A Pseudo Random Bit Generator Based on a Modified Chaotic Map

Chokri Nouar and Zine El Abidine Guennoun
(Corresponding author: Chokri Nouar)
Department of Mathematics, Mohamed V University in Rabat
No. 4, Avenue Ibn Battouta B. P. 1014 RP, Rabat, Morocco
(Email: corresponding chokri.nouar@gmail.com)
(Received Dec. 11, 2017; Revised and Accepted Apr. 8, 2018; First Online Jan. 14, 2019)

Abstract

This paper presents a new pseudo random bit generator based on a modified Gingerbreadman chaotic system, The new model has been well studied to avoid fixed and periodic points. We have verified that the system's Lyapunov exponent is positive, which means the system is a chaotic one. The randomness of the bits generated by the proposed generator is successfully tested by the NIST. We notice that during the execution of the algorithm, the password changes automatically after a number of iterations.

Keywords: Chaotic Systems; Gingerbreadman Map; Lyapunov Exponent; NIST; Pseudo-random Bit Generator

1 Introduction

In the mathematical theory, introduced by Devaney in 1988, the Gingerbreadman map is a simple two-dimensional chaotic map. Several studies, which were devoted to the Gingerbreadman map, show the existence of fixed and periodic points [4]. This property is undesirable by cryptography.

The idea of designing a pseudo-random bit generator by making use of chaotic first order nonlinear differential equation was proposed by Oishi and Inoue [9] in 1982. After their paper, several pseudo-random bit generators were suggested. Notice that the algorithms based on chaos showed a good performance for data encryption such as images, videos or audio data [8].

The aim of this paper is to propose a new pseudo-random bit generator based on a chaotic system. We will, firstly, try to eliminate the fixed and periodic points by adding a perturbation function. Then we will verify that the modified Gingerbreadman map is a chaotic system that includes appropriate features such as high sensitivity to initial conditions, unpredictability, mixing property and high complexity. This will allow the system to integrate into various cryptographic applications.

The rest of this paper is structured as follows: the first section presents how the chaotic system affects the produced sequences. In the second section, basic definitions of Lyapunov exponent and The Gingerbreadman map will be recalled. In the third section, we will test the modified Gingerbreadman chaotic system. In section four, we present a detailed description of our generator. Before concluding, the statistical analysis and validation of the bits sequences generated by our generator are given in section five.

2 Chaotic System

A chaotic system is a non-linear deterministic dynamical system, which exhibits pseudo-random behaviour. The output values of a chaotic system vary depending on specific parameters and initial conditions. Different parameter values yield different periods of oscillations at the output of the system.

Chaotic sequences produced by a chaotic system are pseudo-random ones, their structures are very complex and difficult to analyse and to predict. These sequences appear totally random to an external observer, in spite of their deterministic generation, as they are sensitively dependent on initial conditions. In other words, chaotic systems can improve the security of encryption systems.

In mathematics, "Lyapunov exponent" is a quantity that measures the speed at which those small differences are amplified; it actually measures the degree of sensitivity of chaotic systems. Those that have a chaotic behaviour are defined as a chaotic map [10].

The two following sections present a brief description of the Lyapunov exponent and the chaotic map (GingerBreadMan) used in this paper.

2.1 Lyapunov Exponent

In chaotic systems, the distance between two initially close trajectories tends to increase at an exponential speed and then stabilizes when the distance reaches a limit value.

The Lyapunov exponent is an approximate quantity that measures the exponential divergence of initially close trajectories, it also estimates the amount of chaos in any system.

Definition 1. *The Lyapunov exponent L computed using the derivative method is defined by*

$$L = 1/n(ln \mid f'(x_1) \mid + ln \mid f'(x_2) \mid + \cdots + ln \mid f'(x_n) \mid)$$

where f' represents differentiation with respect to x and $x_1, x_2 ... x_n$ are successive iterates. The Lyapunov exponent may be computed for a sample of points near the attractor to obtain an average Lyapunov exponent. [6]

Theorem 1. *If at least one of the average Lyapunov exponents is positive, then the system is chaotic; if the average Lyapunov exponent is negative, then the orbit is periodic. However when the average Lyapunov exponent is zero, a bifurcation occurs [7].*

2.2 The Gingerbreadman Map

In dynamical systems theory, the System (1) is called "the Gingerbreadman map." It was investigated by Devaney in 1984 as a simple module of a chaotic two-dimensional map presented by the following transformation:

$$\begin{cases} x_{n+1} &= 1 - y_n + \mid x_n \mid \\ y_{n+1} &= x_n \end{cases} \quad (1)$$

The Gingerbreadman map is a piecewise linear application, which has been shown to be chaotic in certain regions and stable in others. Figures 1 and 2 displays the first iterations of $(-0,2; 0,2)$.

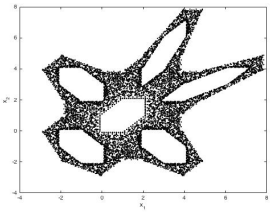

Figure 1: 10000 iterates

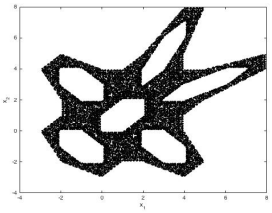

Figure 2: 20000 iterates

Despite its good property, algorithms [3] proved that the Gingerbreadman map has fixed and periodic points in the hexagonal as shown in Figure 3.

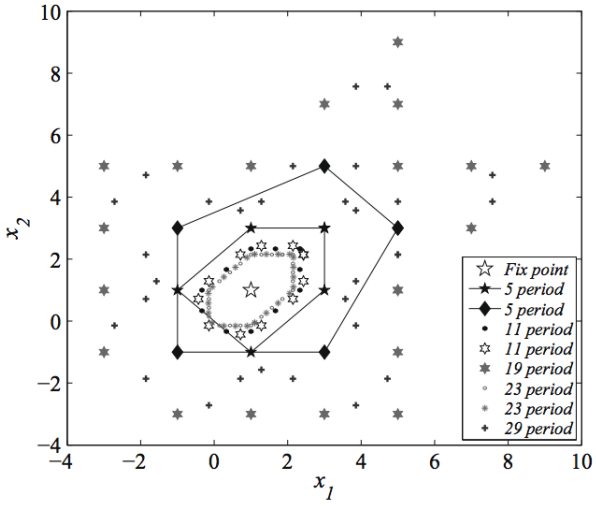

Figure 3: Periodic and fixed points of Gingerbreadman map

new model of Gingerbreadman map (NMGM) is given by the following system

$$H = \begin{cases} x_{n+1} &= 1 - y_n + \mid x_n \mid \\ y_{n+1} &= x_n + r \times \sin(y_n) \end{cases} \quad (2)$$

The aforementioned perturbation function is simple and periodic function to ensure that the dynamical system remains non-linear and deterministic; that is, it does not tend to infinity. It should be noted that it is possible to replace the $\sin(y_n)$ with $\cos(y_n)$ as shown in the Figures 4 and 5.

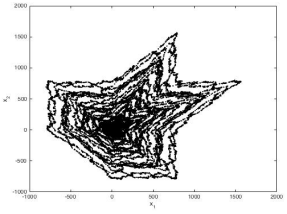

Figure 4: Gingerbreadman with $\cos(y)$ and $r = 3.8$

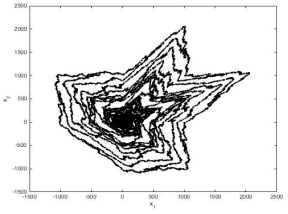

Figure 5: Gingerbreadman with $\sin(y)$ and $r = 3.8$

In the next section, we determine the intervals of the parameter r for which the system remains chaotic, by using the Lyapunov exponent.

3 A Proposed Chaotic System

3.1 Description of the Proposed System

In order to avoid these fixed and periodic points, we add the perturbation function $f(y_n) = r \times \sin(y_n)$ to the second equation in the Gingerbreadman system, where r is a non-arbitrarily chosen real parameter. Therefore, the

3.2 Chaotic Tests of the Proposed System

The Lyapunov exponent of the proposed system varies depending on the parameter r.

The Figure 6 gives the curve of the Lyapunov exponent in function of the parameter r. One can remark that the Lyapunov exponent of the new model of Gingerbreadman

map (NMGM) is positive when $r > 2$ for $x_0 = 0$ and $y_0 = 0$, which implies that the proposed system is chaotic.

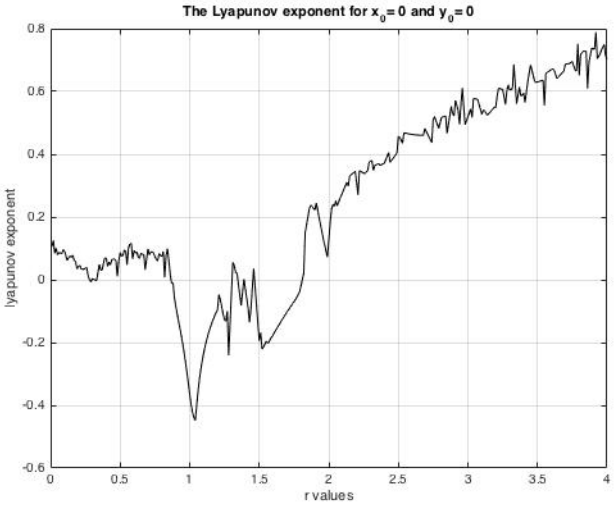

Figure 6: The Lyapunov exponent of the proposed system

Another simple method used to determine whether or not a system is chaotic, is to use the sensitivity to initial conditions. Figures 7, 8, 9 and 10 show how the attractors of the NMGM are affected by small differences in the initial conditions.

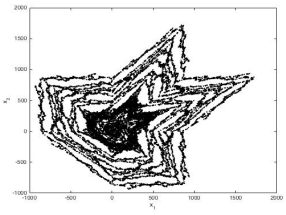

Figure 7: NMGM with $r = 3.8$, $y_0 = 0$ and $x_0 = 0.1$

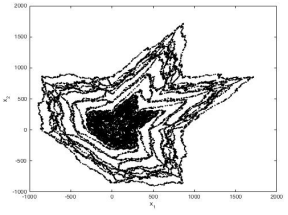

Figure 8: NMGM with $r = 3.8$, $y_0 = 0$ and $x_0 = 0.1 + 0.1 \times 10^{-6}$

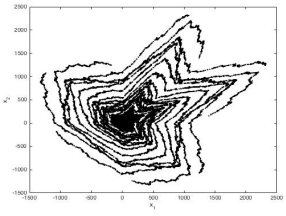

Figure 9: NMGM with $r = 3.8$, $y_0 = 0$ and $x_0 = 0.1 + 0.1 \times 10^{-9}$

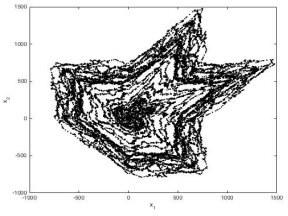

Figure 10: NMGM with $r = 3.8$, $y_0 = 0$ and $x_0 = 0.1 + 0.1 \times 10^{-12}$

3.3 Comparison of the Bifurcation and Lyapunov Exponent

In this section, the comparison between our system and Gingerbreadman map is presented. Figures 1 and 2 shows the bifurcation diagrams of Gingerbreadman map. It is apparent from comparison of Figures 7, 8, 9 and 10, the bifurcation of our system is well distributed.

Figures 11, 12, 13 and 14 shows the Lyapunov exponent of NMGM is more than 0.6 where the Lyapunov exponent of Gingerbreadman map is less than 0.15. So we can concluded that, the NMGM is more chaotic.

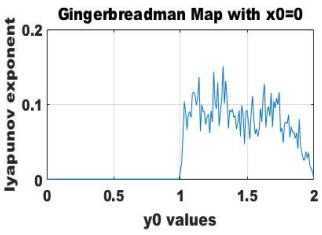

Figure 11: Gingerbreadman map y_0

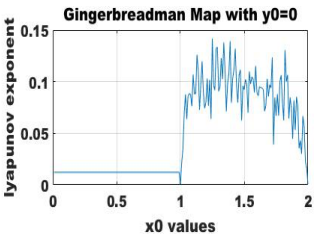

Figure 12: Gingerbreadman map x_0

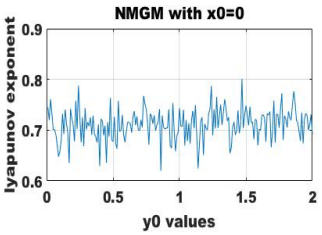

Figure 13: NMGM with y_0

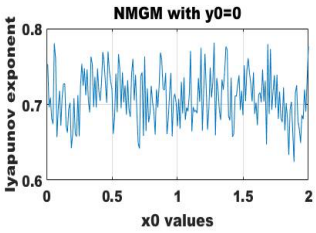

Figure 14: NMGM with x_0

4 Designing a PRBG Based on the NMGM

Our pseudo-random bit generator based on the new model of Gingerbreadman map (PRBG-NMGM) is a deterministic generator initialized by a password Pw of any size, whose output is a cryptographically secure binary sequence.

The initial conditions of our system (NMGM) x_0, y_0 and r_0 are calculated from the password Pw by a method based on a pointer that is positioned on a bit of the Pw. The pointer moves from a position to another according to a linear congruential throughout the ASCII representation of the Pw [2].

After a predetermined number of iterations performed for the three initial conditions x_0, y_0 and r_0, we start generating the numbers needed to construct the final sequence $S = S_1 ... S_n$ with $S_i = X_i \oplus Y_i$, where X_i and Y_i are two 32-bit numbers generated in the i^{th} step.

4.1 The Calculation of the Initialization Values x_0 and y_0

From a binary string of any length n which represents the password $Pw = (P_1P_2...P_n)_2$, we calculate the three initial conditions x_0,y_0 and r_0. For that we extracted 64 bits for each value from the Pw.

We consider a pointer (Z_i) that takes values indicating the bit positions of Pw. The sequences of positions are defined as follows

$$\begin{cases} Z(0) &= \lfloor n/4 \rfloor \\ Z(i+1) &= ((n^2+1) \times Z(i)+1) mod(n) \end{cases} \quad (3)$$

The pointer moves throughout the password and returns to 64 bits stream length, which are classified as follows PT=$(P_{Z(0)}P_{Z(1)}P_{Z(2)}P_{Z(3)}...P_{Z(61)}P_{Z(62)}P_{Z(63)})$.

We calculate the number A B and C from PT ($0 \leq A, B, C < 2^{64}$) as follows:

$$\begin{aligned} A &= (\overline{P_{Z(0)}}P_{Z(1)}P_{Z(2)}\overline{P_{Z(3)}}...P_{Z_{(61)}}P_{Z_{(62)}}\overline{P_{Z(63)}})_2 \\ &= \sum_{i=0}^{21} \overline{P_{Z(3i)}} \times 2^{63-3i} + \sum_{i=0}^{20} P_{Z(3i+1)} \times 2^{63-3i-1} \\ &\quad + \sum_{i=0}^{20} P_{Z(3i+2)} \times 2^{63-3i-2} \quad (4) \\ B &= (P_{Z(0)}\overline{P_{Z(1)}}P_{Z(2)}P_{Z(3)}...\overline{P_{Z_{(61)}}}P_{Z_{(62)}}P_{Z(63)})_2 \\ &= \sum_{i=0}^{20} P_{Z(3i)} \times 2^{63-3i} + \sum_{i=0}^{21} \overline{P_{Z(3i+1)}} \times 2^{63-3i-1} \\ &\quad + \sum_{i=0}^{20} P_{Z(3i+2)} \times 2^{63-3i-2} \quad (5) \\ C &= (P_{Z(0)}P_{Z(1)}\overline{P_{Z(2)}}P_{Z(3)}...P_{Z(61)}\overline{P_{Z(62)}}P_{Z(63)})_2 \\ &= \sum_{i=0}^{20} P_{Z(3i)} \times 2^{63-3i} + \sum_{i=0}^{20} P_{Z(3i+1)} \times 2^{63-3i-1} \\ &\quad + \sum_{i=0}^{21} \overline{P_{Z(3i+2)}} \times 2^{63-3i-2} \quad (6) \end{aligned}$$

Finally, $x_0 = \frac{A}{2^{63}}, y_0 = \frac{B}{2^{63}}$ and $r_0 = \frac{C}{2^{63}} + 2$. So the initial conditions values are in the fallowing intervals: $0 \leq x_0 < 2$, $0 \leq y_0 < 2$ and $2 \leq r_0 < 4$.

Algorithm 1 is used to calculate initial values x_0, y_0 and r_0

4.2 Generating the Pseudo-random Sequence

After extracting the initial values x_0, y_0 and r_0, the system (PRBG-NMGM) will be ready to generate the pseudo random bits sequences.

Algorithm 2 of generating has two input parameters, a password Pw and an integer F that indicates the length of the output binary sequence.

Algorithm 1 Initialization

Input password $Pw = (P_1P_2...P_n)_2$ a binary string of any length n
Output initiation values x_0, y_0 and r_0
$Z \leftarrow \lfloor n/4 \rfloor$
$A, B, C \leftarrow 0$
for $i \leftarrow 0$ **to** 63 **do**
  $Z \leftarrow ((n^2+1) \times Z + 1) mod(n)$
  if i mod (3) $= 0$ **then**
    $A \leftarrow A + \overline{P_Z} \times 2^{63-i}$
    $B \leftarrow B + P_Z \times 2^{63-i}$
    $C \leftarrow C + P_Z \times 2^{63-i}$
  else
    if i mod (3) $= 1$ **then**
      $A \leftarrow A + P_Z \times 2^{63-i}$
      $B \leftarrow B + \overline{P_Z} \times 2^{63-i}$
      $C \leftarrow C + P_Z \times 2^{63-i}$
    else
      $A \leftarrow A + P_Z \times 2^{63-i}$
      $B \leftarrow B + P_Z \times 2^{63-i}$
      $C \leftarrow C + \overline{P_Z} \times 2^{63-i}$
    end if
  end if
end for
$x_0 \leftarrow \frac{A}{2^{63}}$; $y_0 \leftarrow \frac{B}{2^{63}}$; $r_0 \leftarrow \frac{C}{2^{63}} + 2$;
return $x_0; y_0; r_0$

Step 1: In the first step leaving the system NMGM looping up to n_0 iterations to avoid the harmful effects of transitional procedures [1], where n_0 is determined from the length of Pw such as

$$n_0 = r_0 \times length(Pw).$$

Step 2: The iteration of the system NMGM continues, for each $i \leq F$ and $mod(i, length(Pw)) \neq 0$ we obtain the pairs (x_i, y_i) to construct the sub-sequence $S_i = X_i \oplus Y_i$ such as

$$X_i = floor(mod(x_i, 1) \times 2^{32}),$$
$$Y_i = floor(mod(y_i, 1) \times 2^{32}),$$

where the $floor(x)$ returns the largest integer less than or equal to x, $mod(x, y)$ returns the reminder after division of x by y, X_i and Y_i are two 32-bit numbers generated in the i^{th} step, and $\oplus$ represents operator of exclusive-OR.

Step 3: If $mod(i, length(Pw) = 0$ the parameter r_j of the system NMGM change automatically, we obtain the new r_j with the following equation:

$$r_{j+1} = (r_j + 1)^2 mod(2) + 2,$$

else return to step 2 until the bit stream limit is reached.

The output S of the PRBG-NMGM is the concatenation of the sub-sequences $S_1S_2...S_i...S_F$. Figure 15 shows scheme of i^{th} generation step of our PRNG-NMGM.

Algorithm 2 Generation

Input password $Pw = (P_1P_2...P_n)_2$ and F the length of the requested binary sequence
Output S the random binary sequence
$x, y, r \leftarrow$ Initialize(Pw)
$M \leftarrow r \times length(Pw)$
$S \leftarrow 0$
$i, j, k \leftarrow 0$
while $j \leq F$ **do**
  if $i \leq M$ **then**
    $(x_{i+1}, y_{i+1}, r) \leftarrow H(x_i, y_i, r)$
    $i \leftarrow i + 1$
  else
    if $i\%n \neq 0$ **then**
      $(x_{i+1}, y_{i+1}, r) \leftarrow H(x_i, y_i, r)$
      $X \leftarrow \lfloor (x_{i+1} mod(1) \times 2^{32} \rfloor$
      $Y \leftarrow \lfloor (y_{i+1} mod(1) \times 2^{32} \rfloor$
      $R = X \bigoplus Y$
      $S \leftarrow S \parallel R$
      $i \leftarrow i + 1$
      $j \leftarrow j + 1$
    else
      $r \leftarrow (r+1)^2 mod(2) + 2$
      $i \leftarrow i + 1$
    end if
  end if
end while
return S

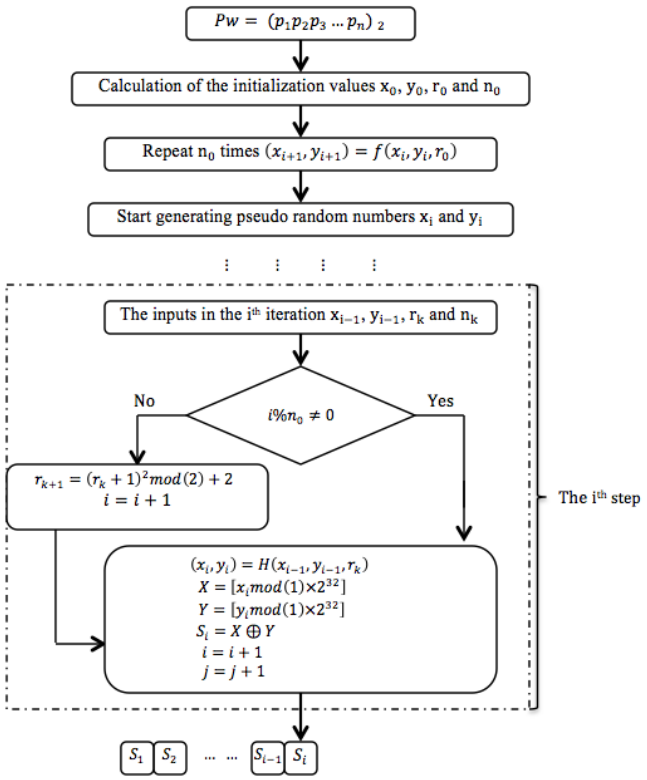

Figure 15: Scheme of i^{th} generation step of our PRNG

5 Security Analysis

Security analysis is a tool for evaluating the performance of our proposal PRBG-NMGM. In this section we examine the key space size, the sensitivity to the initial conditions and the randomness level of the generated sequences.

5.1 Key Space

Among the most important criteria of a cryptosystem is the size of key space. A space large enough security keys make exhaustive attacks infeasible. Our PRBG-NMGM initialized by a key of any size as already mentioned.

On the other hand a key space of size larger than 2^{128} is computationally secured against exhaustive attacks [5], therefore, the key size N must be greater than 128.

The calculation of the three initial values x_0, y_0 and r_0 needs exactly 192 bits, which are extracted via a pointer that traverses the binary string, so the space of initial values is 2^{192}. This leads us to say that the size of the key space is large enough to be attacked exhaustively.

5.2 Key Sensitivity

The key sensitivity implies that the small change in the secret key should produce a big change in the pseudo-random sequences, this property is essential to make a highly secured PRBG against statistical and differential attacks. This property is also basic for the PRGB not to be broken even if there is a small difference between the keys. The proposed generator is based on a chaotic map of the positive Lyapunov exponent meaning that is very sensitive to the initial conditions.

In order to examine the security of our generator, we have performed the key sensitivity test; we place several keys k_i in the input of the generator with a bit of difference between them, then we calculate the hamming distance between two pseudo-random sequences S_i of the size N generated by each key.

The calculation of the hamming distance between two binary sequences is the number $DH(S_i, S_j) = card\{e/x_e \neq y_e\}$ with $S_i = x_1x_2...x_N$ and $S_j = y_1y_2...y_N$. In general, this distance is given by:

$$DH(S_i, S_j) = \sum_{k=1}^{N} (x_k \bigoplus y_k).$$

The fact that a generator is very sensitive to the key makes the hamming distance vary in the neighbourhood of $N/2$, resulting the $DH(S_i, S_j)/N$ being about 0.50 for each pair of sequences produced.

In the next test we will generate a set of pseudo-random sequences $\{S_i\}$ from the keys $\{k_i\}_{0 \leq i \leq 55}$.

We consider $k_0 =$ "$ABIDINE$" whose binary representation in ASCII code is $k_0 = (01000001\ \ 01000010\ \ 01001001\ \ 01000100\ \ 01001001\ \ 01001110\ \ 01000101)_2$. The other 55 keys $\{k_i\}_{1 \leq i \leq 55}$ are derived from the k_0,

modifying the i^{th} bit among the 56 bits of the k_0 to find k_i.

The result of the Hamming distance between the sequences is given in Figure 16.

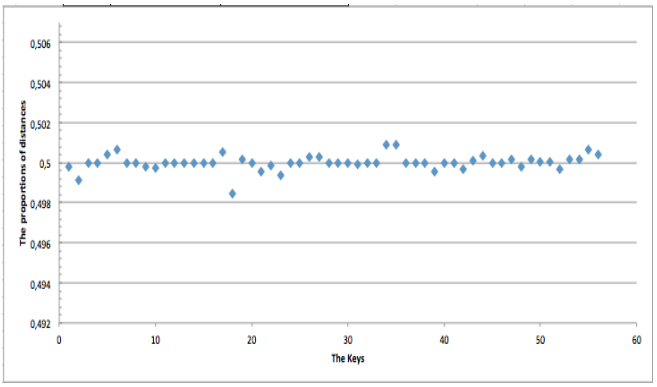

Figure 16: The Hamming distance between the sequences

It is clear that the proportions of difference between the sequences are about 50%, which implies that the proposed generator is purely sensitive to the initial conditions.

This sensitivity is due to the chaotic system that constructs the generator, meaning that the generated sequences are chaotic and unpredictable. It is also due to the initialization values of the PRBG-NMGM that are derived from a password, using a method based on a Linear Congruential Generators.

Indeed, a change of one bit between two keys leads to a totally different initialization values as well as different generated sequences.

5.3 Randomness Level

We used the NIST tests and DIEHARD tests in order to measure the level of randomness of the bits sequences generated by PRBG-NMGM .

The NIST tests suite consists of 15 tests developed to quantify and to evaluate the degree of randomness of the binary sequences produced by the cryptographic generators.

These tests are: frequency (monobit), block-frequency, cumulative sums, runs, longest run of ones, rank, Fast Fourier Transform (spectral), non-overlapping templates, overlapping templates, Maurers Universal Statistical, approximate entropy, random excursions, random-excursion variant, serial, and linear complexity.

For each statistical test, a P_{value} is calculated from the bit sequence. This P_{value} is compared to a predefined threshold $\alpha = 0.01$, which is also called significance level.

If P_{value} is greater than 0.01, then the sequence is considered to be random, and it proceeds the statistical test successfully. Otherwise, the sequence does not appear random.

To apply the NIST tests on our generator we generated 1000 sequences, the size of each sequence is 10^6 bit from a different key. The table 1 bellow presents the test results in the sequences.

Table 1: NIST statistical test suite results for 1000 sequences of size 10^6 bit each generated by the our generator

NIST statistical test	P_{value}	Pass rate
Frequency	0,800005	989/1000
Block-Frequency	0,100709	990/1000
Cumulative Sums	0,233162	989/1000
Runs	0,350485	994/1000
Longest Run	0,719747	989/1000
Rank	0,402962	995/1000
FFT	0.345650	987/1000
Non-Overlapping	0,509841	990/1000
Overlapping	0.709558	987/1000
Universal	0.390721	990/1000
Approximate Entropy	0.846338	986/1000
Random Excursions	0,338148	620/628
Random Excursions Variant	0,592461	623/628
Serial	0,490572	990/1000
Linear Complexity	0.805569	987/1000

The minimum pass rate for each statistical test with the exception of the random excursion (variant) test is approximately 980 for a sample size 1000 binary sequences.

The minimum pass rate for the random excursion (variant) test is approximately 613 for a sample size 627 binary sequences.

The DIEHARD tests consists of a set of statistical tests for measuring the quality of randomness, developed by George Marsaglia, these tests are: birthday spacings, overlapping 5-permutations, binary rank (31 x 31), binary rank (32 x 32), binary rank (6 x 8), bitstream, Overlapping-Pairs-Sparse-Occupancy, Overlapping-Quadruples-Sparse-Occupancy, DNA, stream count- the-ones, byte-count-the-ones, 3D spheres, squeeze, overlapping sums, runs up, runs down, craps.

For the DIEHARD tests, we generated 1000 sequences of one million bits each, by the proposed pseudo-random bit generators. The results are given in Table 2.

We can see from **Table 1** that the NIST tests suite is passed successfully. The p_{value} of all tests is greater than the minimum rate (0.01).

Table 2 shows the DIEHARD P_{values} are in acceptable range of $(0, 1)$, and all tests are passed successfully.

Based on these results, the NMGM based random generator is suitable for cryptographic applications.

6 Conclusion

In this paper, a novel pseudo-random bits generator based on modified chaotic map was presented.

Table 2: DIEHARD statistical test suite results for 1000 sequences of size 10^6 bit each generated by the our generator

DIEHARD test name	P_{value}	Assessment
Birthday	0.92580677	passed
Overlapping 5-permutation	0.22867882	passed
Binary rank (32 x 32)	0.82601210	passed
Binary rank (6 x 8)	0.59379048	passed
Bitstream	0.90091209	passed
OPSO	0.09739253	passed
OQSO	0.54519450	passed
DNA	0.17853645	passed
Stream count-the-ones	0.50254509	passed
Byte count-the-ones	0.39753149	passed
Parking lot	0.91507220	passed
Minimum distance	0.91222820	passed
3D spheres	0.41362890	passed
Squeeze	0.63363326	passed
Runs up	0.09807447	passed
Runs down	0.49918763	passed
Craps	0.61161595	passed

The new model has been selected after a rigorous analysis that showed high dimensional chaotic which generate more complex and unpredictable chaotic sequences.

The results of statistical analyses like randomness, key space and key sensitivity indicate high security and suitability of the proposed generator for practical encryption.

In future works, we will apply our proposal PRBG-NMGM especially in audio and image encryption and in other cryptographic applications

References

[1] S. Borislav and K. Krasimir, "Novel secure pseudo-random number generation scheme based on two tinkerbell maps," *Advanced Studies in Theoretical Physics*, vol. 9, no. 9, pp. 411–421, 2015.

[2] K. Charif, A. Drissi, and Z. E. A. Guennoun, "A pseudo random number generator based on chaotic billiards," *International Journal of Network Security*, vol. 19, pp. 479–486, May 2017.

[3] G. Feiab, F. Feng-xiaa, D. Yan-fanga, Q. Yi-boa, and I. Balasingham, "A novel non-lyapunov approach through artificial bee colony algorithm for detecting unstable periodic orbits with high orders," *Expert Systems with Applications*, vol. 39, pp. 12389–12397, Nov. 2012.

[4] F. Gao, H. Gao, Z. Li, H. Tong, and J. J. Lee, "Detecting unstable periodic orbits of nonlinear mappings by a novel quantum-behaved particle swarm optimization non-lyapunov way," *Chaos, Solitons and Fractals*, vol. 42, pp. 2450–2463, Nov. 2009.

[5] E. Hato and D. Shihab, "Lorenz and rossler chaotic system for speech signal encryption," *International Journal of Computer Applications*, vol. 128, pp. 25–33, Oct. 2015.

[6] M. B. Jacques, C. Jean-François, and G. Christophe, "Quality analysis of a chaotic proven keyed hash function," *International Journal On Advances in Internet Technology*, Aug. 2016. (`https://arxiv.org/abs/1608.05928`)

[7] S. Lynch, *Dynamical Systems with Applications Using MATLAB*, Boston, USA: Birkhuser, 2014. (`https://www.springer.com/us/book/9783319068190`)

[8] A. Musheer, A. Bashir, and F. Omar, "Chaos based mixed keystream generation for voice data encryption," *International Journal on Cryptography and Information Security*, vol. 2, no. 1, pp. 36–45, 2014.

[9] S. Oishi and H. Inoue, "Pseudo-random number generators and chaos," *IEICE Transactions*, vol. E65, pp. 534–541, September 1982.

[10] R. Swati and T. Sanjeev, "Security analysis of multimedia data encryption technique using piecewise linear chaotic maps," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 1, pp. 458–461, May 2013.

Biography

Chokri NOUAR He is received his Master's degree in mathematics and statistics, option cryptography and information security from Mohammed-V University in Rabat, Morocco. He is actually a PhD student in the Laboratory of Mathematics, statistic and applications. His major research interests include information security and cryptography.

Zine El Abidine GUENNOUN He is a full professor of Department of Mathematics at the Faculty of Science, Mohamed V University in Rabat, Morocco. He received his Ph.D. (1989). His research interests include non linear analysis, fixed point theory, differential equation, financial mathematics and cryptography.