Vol. 5(15) Apr. 2015, PP. 2120-2129

Contents list available at IJMEC

International Journal of Mechatronics, Electrical and Computer Technology (IJMEC)

Journal Homepage: www.aeuso.org

PISSN: 2411-6173 - EISSN: 2305-0543

# A Pseudo Random Number Generator Based on Chaotic Henon Map (CHCG)

Behrouz Fathi Vajargah[1] and Rahim Asghari[2]

[1]*Department of Statistics, Faculty of Mathematical Science, University of Guilan, Rasht, Iran.*

[2]*Department of Applied Mathematics, Faculty of Mathematical Science, University of Guilan, Rasht, Iran.*

***Corresponding Author's E-mail:** fathi@guilan.ac.ir*

## *Abstract*

In the present paper, the properties of making a deterministic algorithm for generating a pseudo random sequence of numbers for application in cryptography, are discussed. The Chaotic Henon Congruential Generator is proposed as a pseudo random number generator. It is shown that what chaotic features of the Henon Map are useful for generating pseudo random numbers in cryptographic point of view. To evaluate the randomness as well as independence of the number sequences, generated by the PRNGs, two well-known statistic tests were performed and the results approved that the generated sequences are statistically proper in cryptographic applications.

***Keywords:*** *Chaotic function, Pseudo random numbers sequence, Henon Map, Linear Congruential Generator, Correlation Test, Chi-square Goodness-of-fit Test.*

## 1. Introduction

Complexity of communications in human society is increased by developments in electronic communication technologies. Such complexity needs to provide authentication in prescribed communications. Cryptography is an attempt for this vital requirement. Creating efficient crypto algorithms are recently becoming the subject of researches in academic societies, which depicts human society worries about authentication. The security in most crypto systems is highly depends on generating unpredictable numbers, e.g. hidden key in DES, prime numbers p, q, in RSA, and key stream in stream ciphers to guaranty that generated numbers are not guessable. As generating true random numbers (e.g., Johnson Noise) are very difficult in practice and regeneration of them is not possible, debugging and testing of programs become difficult, therefore pseudo random numbers (generated by mathematical algorithms) are applied in daily applications. The stream cipher technique has a special importance among other crypto algorithms and has a maximum dependency to pseudo random number generators. In designing stream cipher, a single pseudo random bit generator, plays the role of the key stream generator for the stream cipher system which is indeed generator of key stream. From the cryptographically point of view, a key stream generator should have the following important parameters:

- The period of generating key should be sufficiently large to be consistent with the size of the sent message
- Generating bit sequence should be practical and easy
- Generated bits should be unpredictable

Also, it should be noted that in order to guaranty unpredictability, the key stream should have two important properties: independence of generating numbers and having large period. These properties can be tested by statistical tests.

Recently most of practical stream ciphers are based on (LFSR) which makes stream ciphers practical and efficient [1]. But LFSR and therefore stream ciphers are inefficient in software [13]. In 1984 Blum



***Article History:***

**Received Date: Jan. 04, 2015**

**Accepted Date: Mar. 11, 2015**

**Available Online: Apr. 10, 2015**

and Micali described how to generate a PRBG [15]. In 1999 Pascal Junod discussed and proved the security of The Blum-Blum-Shub Generator from a cryptographic point of view [11], and it was implemented in 2014 with Aïssa et. al. [4, 1]. In 2003 Edkal in his Ph.D. thesis discussed the design and analysis of stream cipher based on LFSR [15]. In 2006, Parschi studied and analyzed Chaos-based random number generators in the University of Bolonia [6]. In 2009 Krhovj´ak studied the relation between cryptography and PRNG in his Ph.D. thesis [8]. And in 2013 Babu and Kumar described the design of a new stream cipher based on PRNG [17].

In the present paper, we discussed the design of a PRNG based on LCG and Chaotic Henon map which has very simple implementation and is fast in generating pseudo random numbers (PRNs). On the other hand, with statistical tests we prove the independence as well as a scatter of generating PRNs. The most important property of generating PRNs by the proposed Algorithm is a long period of the sequences which is vital in cryptography. This property is illustrated within a numerical example.

In this work, first, we present the notion of pseudo random generators (PRNG) and Henon map. In the second part, we design a new PRNG based on Henon map. In the third part, the proof of its security is treated in details by two suit statistical tests.
The paper is organized as follows: In section 2, pseudo random numbers and Discrete Chaotic Henon Map and the Linear Congruential Generator are introduced, in section 3, we present Chaotic Henon Congruential Generator, in section 4, statistical tests are implemented and in section 5 a numerical example is presented including some comparisons. Finally in section 6 we drive the conclusion.

## 2. Preliminaries

### 2.1. Pseudo-Random Number Generator

Random numbers are critical in every aspect of cryptography. We need such numbers to encrypt e-mails, to digitally sign documents, for electronic payment systems, and so on. In the context of cryptography, a random number is a number that cannot be predicted by an observer before it is generated. Unfortunately, true random numbers are very difficult to generate, especially on computers that are typically designed to be deterministic. This brings us to the concept of pseudo-random numbers, which are numbers generated from some random internal values, and that are very hard for an observer to distinguish from true random numbers.

“Pseudo”, because generating numbers using a known method removes the potential for true randomness. The Goal of the pseudo random number generator is to produce a sequence of numbers in the interval [0,1] that simulates, or imitates, the ideal properties of random numbers (RNs).

Two important statistical properties of the pseudo random number generator are it uniformity and independence.
The most well-known pseudo random number generator are:

- Midsquare method
- Linear Congruential Method (LCM)
- Combined Linear Congruential Generators (CLCG)
- Random-Number Streams

Since the PRNG runs on a computer, which is finite, it must be a finite state machine.
Formally, a PRNG is defined by three things:

1. An initial state $s_0$;
2. A transition function $S$ on states, such that $s_{k+1} = S(s_k)$ for all $k \geq 0$;
3. An output functions $V$ on states, such that for all$k \geq 0$, $V(s_k)$ is the pseudorandom number at $k$th invocation of the PRNG.

Figure1 illustrates these three parts of the PRNG. The sequence of pseudorandom numbers produced by the PRNG is $(s_0)$, $V(s_1)$, … .

The transition function of most PRNG's is a recurrence relation based on modular integer arithmetic [9].

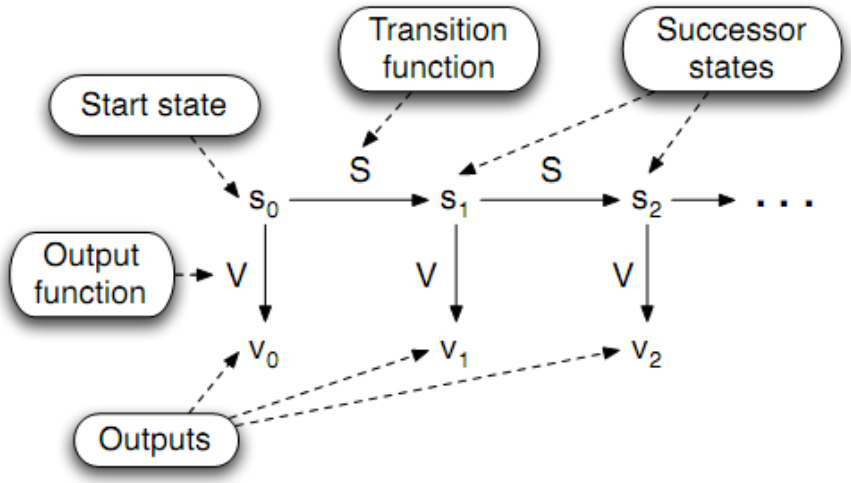



**Figure1:** The state model of a pseudorandom number generator

### 2.2. Chaotic Henon map

In 1989, Matthews suggested a new efficient way to deal with the intractable problem of fast and highly secure encryption by proposing the chaotic encryption algorithm. Chaotic system exhibit dynamics that are sensitive to initial conditions [1, 19]. The basic property of chaotic systems is the sensitivity to initial conditions. Using chaotic cryptosystems in practical applications is successful to increase the security level for cryptographic systems. Henon map is represented by the state equations with a chaotic attractor and is a simplified model of the Poincare map for the Lorenz equation proposed by henon in 1976 [1, 19].

The two dimensional henon maps is given in the following equation:

$$\begin{cases} x_{k+1} = -ax_k^2 + y_k + 1 \\ y_{k+1} = bx_k \end{cases} \quad (1)$$

The pair $(x, y)$ is the two dimensional state of the system. The initial point is $(x_0, y_0)$.

### 2.3. The Linear Congruential Generator

Lehmer proposed a simple linear congruential generator as a source of random numbers [4]. Although these processes are completely deterministic, it can be shown that the numbers generated by the sequence appear to be uniformly distributed and statistically independent. Understanding its properties is necessary in order to use it to build better generators[3].
The form of the linear congruential generator is:

$$\begin{aligned} x_i &\equiv ax_{i-1} + c \pmod m, \\ 0 &\le x_i < m \qquad i = 1, 2, \ldots \end{aligned} \quad (2)$$

Where the multiplier a, the increment c, and the modulus m are nonnegative integers.
Parameters:
a) $m$, the modulus, $m > 0$
b) a, the multiplier, $0 < a < m$
c) c, the increment, $0 < c < m$
d ) $x_0$, the seed, $0 < x_0 < m$

Choice of a, c and m is important. The LCG passes the usual statistical tests when initialized and used properly.

M should be large, prime, e.g., $2^{31} - 1$. If c=0, few good values of a, e.g., $7^5 = 16807$.

Two LCGs can be combined to create a combined linear congruence generator, CLCG. With good constants in the underlying LCGs the generator has a period that is the product of the period of each LCG. The first CLCG was presented by the Wichmann and Hill generator [14]. The equivalence between this generator and an LCG was shown by Zeisel [7].

**2.3.1. Moduli and Multipliers:** A commonly used modulus is the Mersenne prime $2^{31}-1$, and for that modulus, a common multiplier is 75. The Mersenne prime $2^{61} - 1$ is also used occasionally. The primitive roots for these two moduli have been extensively studied. Wu in [12] suggests multipliers of the form $\pm 2^{q1} \pm 2^{q2}$ because they're resulted in particularly simple computations, yet seem generally to have good properties.Wu suggests $2^{15} - 2^{10}$ and $2^{16} - 2^{21}$ for a modulus of $2^{31} - 1$, and $2^{30} - 2^{19}$ and $2^{42} - 2^{31}$ for a modulus of $2^{61} - 1$.

The number of 1s in the binary representation of a value is called its Hamming weight. Lecuyer and Simard in [14] defined a test of independence of Hamming weights of successive values in the output streams of random number generators and, in applying the test to generators with multipliers of the form $\pm 2^{q1} \pm 2^{q2}$, found that such generators perform poorly with respect to this criterion.

## 3. Chaotic Henon_ Congruential Generator (CHCG)

As mentioned before, a problem of Linear Congruential Generator (LCG) has considerably small period. That's why LCG is not sufficient for encryption. In this paper, we want to combine the LCG and henon map, until the generated number becomes suitable for application in encryption. Our purpose is to have a key stream with high period. As we know, henon map is sensitive to initial value and has chaotic behavior. Because of this property we are going to use the henon map when the number, generated by LCG, is same. With this procedure, it is possible to generate a suitable key stream.

**Algorithm 1:** CHCG

**Algorithm1: CHCG**

*select* $x_0$

*for* $i = 1$ *to* $n$

$x_i = a x_{i-1} + b \pmod m$

*for* $j = 1$ *to* $i - 1$

*if* $x_i == x_j$

$x_{i+1} = 1 - \alpha x_i + \beta x_{i-1}^2$

*end if*

*end for*

*end for*

## 4. Statistical Tests Results

### 4.1. Correlation Test

This statistic test checks the "independency condition" of numbers generated by pseudo random number generator [16]. We can set numbers in two subset of$x, y$. Next, we calculate covariance of $x, y$ and correlation coefficient by following:

$$\rho = \frac{cov(x,y)}{\mathrm{var(x)var(y)}} \tag{3}$$

$$cov(x,y) = \frac{\sum_{i=1}^{n} x_i y_i}{\mathrm{n}} - \bar{x}\bar{y} \tag{4}$$

$$var(x) = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \tag{5}$$

As it is seen, If ρ be close to zero, then the generated numbers becomes independent, otherwise if ρ be away from zero, then the independency will fail.

**4.1.1. Correlation Test result:** We generate 500000 numbers by chaotic henon congruential generator, then correlation test is applied for these numbers. We getρ $= 0.0027$. This result shows that, generated numbers are approximately independent in the sense that to be applied in cryptography.

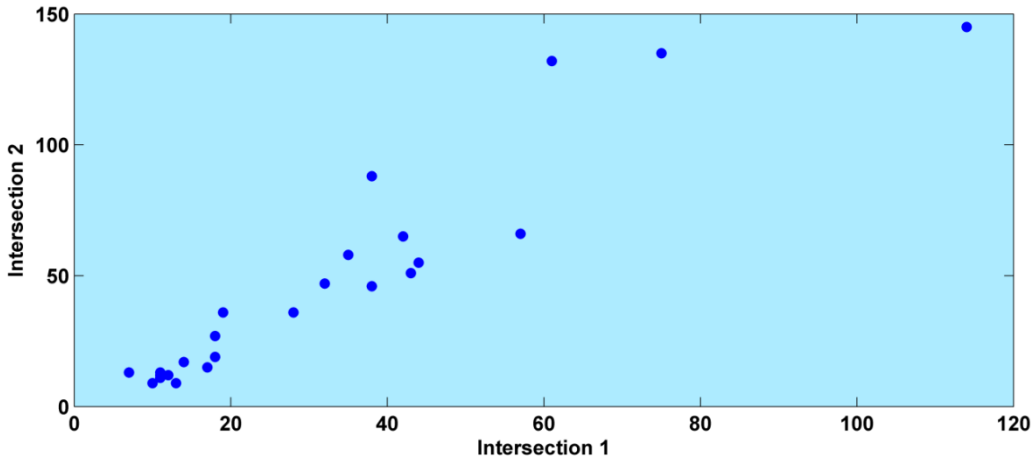



**Figure2:** Scatter plot of the Generated numbers by CHCG shows a positive correlation

### 4.2. Chi-square Goodness-of-fit Test

The chi-square goodness-of-fit test, proposed by Pearson in 1900 is perhaps the best known among all other statistical tests. It is designed for testing discrete distributions and large samples. The test can be used for testing any distribution: uniform random number generators as well as random variate generators. The statistical test formula is of the form:

$$\sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i} \le \chi^2_{[1-\alpha,k-1]} \tag{6}$$

Where;

a. k is the number of bins in the histogram.

b. $o_i$ is the number of observed values in bin i in the histogram.

c. $e_i$ is the number of expected values in bin i in the histogram.

d. The test results can be presented as follows: if the sum is less than $\chi^2_{[1-\alpha,i-1]}$, then the hypothesis that the observations come from the specified distribution cannot be rejected at a level of significance α [11].

**4.2.1. Chi-Square Constraints:**

1. Data must be a random sample of the population to which you wish to generalize the claims.
2. Data must be reported in raw frequencies (not percentages).
3. Measured variables must be independent.
4. Values/categories on independent and dependent variables (must be mutually exclusive and exhaustive).
5. Observed frequencies cannot be too small.
6. Use Chi-square test only when observations are independent, no category or response is influenced by another.
7. Chi-square is an approximate test of the probability of getting the frequencies you've actually observed if the null hypothesis were true.

**a**. Based on the expectation that within any category, sample frequencies are normally distributed about the expected population value.
**b**. Distribution cannot be normal when expected population values are close to zero since frequencies cannot be negative.
**c**. When expected frequencies are large, there is no problem with the assumption of normal distribution.

**4.2.2. Chi-square Goodness of fit Test Result**

500000 numbers are generated by Chaotic Henon Congruential Generator, and then these numbers are tested via chi-square Goodness-of-Fit. Final result is obtained as following:

$$\begin{cases} n = 500000 \\ k = 30 \\ e_i = 500000/30.0 \end{cases} \Rightarrow \sum_{i=1}^{30} \frac{(o_i - e_i)^2}{e_i} = 15.7125 \le \chi^2_{[0.5,29]} = 17.786$$

According to the result of the Chi-Square test, we cannot reject the null hypothesis that CHCG generates pseudo random numbers with only 5% confidence.

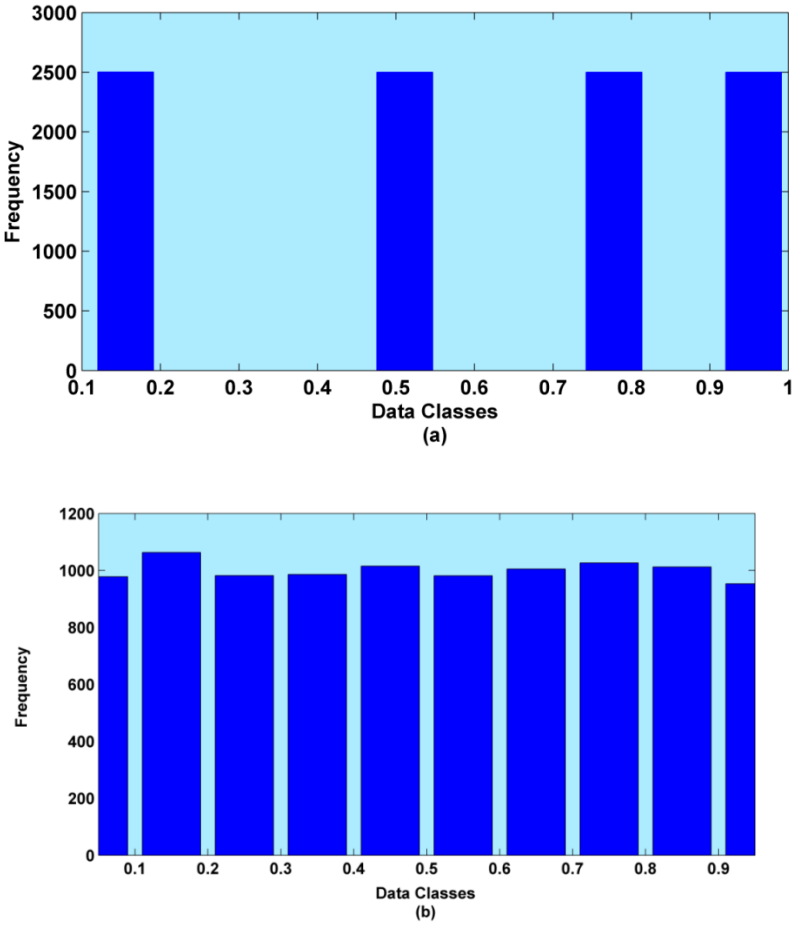
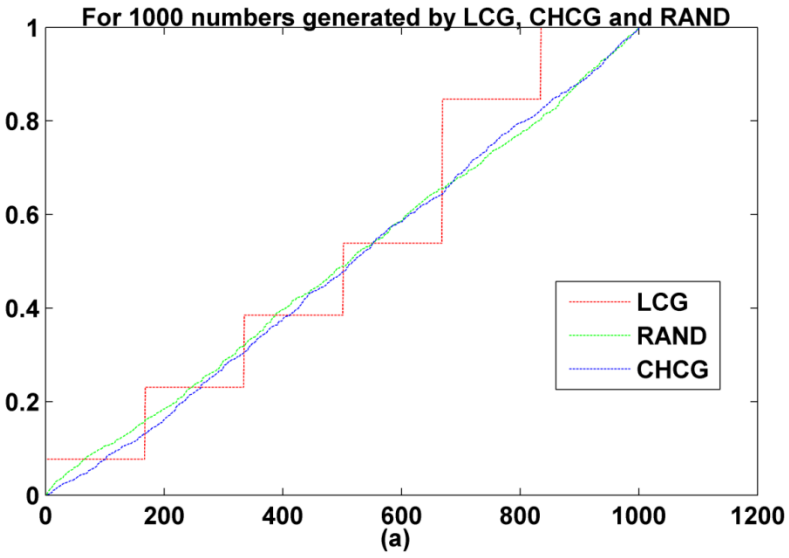



Figure 3: Histogram of frequency: frame (a) and frame (b) respectively, for 10000 generated numbers by LCG and CHCG



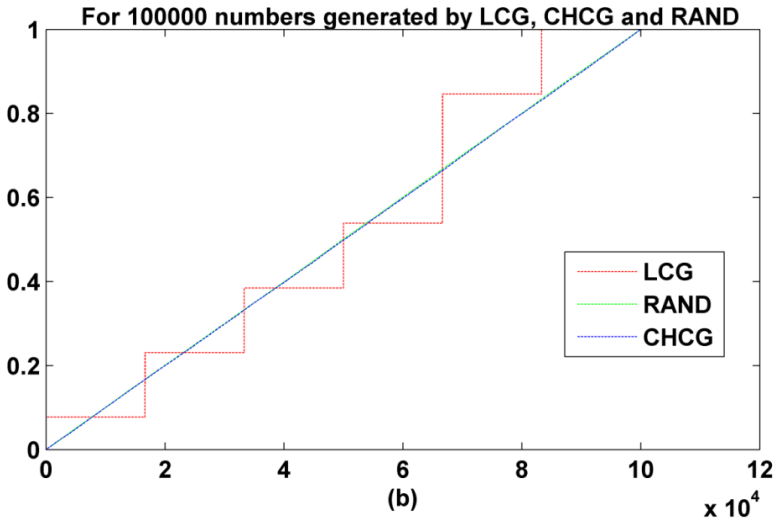



**Figure4:** Distribution of numbers generated by LCG,CHCG and RAND for 1000 numbers (a) and 100000 numbers (b)

Figures 3 and 4 are illustrating results of the statistical tests for uniformity of the generated numbers. Figure 3 compares uniformity of distribution of the generators, CHCG (b) w.r.t LCG (a) it can be seen that the presented method, i.e. CHCG is very close to uniform distribution, comparing with LCG method. In Figure 4 (a) and (b) we do the tests for 1000 and 100000 tests respectively. It can be seen that when we increase the number of samples, the CHCG method becomes very close to uniform distribution. In Figure 4 (b) we can see that numbers generated by CHCG are fall into those of RAND function.

## 5. Example

First we run Linear Congruential Generator (LCG) with $x_1 = 0.5, a = 3, b = 7, m = 5$.

Generated numbers sequence is given in Table 1.

**Table1**: Generated numbers by CHCG

| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| x(i) | 0.1111 | 0.7778 | 0.5556 | 1.0000 | 0.1111 | 0.7778 | 0.5556 | 1.0000 | 0.7778 | 0.5556 |
| i | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| x(i) | 1.0000 | 0.1111 | 0.7778 | 0.5556 | 0.7778 | 1.0000 | 0.1111 | 0.7778 | 0.5556 | 1.0000 |

Next, we run new generator (CHCG) with $x_1 = 0.5, a = 3, b = 7, m = 5$.

Generated numbers sequence is given in Table 2.

**Table2**: Generated numbers by CHCG

| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| x(i) | 0.1111 | 0.7778 | 0.5556 | 1.0000 | 0.1111 | 0.5401 | 0.0200 | 0.4601 | 0.7801 |
| i | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| x(i) | 0.7401 | 0.6201 | 0.2600 | 0.1800 | 0.9401 | 0.2200 | 0.0600 | 0.5801 | 0.1400 |
| i | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| x(i) | 0.9400 | 0.8200 | 0.06208 | 0.3800 | 0.6250 | 0.8591 | 0.9395 | 0.2181 | 0.0542 |
| i | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 |
| x(i) | 0.0205 | 0.0614 | 0.1843 | 0.5530 | 0.6590 | 0.9770 | 0.9311 | 0.7933 | 0.3816 |

## 6. Result

The result is so surprised. In first Generated numbers sequence, period of numbers sequence is 4. But at the new generator (CHCG), period of numbers sequence is very high, so that in 100000 Generated numbers, just one number has been iterative as x (5).

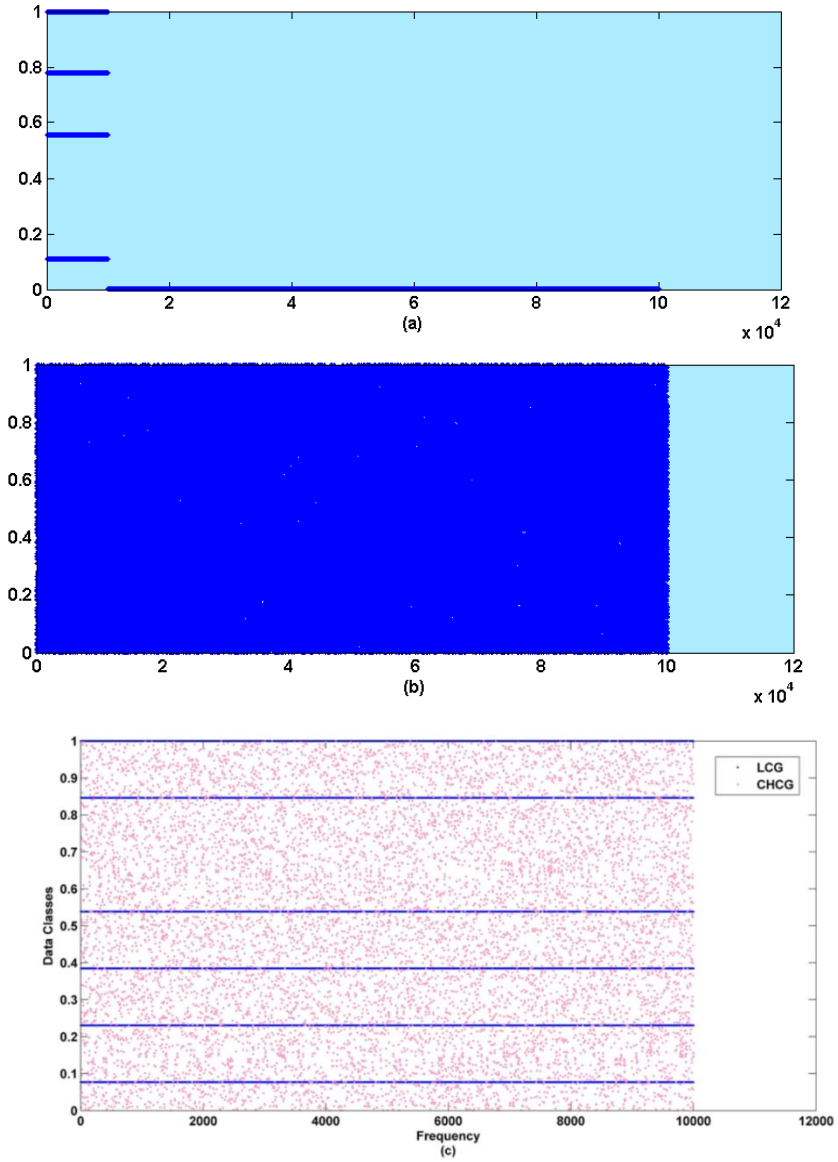



**Figure5:** Frame (a) ,frame (b) and frame(c) respectively, 10000 generated number by LCG and CHCG and Compare of LCG and CHCG

## Conclusion

The paper designs a new secure pseudo-random generator based on the chaotic Henon congruential generator called CHCG. We believe that, the proposed generator is a secure PRNG from the cryptographic point of view. The quality of new the generator, are discussed with testing via the chi-square goodness-of-fit as well as correlation test. Based on these tests which are among the best and commonly applied statistical testers, we claim that CHCG provides high levels of security properties which are vital in cryptographic applications. As mentioned in the context, weakness of regular LCGs is the shortness of the period, in addition to the proposed chaotic map, researches can be focus developing other forms of chaotic maps such as Lorenz attractor, Lozi maps ant etc.

## References


[1] B. Assa, M. Khaled, G. Lakhdar, Implementation of Blum Blum Shub Generator for Message Encryption , international Conference on Control, Engineering Information Technology (CEIT14), 2014.

[2] B.A. Wichmann, I.D. Hill, “An efficient and portable pseudo-random number generator,” Applied Statistics 31, 188-190, 1984.

[3] B. Jansson, Random number generators, PhD thesis, University of Stockholm, 1966.

[4] D.H. Lehmer; Mathematical methods in large-scale computing units; In Proc. 2nd Symposia. On Large-Scale Digital Calculating Machinery; 1951; pages 141-146.

[5] F. Dachselt, W. Schwarz; Chaos and Cryptography; IEEE Transactions on Circuits And Systems-I: Fundamental Theory And Applications; 2001; 48(6):1498-1501.

[6] F. Pareschi, Chaos-Based Random Number Generator: Monotonic implementation, Testing and Applications, PhD Thesis, Bologna University, December 2006-2009.

[7] H. Zeisel, “A remark on algorithm AS183”, Applied Statistics35, 1986.

[8] J. Krhovjak, Cryptographic random and pseudorandom data generators; PhD Thesis, Masaryk University, January 2009.

[9] L. Blum, M. Blum, M. Shub,”Comparison of two pseudo-random number generators”, Proc. CRYPTO 82, 1983, 61-78.

[10] L. Blum, M. Blum and M. Shub, Mike; “A Simple Unpredictable Pseudo-Random Number Generator”; SIAM Journal on Computing 15(2): 364–383. Doi: 10.1137/0215025, 1986.

[11] M. Hoemmen,”Generating random numbers in parallel”, Note, 2007.

[12] P. C. Wu; "Multiplicative, congruential random number generators with multiplier 2k1± 2k2; ACM Trans. Math. Sof.; 23; 1997; 367-374.

[13] P. Ekdahl, On LFSR based Stream Ciphers, PhD Thesis, Lund University, November 2003.

[14] P. L'Ecuyer, R. Simard; "Beware of linear congruential generators with multipliers of the form a =±2q ±2r "; ACM Trans. Math. Sofa.; 25; 1999; 255-265.

[15] P. Junod, "Cryptographic Secure Pseudo-Random Bits Generation: The Blum-Blum-Shub Generator", Note, 1999.

[16] R. A. Fisher, "Frequency distribution of the values of the correlation coefficient in samples of an indefinitely large population", Biometrical, vol10 (1915), 507–521.

[17] ] S. Dilli Babu, M. K. Patnala, Design of a New Cryptography Algorithm using Reseeding-Mixing Pseudo Random Number Generator, International Journal of Innovative Technology and Exploring Engineering, vol2(2013)284-286.

[18] S. Gorenstein, "Testing a random number generator", Comm. Assoc. Cow, March, 1967, 111-118.

[19] S. Parker, O. Chua; Chaos: a tutorial for engineers; Proceedings of the IEEE Transactions; 1995; 75(8): 982–1008.