Indian Journal of Science and Technology, Vol 9(6), DOI: 10.17485/ijst/2016/v9i6/73922, February 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

A Novel Pseudo-Random Number Generator for Cryptographic Applications

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

[1]Department of Statistics, Faculty of Mathematical Sciences, University of Guilan, Rasht, Iran; fathi@guilan.ac.ir
[2]Department of Applied Mathematics, Faculty of Mathematical Sciences, University of Guilan, Rasht, Iran; meisam.mathhom@gmail.com

Abstract

Background: Pseudo random numbers have indispensable role in designing cryptography systems such as key stream in stream ciphers. Efficiency of most crypto systems are in depend on the quality of secret key generated by a pseudo random number generator. **Improvements/Methods:** In the present paper, an efficient pseudo random number generator is presented for cryptographic applications. The algorithm is based on controlling distribution of generated random numbers with the chaotic henon congruential generator. **Statistical Analyses:** Statistical tests and histograms are performed over the proposed generator and the results confirm the improvements over the proposed algorithms. According to the results of statistical tests, proposed algorithms generate pseudo random numbers with acceptable independency and uniformity and random sequences with long enough period. **Applications:** Key streams in stream cipher system can be considered as the most important applications of pseudo random numbers. With this regard proposed generators are statistically proved as proper key stream generator for designing stream cipher systems.

Keywords: Cryptography, Pseudo Random Number Generator, Statistical Test, Stream Cipher, The Henon Map, The Linear Congruential Generator

1. Introduction

Random numbers and random bit generators, RNGs and RBGs, respectively, are a fundamental tool in many deferent areas. The two main fields of application are stochastic simulation and cryptography. In stochastic simulation, RNGs are used for mimicking the behavior of a random variable with a given probability distribution. In cryptography, these generators are employed to produce secret keys, to encrypt messages or to mask the content of certain protocols by combining the content with a random sequence. A further application of cryptographically secure random numbers is the growing area of internet gambling since these games should imitate very closely the distribution properties of their real equivalents and must not be predictable or in uneatable by any adversary[1–3]. A random number generator is an algorithm that, based on an initial seed or by means of continuous input, produces a sequence of numbers or respectively bits. We demand that this sequence appears "random" to any observer.

Independently of whether a RNG is used for stochastic simulation or for cryptographic applications, it has to satisfy certain conditions. First of all the output should imitate the realization of a sequence of independent uniformly distributed random variables. Random variables that are not uniformly distributed can be simulated by applying special transformations on the output of uniformly distributed generators. In this paper we limit our discussion on generators that imitate uniformly distributed variables.

Moreover, a good RNG should work efficiently, which means it should be able to produce a large amount of random numbers in a short period of time. For applications like stochastic simulation, stream ciphers, the masking of protocols or online gambling, huge amounts of random numbers are necessary and thus fast RNGs are required. In addition to the conditions above RNGs for cryptographic

*Author for correspondence

applications must be resistant against attacks, a scenario which is not relevant in stochastic simulation[4,5]. In designing stream cipher, a single pseudo random bit generator, plays the role of 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.

Today's most of practical stream ciphers are based on (LFSR) which makes stream ciphers practical and efficient. But LFSR and therefore stream ciphers are inefficient in implementation[4]. In 1984 Blum and Micali described how to generate a PRBG[6]. In 1999 Pascal Junod discussed and proved the security of The Blum-Blum-Shub Generator from crypto graphical point of view[6], and it was implemented in 2014 with Aissa et al[7,8]. In 2003 Edkal in his PhD thesis discussed design and analysis of stream cipher based on LFSR[4]. In[9] author studied and analyzed Chaos-based random number generators in the University of Bologna. In[10] author studied the relation between cryptography and PRNG in his PhD thesis and in[11] author described the design of a new stream cipher based on PRNG.

In the sequel of the work of authors[12] in which the Linear Congruential Generator is modified by controlling the generated numbers with the CHCG and new generator called (MCHCG) gained suitable properties to be applied in cryptography, in the present paper we proved that implementing controller on the CHCG lead to far better results according to statistical tests.

First, we present the notion of cryptographically secure pseudo random bit enumerators (PRBG), and the Chaotic Henon Congruential Generator (CHCG).

In the second part, the Modified Chaotic Henon Congruential Generator (CHCG), as a very simple and provably secure PRBG, is presented, with all the mathematical background needed to understand it. In the third part, the proof of its security is treated in details.

The paper is organized as follows: In Section 2, pseudo random numbers and Discrete Logistic Map and the Chaotic Henon Congruential Generator (CHCG) are introduced, in Section 3, we present Modified Chaotic Henon Congruential Generator, in Section 4, statistical tests some comparisons are implemented. Finally in section 5 we drive the conclusion.

2. Preliminaries

2.1 Pseudo-Random Number Generator

A pseudo Random Number Generator (RNG) is defined by a structure $(S, \mu, f, U\, g)$ where

- S is a finite set of states.
- μ is a probability distribution on S, called the initial distribution.
- A transition function $f : S \to S$.
- A finite set of output symbols U.
- An output function $g : S \to U$.

Then the generation of random numbers is as follows:

- Generate the initial state (called the seed) s_0 according to μ and compute $u_0 = g(s_0)$
- Iterate for $i = 1, 2, 3, \ldots, s_i = f(s_{i-1}), u_i = g(s_i)$.

Generally, the seed s_0 is determined using the clock machine, and so the random variables $u_0, u_1, \ldots, u_i, \ldots$ seems "real" i.i.d. uniform random variables. The period of a RNG, a key characteristic, is the smallest integer $p \in N$, such that $\forall n \in N, u_{n+p} = u_n$.

Two important statistical properties of the pseudo random number generator are uniformity and independence[13].

The most well-known pseudo random number generator are:

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

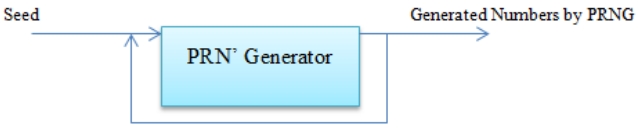

Figure 1. Structure of PRNG.

2.2 The Chaotic Henon Congruential Generator (CHCG)

As mentioned in many contexts[12–14], a problem of Linear Congruential Generator (LCG) has considerably small period. That's why LCG is not sufficient for encryption. In[12], author's to combine the LCG and henon map, until the generated number becomes suitable for application in encryption and named The Chaotic Linear Congruential Generator (CHCG). The author's purpose was to have key stream with high period. As we know, the henon map is sensitive to initial value and has chaotic behavior. Because of this property, to use a henon map when the number, generated by LCG, is same. With this procedure, it is possible to generate suitable key stream. The algorithm of CHCG has presented as following:

Algorithm 1: CHCG

select x_0
 for $i = 1$ *to* n
  $x_i = ax_{i-1} + b \ (mod\ m)$
   for $j = 1$ *to* $i - 1$
    if $x_i == x_j$
     $x_{i+1} = 1 + \alpha x_i + \beta x^2_{i-1}$
    end if
   end for
  end for

3. The Modified Chaotic Henon Congruential Generator (MCHCG)

Although the CHCG has proved suitable for stream cipher systems application but the statistical properties can still be improved. In the present paper, we are going to improve the uniformity as well as independency of the CHCG. This is done by applying a proper controller to regulate the distribution of PRNs generated by CHCG.

The interval in which we are going to produce random numbers is divided into number of subintervals, and at every step of the algorithm a subinterval is chosen randomly and then recall BBS to generate PRN at this interval. The process of randomly choosing subintervals leads to far better results for independency of the CHCG referring to statistical tests presented the next section. On the other hand the subintervals forced controlled CHCG to generate PRN more uniform according to the statistical tests and Histogram plots. Noting that subintervals should be chosen with same chance as much as possible to guarantee the uniformity. The algorithm is presented as follows:

Algorithm 2: Modified CHCG

Start: Choose a seed S.

Preparation: Divide the interval I into some subintervals I_j.

Run: CHCG generator for generating $y_1, y_2, \ldots$

For $j = 1, 2, \ldots$

1. Choose a subinterval randomly and name it I^*.
2. Transform y_j into subinterval I^* put into x_j.

***Return*:** $x_1, x_2, \ldots$ As generated random numbers by MCHCG.

4. Statistical Tests

To ensure the security of an encryption system the cipher must have some properties such as better distribution, long period and high complexity. We used the four suite tests in order to test the randomness of sequences generated by our pseudorandom number generator. The results of statistical tests are shown in tables and figures.

In this section the MCHCG is tested by different statistical tests in order to show the improvements over the CHCG algorithm. In this section, in all tests and figures the CHCG and MCHCG generators run by $x(0) = 0.5$, $a = 75$, $b = 250$, $m = 2^{31} - 1$, $r = 3.999$.

4.1 Histogram Analysis

Figure 2, Figure 3, Figure 4 comparative uniformity of distribution of the generators, MCHCG (b) w.r.t CHCG (a). First, we generate 1000, 10000, 50000 numbers by CHCG and MCLCG. Next, we present, histograms of generating numbers with CHCG and MCHCG. It can be seen that the presented method, i.e., MCHCG is very close to a uniform distribution, compared with CHCG method.

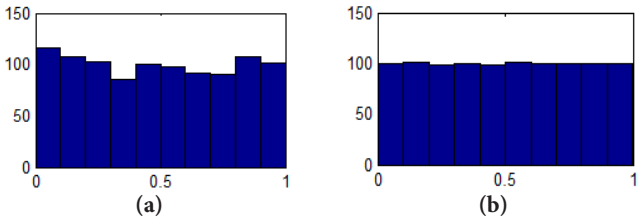

Figure 2. Histogram of frequency. Frame **(a)** and Frame **(b)** respectively, for 1000 generated numbers by CHCG and MCHCG.

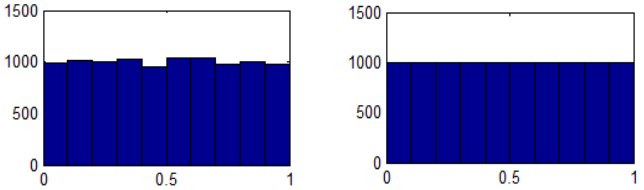

Figure 3. Histogram of frequency. Frame **(a)** and Frame **(b)** respectively, for 10000 generated numbers by CHCG and MCHCG.

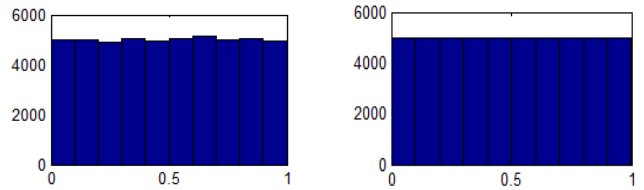

Figure 4. Histogram of frequency. Frame **(a)** and Frame **(b)** respectively, for 50000 generated numbers by CHCG and MCHCG.

4.2 Correlation Test Results

We generated 1000, 10000, 50000, 100000 numbers by CHCG and MCHCG Generators, then the correlation test is applied to these numbers. The results show that, the generated numbers by MCHCG have more independence, according to CHCG and MCHCG is efficient in the sense that to be applied in cryptography. The final result is obtained and present in Table 3.

Now, we generate 1000 numbers by CHCG and MCHCG and present, a scatter plot of generating numbers with CHCG and MCHCG. It can be seen that the CHCG and the MCHCG are independent approximately, but presented method, i.e., MCHCG has more independence comparing with CHCG method.

4.3 Chi-Square Goodness of Fit Test Results

1000, 10000 numbers are generated by CHCG and MCHCG generators and then these numbers are tested via chi-square goodness-of-fit. The final result is obtained and present in Table 1.

4.4 Serial Test Results

The focus of this test is the frequency of all possible overlapping m-bit patterns across the entire sequence. The purpose of this test is to determine whether the number

Table 1. Correlation coefficient results for CHCG and MCHCG

N	CHCG	CHCG
1000	0.0370	–0.0019
10000	0.0386	0.0007
50000	0.0405	0.0003
100000	0.0418	0.0009

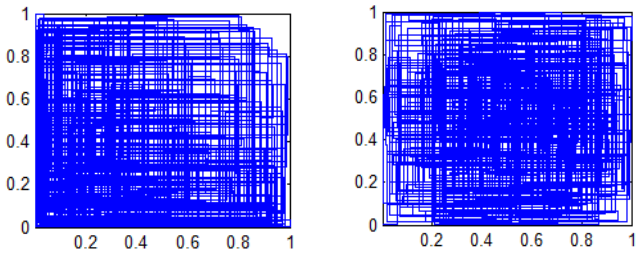

Figure 5. Scatter plot. Frame **(a)** and Frame **(b)** respectively, for 1000 generated numbers by CHCG and MCHCG.

Table 2. Chi-square goodness of fit test results

CHCG					MCHCG				
N	Result	P	H	ϑ	N	Result	P	H	ϑ
1000	8.6587	0.5782	0	9	1000	0.0569	1	0	9
10000	16.9873	0.1680	0	9	10000	0.0359	1	0	9

Table 3. Serial test results

CHCG			MCHCG		
N	Result	P	N	Result	P
1000	pass	0.3724	1000	pass	0.6980
10000	pass	0.6247	10000	pass	0.7682
20000	pass	0.7328	20000	pass	0.8056

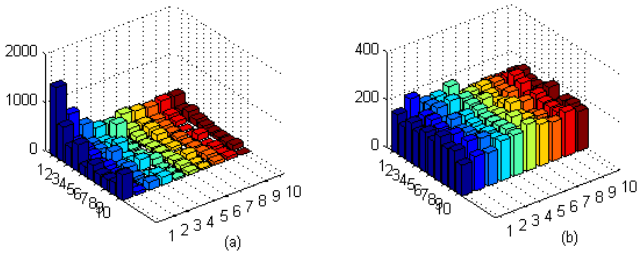

Figure 6. Histogram of frequency in 2D. Frame **(a)** and Frame **(b)** respectively, for 2^{15} – 1 generated numbers by CHCG and MCHCG.

Table 4. Run test results

N	CHCG	MCHCG
2^{15}-1	0.39453	0.60957

of occurrences of the 2^m m-bit overlapping patterns is approximately the same as would be expected for a random sequence. Random sequences have uniformity; that is, every m-bit pattern has the same chance of appearing as every other m-bit pattern[15].

In this section, we run serial test for generating numbers with CLCG and MCHCG and check randomness of the generators. Figure 5, comparative uniformity of distribution of the generators, MCHCG (b) w.r.t CHCG (a) in 2D. It can be seen that the presented method, i.e., MCHCG is very good, compared with CHCG method.

4.5 Run Test Results

The focus of this test is the total number of runs in the sequence, where a run is an uninterrupted sequence of identical bits. A run of length k consists of exactly k identical bits and is bounded before and after with a bit of the opposite value. The purpose of the runs test is to determine whether the number of runs of ones and zeros of various lengths is as expected for a random sequence. In particular, this test determines whether the oscillation between such zeros and ones is too fast or too slow[15]. We check MCHCG and CHCG methods with the run test and results present in Table 4.

5. Conclusions

In the present paper, we introduced a modification of CHCG algorithm to increase the independence and uniformity of the CHCG algorithm. This applies a controller for proper scattering of generating random numbers. In order to show the improvements, we performed statistical tests on CHCG and MCHCG and the results corned the improvements. The statistical tests and histograms showed to enhance statistical properties of proposed modified generator clearly. The proposed MCHCG is targeted stream crypto systems which are context based. The CHCG algorithm is secure algorithm that attracted researchers in the field of cryptography, but it still can be improved for deferent purposes. As an example, it can speed up for efficient implementation.

6. References

1. Dutang C, Wuertz D. A note on random number generation. Overview of Random Generation Algoritms. 2009 Sep; 1–29.
2. Ecuyer PL. Random numbers for simulation. Communications of the ACM. 1990; 33:5–98.
3. Blum L, Blum M, Shub M. A simple unpredictable pseudo-random number generator. SIAM Journal on Computing. 1986 May; 15(2):364–3.
4. Kenny C. Random number generators- An evaluation and comparison of random.org and some commonly used generators'; Trinity College Dublin; 2005 Apr. p. 1–107.
5. Baptista MS. Cryptography with chaos. Physics Letters. 1998 Jan; 240(1-2):50–4.
6. Blum L, Blum M, Mike SM. A simple unpredictable pseudo random number generator. SIAM Journal on Computing. 1996; 15(2):364–83.
7. Alligood KT, Sauer TD, Yorke AJ. Chaos: An introduction to dynamical systems. New York: Springer Verlag; 1996.
8. Assa B, Khaled M, Lakhdar G. Implementation of blum blum shub generator for message encryption. International Conference on Control, Engineering and Information Technology (CEIT14); Bonn-Germany. 2014. p.118–23.
9. Pareschi F, Chaos-based random number generator: monotonic implementation, testing and application [PhD thesis]. Bologna University; 2006-2009 Dec.
10. Krhovjak J. Cryptographic random and pseudorandom data generators [PhD thesis]. Masaryk University; 2009 Jan.
11. Babu DS, Patnala MK. Design of a new cryptography algorithm using reseeding-mixing pseudo random number generator. International Journal of Innovative Technology and Exploring Engineering. 2013; 2(5):284–6.
12. Vajargah BF, Asghari R. A pseudo random number generator based on chaotic henon map (CHCG). IJMEC. 2015; 5(15):2026–37.
13. Ekdahl P. On LFSR based Stream Ciphers. [PhD thesis]. Lund University; 2003.
14. Panneton F, Ecuyer PL, Matsumoto M. Improved long-period generators based on linear recurrences modulo 2. ACM Transom Mathematical Software. 2006; 32(1):1–16.
15. Rukhin A, Soto J, Nechvatal J, Smid M, Barker E, Leigh S, Levenson M, Vangel M, Banks D, Heckert A, Dray J, Vo S. A statistical test suite for random and pseudorandom number generators for cryptographic applications. National Institute of Standards and Technology. Special Publication 800-22 Revision 1; 2008.