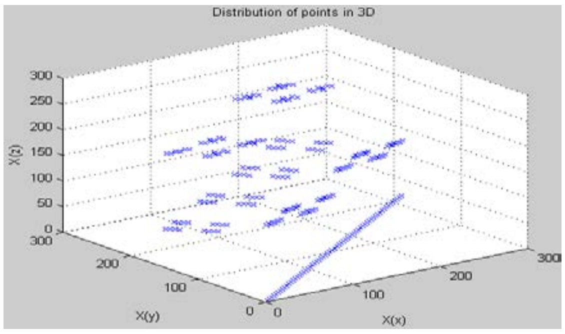

Figure 11a. Spectral analyses in 3D of figure 3.1

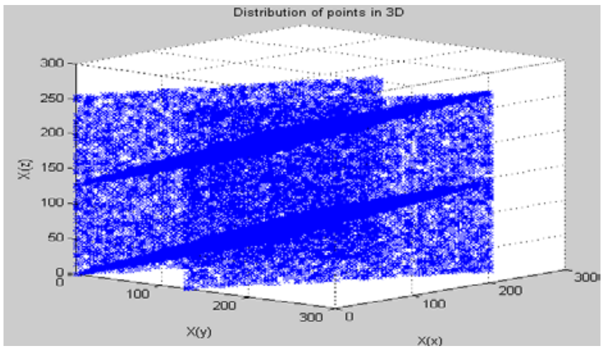

Figure 11b. Spectral analyses in 3D of figure 5b

5.3.2 Lengthening of the period of LFSR in Galois mode

We will present the results of the data generated by the generators of ***Figure 3.1*** and ***Figure 5b***, to see the contribution of the lengthening of the period of a register operating in Galois mode, on the data generated.

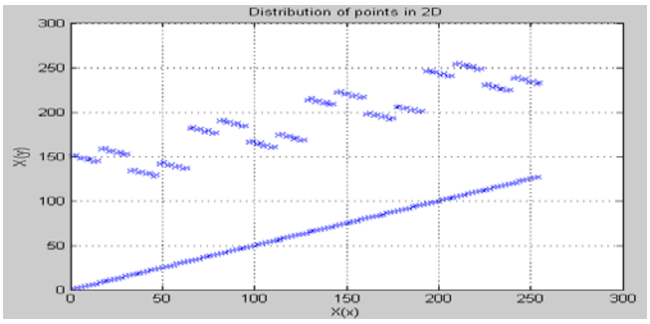

Figure12. Spectral analyses in 2D of figure 3.1

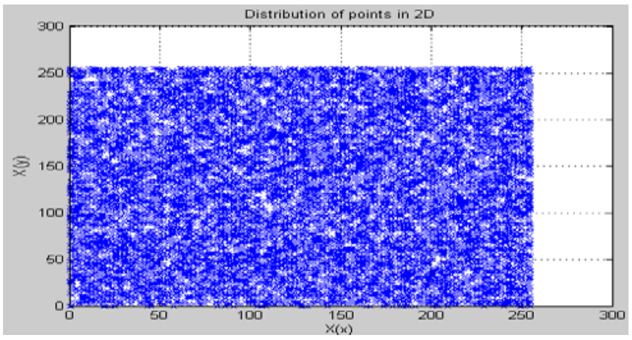

Figure 12b. Spectral analyses in 2D of figure 5b

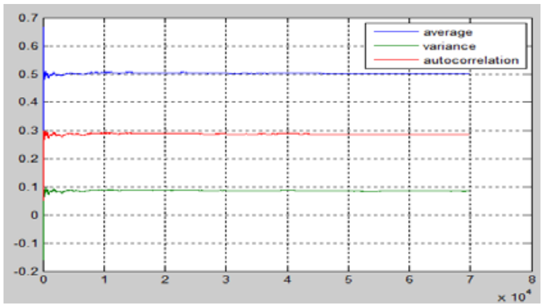

Figure 13a. Statistic test of figure 3.1

FIG 11 and FIG 12. The spectral test of the 2D and 3D of the generator schematized by figure 3.1, shows that the distribution of the values are on, the spaced lines, and are not distributed in all the surface of the plane 2D (of the sphere 3D). On the other hand, the generator was schematized by FIG. 5b, shows a very good distribution of the values of the surface of the plane (of the sphere).

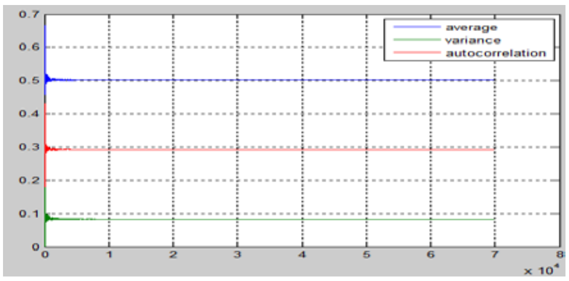

Figure 13b. Statistic test of figure 5b

FIG 13. The statistics test (mean, variance, and autocorrelation) of generators schematized by figure 3.1 and by figure 5b are near a statistic test of random sequence values.

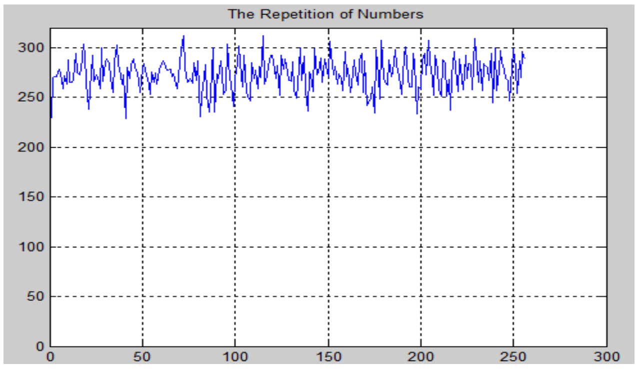

Figure 14a. Frequency test of figure 31

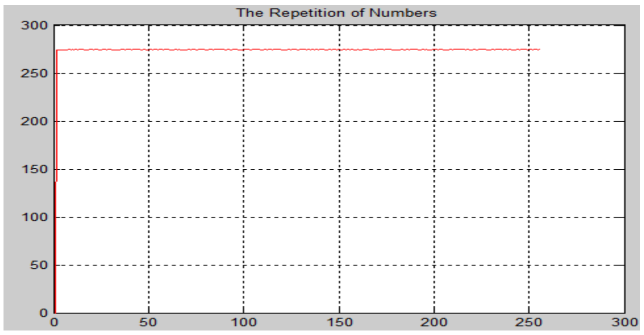

Figure 14b. Frequency test of figure 5b

The generator frequency test shown in Figure 5b, shows that all the values from 0 to 255 are distributed with a normal distribution (same probability of occurrence), but those of the generator shown in Figure 3.1, follow an almost normal distribution.

Table 4. Entropy and period of LFSR in Galois mode

Test	Generator based on the schema of	
	figure 3.1	*figure 5b*
Entropy	**7.994351**	**7.997286**
	99.93%	**99.97%**
	% in ideal entropy test of random sequence	
Period	**255**	**>70000**

Table 4 shows the positive contribution of the generator schematically in Figure 5b to generate random data with a very large repetition period of its data. It is appropriate to use it in encryption applications.

6. Stream cipher with LFSR in lengthening the period

We propose to use the stream cipher [12] which consists to produce a keystream sequence, generated an LFSR in the lengthening period (generator schematized by FIG. 5a or FIG. 5b), which will be XORed to the plaintext. Therefore, the ciphertext will be obtained.

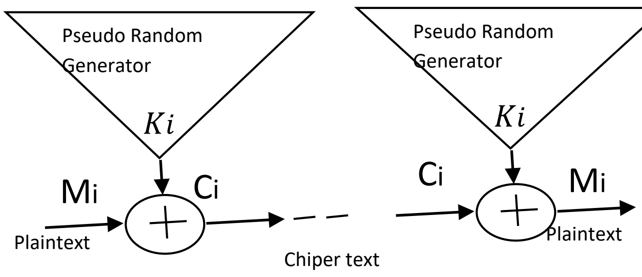

Figure 15. Stream Cipher

$\boldsymbol{mi}$: Plaintext, $\boldsymbol{ci}$: ciphertext, $\boldsymbol{ki}$: keystream generated by LFSR in lengthening period (generator schematized by FIG. 5a or FIG. 5b). This type of generator generates key streams $k1, k2, k3, \dots, ki.$ these key streams are XORed to the plaintext m_1, m_2, m_3... m_i modulo 256, to produce the ciphertext [12].

$$ci = mi + ki \ mod\ (256) \tag{17a}$$

For receptor, the ciphertext is XORed with an identical keystream, to retrieve the plaintext:

$$mi = ci - ki \, mod\ (256) \tag{17b}$$

Any continuous synchronous encryption algorithm uses keys (secrets) and generates the same keystream [12] used for encryption and decryption, this flow is generated independently of the flow of the message.

To circumvent the effect of finite precision on chaotic dynamics like the one used in [14, 15], we propose a new technique that allows not only to have a very long cycle of length but also to impose a length minimum cycle, directly dependent on the disturbing signal.

6.1. Implementation

We propose to use LFSR in the lengthening period (generator schematized by FIG. 5a or FIG. 5b) for stream cipher, to generate a data stream that will be XORed with an image. We take the following assumptions:

1. The feedback function is : $\boldsymbol{f(x) = X^8 + X^6 + X^5 + X^3 + 1}$
2. Initial conditions of the logistic sequence: the seed of the cells of the register is 10101010.
3. $X(0) = 0.1, \mu = 3.9999, F = 10^7$.
4. The image is "Cameraman" of 256 * 256 pixels.
5. Counter r=500.
6. Generator schematized by FIG. 5b.

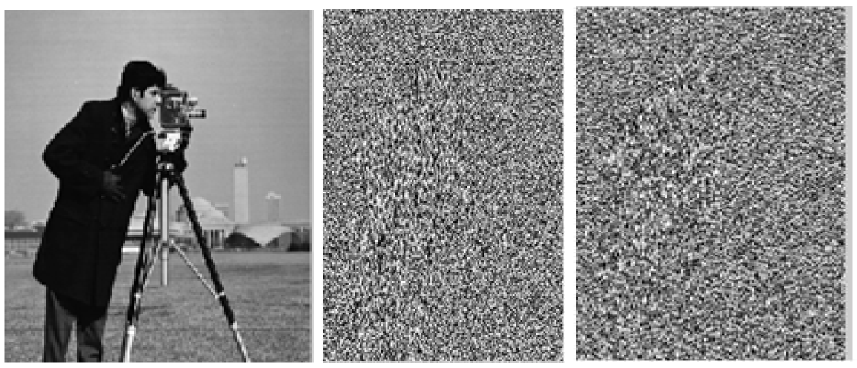
Figure 16. Plaintext and Encrypted Images

Histogram of the Images [13]**: "**For a monochrome image, tha63.t is to say with a single component, the histogram is defined as a discrete function that maps to each value intensity, the number of pixels of this value.

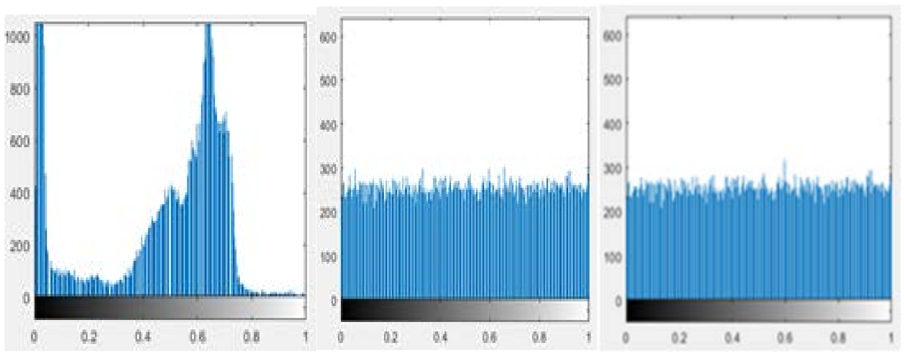
Figure 17. Histogram of plaintext and encrypted images

The determination of the histogram is carried out by counting the pixel intensity for each of the images. The histogram can then be seen as a probability density. The histograms are resistant to several transformations on the image. They are invariant to rotations and translations, and a lesser extent to changes of point of view, and changes of scale. Referring to the results obtained, we can see that the plaintext image differs substantially from the corresponding ciphered one. Moreover, the histogram of the ciphered image is uniform which makes it difficult to extract the pixels statistical nature of the plaintext image".

The histograms of the plaintext and the cipher images of the "cameraman" show that the proposed cryptosystem works correctly.

Entropy: From figures of the histogram of the encrypted image and table 5, one-note have a uniform histogram, which means that the gray levels have the same number of occurrences, and hence the entropy is the maximum.
Therefore, a grayscale image, where each pixel is represented by 8 bits, must have entropy for the encrypted image, the closest possible 8 bits/pixel.

Table 5. The entropy of the ciphered images

Camera man Image	Entropy in bits/symbols		
	plaintext	Fibonacci	Galois
Entropy	7.009716	7.997233	7.996905

The obtained value is very close to the theoretical one (99.97%). Referring to the results, we can see that the plaintext images differ significantly from her corresponding encrypted. Moreover, the histogram of the encrypted images is quite uniform which makes it difficult for the statistical extraction of pixels of the plaintext image.

The computation of the entropy of the images encrypted by the LFSR in the lengthening period (generator schematized by FIG. 5a or FIG. 5b) reveals that, the proposed crypto-system functions in a correct way.

Correlation of the Adjacent Pixels [13]**:** "In probability and in statistics, to study the correlation between two random variables or numerical statistics is to study the strength of the bond that can exist between these variables. The searched link is an affine relationship, it is the linear regression. For example, we calculate the correlation coefficient between two sets of the same length (typical case: a regression). Assume we have the following table of values: $X(x1, \dots, xn)$ and $Y(y1, \dots, yn)$ of each of the two series.

A measure of this correlation is obtained by calculating **the linear correlation coefficient of Bravais-Pearson** [13]**.**

For the correlation coefficient linking these two sets, we apply the following formula:

$$Coef(X,Y) = \frac{cov(X,Y)}{\sqrt{D(X)}.\sqrt{DY}} \tag{18a}$$

Covariance between x and y is given as follows:

$$cov(X,Y) = \frac{1}{N}\sum_{i=1}^{N}\left((X_i - E(X)).(Y_i - E(Y))\right) \tag{18b}$$

The average of X is: $E(X) = \frac{1}{N}\sum_{i=1}^{N} X_i$ (18c)

The average of Y is: $E(Y) = \frac{1}{N}\sum_{i=1}^{N} Y_i$ (18e)

The standard deviation of X is:

$$D(X) = \frac{1}{N}\sum_{i=1}^{N}(X_i - E(X))^2 \tag{18f}$$

The standard deviation of Y is:

$$D(Y) = \frac{1}{N}\sum_{i=1}^{N}(Y_i - E(Y))^2 \tag{18g}$$

The correlation coefficient is between -1 and 1. Intermediate values provide information on the degree of linear dependence between two variables. The closer the coefficient is close to extreme values -1 and 1, the closer the correlation between variables is strong we simply use the term "highly correlated" to describe the two variables. A correlation equal to 0 means that the variables are not correlated. To test the correlation coefficient, we selected randomly **1500 pairs** of two adjacent pixels in both encrypted and clear picture".

Value of the pixel at the position (x, y)

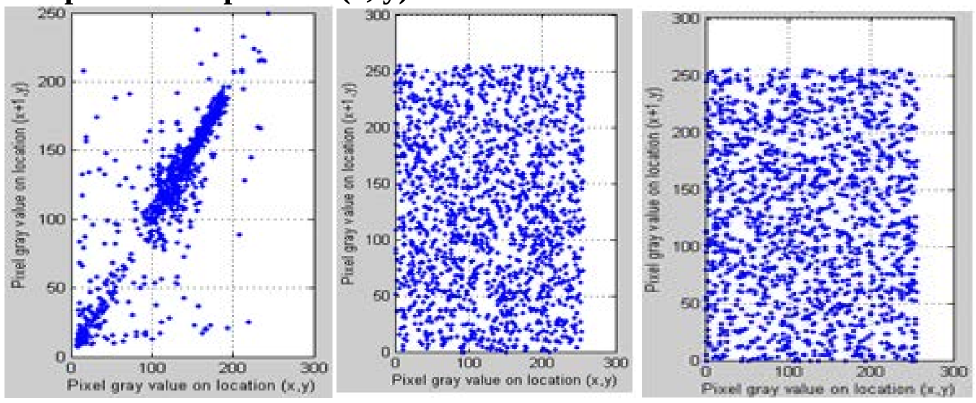

Fig.18. Correlation between the horizontally adjacent pixels of respectively the plaintext image and the encrypted images

Figure 18 and Table 6 show the correlation between two horizontally adjacent pixels of the plaintext image and encrypted (for both generators schematized by FIG. 5a or FIG. 5b) image.

Table 6. Autocorrelation coefficient factor

Cameraman Image	autocorrelation coefficient factor	
	Fibonacci	Galois
plaintext	0.95215	0.95122
Encrypted	-0.0229	-0.0208

We see that the neighboring pixels in the plaintext image have a factor correlation close to one; while the encrypted will have one little correlation close to zero. This low correlation between two neighboring pixels in the encrypted image makes the attack of our cryptosystem difficult.

In addition, it is clear that in the image clear, several lines can be adjusted to scatter but among all these lines can be retained which has a remarkable property giving rise to the right of the form $Y = aX + b$ representing a **linear correlation**.

6.2. Secret key field

In the proposed algorithm, the secret key field is set as follows:

ST = {x_0, μ, F, K, D, T, r}.

The initial state of the logistics map x_0=0.1, μ=3.9999, F=10^7, the encryption key can be represented by the following fields:

- ✓ x_0,
- ✓ μ,
- ✓ F: scalar
- ✓ K: starting point or the starting moment k, where we begin to do the encryption.
- ✓ D: Initial state of the register (8 flip-flops = 8 bits)
- ✓ T Mode (Fibonacci / Galois), Fibonacci mode T=0 or T=1 (1bit)
- ✓ r, counter

Where $x0, \mu,$ are double-precision numbers. r and K are integer constants. If the precision of calculating$x0, \mu, is\ 10 - 16$, and K, r ∈ [1, 1000].

Therefore, the keyspace is larger than 2^8x 2^1x 10^{16} x 10^{16} x 10^7 x 10^3 x 10^3=10^{45} (with $10^3 \approx 2^{10}$) in this case we will have a key field of the order of 2^{159}.

We have 159 bits larger of a key.

7. Conclusion

We managed to create a generator that generates very long random sequences based on ordinary generators which we lengthen their period.

We have tested this generator with (Frequency Test, Entropy Test, Average Test, Variance Test, Spectral Test ...) The results are satisfactory. We also encrypted an image with this generator and tested this encryption. The results of the encryption tests are satisfactory.

The use of the logistics map in this paper is essentially related to its recurrent formula, which is very simple to implement. Nevertheless, we recommend you to work with the Piecewise Linear Chaotic Map (PWLCM).

These new generators are tested and are suitable for use in cryptographic applications.

In addition, we have 159 bits larger of key, this number is huge. Therefore, the encryption algorithm has a very large keyspace to withstand all kinds of brute force attacks.

References

[1] Schneier, B., “Applied Cryptography-Protocols, Algorithms and Source Code in C”, John Wiley & Sounds, Inc, New York, Second Edition, (1996).

[2] Menezes, A.J., Oorschot, P.C.V., Vanstone, S. A., “ Handbook of applied cryptography” by CRC Press LLC (1997).

[3] George, M., Alfke, P., “Linear feedback shift registers in virtex devices (application note)”_http://www.xilinx.com/bvdocs/appnotes/xapp210.pdf.

[4] Goresky, M., Klapper, A., “Fibonacci and galois representations of feedback withcarry shift registers”, IEEE Transactions on Information Theory 48(11) (2002 : 2826-2836.

[5] Stackoverflow. “Galois VS Fibonacci LFSR, more computer-friendly but what else?”, novembre 2011, .[https://stackoverflow.com/questions/ 5781458/galois-vs-fibonacci].

[6] S. W. Golomb, Shift Register Sequences, Aegean Park Press, Laguna Hills, CA, 1982 (consulté le 21 mars 2018).

[7] Nyathi, J., Delgado-Frias, J.G., Lowe, J., “A high-performance, hybrid wave--pipelined linear feedback shift register with skew tolerant clocks” 46th IEEE Midwest Symposium on Circuits and Systems, Cairo, Egypt, In Press, Dec. (2003).

[8] Mioc, M.A., Stratulat, M., “Study of software implementation for linear feedback shift register based on 8th degree irreducible polynomials”, International Journal Of Computers 8 (2014) : 46-55.

[9] Devaney, L, “A First course in chaotic dynamical systems”, Westview Press Studies in Nonlinearity (1992).

[10] Gleick, J., “Chaos: Making a new science”, Albin Michel edition (1987).

[11] Knuth, D.E., “The Art of Computer Programming", Addison-Wesley (1998).

[12] Berbain, C., “Analysis and design of stream algorithm - in French language - » PhD thesis, University Paris 7. Diderot, (2007).

[13] Chen, G., Mao, Y., Chui, C. K., (2004), “A symmetric image encryption scheme based on 3D chaotic cat maps”, Chaos, Solitons and Fractals 21 (2004) : 749-761.

[14] Noura, H., “Conception et simulation des générateurs, crypto-systèmes et fonctions de hachage basés chaos performants”, Thèse de Docteur de l’Université de Nantes, (2012).

[15] Li, S., Mou, X., Cai, Y., Ji, Z., Zhang, J., “On the security of a chaotic encryption scheme: problems with computerized chaos in finite computing precision”, Computer Physics Communications 153 (1) (2003) : 52-58.