2009 International Conference on Computer Engineering and Applications
IPCSIT vol.2 (2011) © (2011) IACSIT Press, Singapore

Randomness Analysis of Pseudorandom Bit Sequences

Rashidah Kadir [+] and Mohd Aizaini Maarof

Faculty of Computer Science and Information Systems, Universiti Teknologi Malaysia, Johor, MALAYSIA

Abstract. Many users applied built-in random generator for their cryptography applications which is simple and fast. However, the randomness of generated pseudorandom numbers (PRNs) is under questioned whether it can support the reliable security in secure communication. In this paper, we examined two kinds of pseudorandom bit sequence (PRBS); conventional PRBS and chaos-based PRBS. Linear Congruential method, Marsaglia's Ziggurat algorithm and Mersenne Twister method are used to generate the former, and the latter is produced using chebychev map, logistic map and sawtooth-like map. To verify the randomness of the PRBS, the standard statistical test FIPS140-2 is performed. Overall four tests involved namely monobit, poker, runs and long runs. The results have been compared and analysed between conventional PRBS and chaos-based PRBS on their reliability as encryption key.

Keywords: random number generation, chaos, PRNG, PRBS, chaos-based PRNG, randomness test

1. Introduction

Random bit generators have been widely employed in many electronic equipment [1] and several applications such as statistical sampling, computer simulation and cryptography [2,3]. In this paper, we concentrate on random number generation in cryptography which required for producing key. The security level of cryptographic system depends on the random number generators features of unpredictable and unlimited period [4]. Good cryptography requires good random numbers. Inadequate source of randomness can compromise the strongest cryptographic protocol and application. There are two kinds of the random number generator (RNG): truly random (TRNG) and pseudorandom (PRNG). The TRNGs are very unpredictable but difficult to handle because it's too sensitive to the changing environment. As a result, TRNGs are uncontrollable and not suitable for those applications demanded strongly stable randomness quality. PRNG is widely used method improving the controllability of random number. The PRNG can be implemented through software or hardware. For hardware implementation, LFSR is the most popular [1,5-8]. It advantages depend on simple implementation and high speed performance. However, it has poor security in terms of violability. For software implementation, most of users applied built-in random generator for their applications which is simple and fast. However, the randomness of generated pseudorandom numbers is claimed not secure enough to be used for cryptography. Over the past decades, chaos-based pseudorandom generator is seemed promising. Due to some interesting characteristics of chaos, such as sensitivity to initial conditions, ergodicity, unstable periodic orbits with long periods and random-like behavior, chaotic systems may appear to be good sources of randomness.

This paper is organized as follows. Section 2 presents the pseudorandom number generator related works. Section 3 describes the process of pseudorandom bit sequence generation and the randomness test. In Section 4, the randomness test is performed on PRBS generated and the results are analyzed in detail. Finally, Section 5 some conclusion are drawn and future work is discussed.

[+] Corresponding author. Tel.: + 607-5532244; fax: +607-5565044.
E-mail address: rashidah@utm.my

2. Pseudorandom Number Generator

Many methods have been used for generating pseudorandom sequence in many applications. However, most of the pseudorandom number generators (PRNGs) are claimed insecure. As pointed out in [9], a good pseudorandom bit sequence (PRBS) should at least satisfy the following cryptographic properties: 1) balance on {0, 1}; 2) high linear complexity approximately to half of sequence period; 3) long period; 4) d-like auto-correlation, nearly zero cross-correlation; 5) passing empirical statistical test (or randomness test).

2.1. Conventional PRNG

A Linear Congruential Generator (LCG) represents one of the oldest and best-known pseudorandom number generators which is easy to understand, and they are easily implemented and fast. It is the basis for many of the random number generators in use today. Starting with an initial value, x_0 the linear congruential method uses the following formula:

$$x_{n+1} = (ax_n + c) \bmod m \tag{1}$$

Most common pseudo random number generators (PRNG) implemented in standard libraries such as C/C++, use the LCG but with different parameters. Table 1 shows the list of parameters of LCGs in various compilers. However, LCGs are not recommended to be used in cryptography, nor any other purposes which require higher degrees of randomness [10-12].

TABLE I. THE PARAMETER OF LCGS IN COMMON USE

Compiler	Parameter
rand() in Visual C++,	a=214013,c=2531011, m=4294967296
rand() in Matlab	a=16807, c=0,m= 2147483647
Knuth [13]	a= 16807; c=0; m=2147483647;

Another pseudorandom generator is Mersenne Twister method that based on a matrix linear recurrence over a finite binary field F2, It developed by Makoto Matsumoto and Takuji Nishimura [14]. If higher quality random numbers are needed, and sufficient memory is available, then the Mersenne Twister algorithm is a preferred choice. A common Mersenne twister implementation, interestingly enough, uses an LCG to generate seed data. A Marsaglia Ziggurat algorithm is specifically designed to produce floating-point values. The ziggurat algorithm is more complex to implement and requires precomputed tables, so it is best used when large numbers of random values are required [15].

2.2. Chaos-based PRNG

Easy realization and tight relationship with cryptography makes chaos as a suitable choice for pseudorandom number generators in cryptography area such as multimedia encryption. Pseudorandom numbers are produced using simple mathematical equation that iteratively run called chaotic map. Many chaos-based PRBG employed chaotic map such as logistic [16-18], chebyshev [19], sawtooth-like [20], H´enon [21], skew tent [22-23]; have been widely used for generation of the bit sequences.

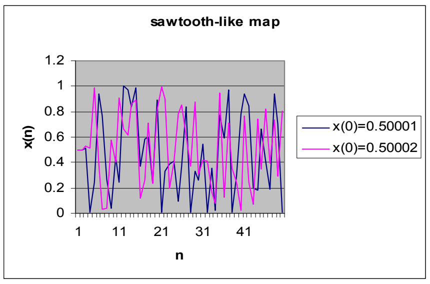

Fig. 1: Two evolution curves of the sawtooth-like map with slight difference in the initial condition (0.50001 and 0.50002).

Compared to conventional PRBG, chaotic systems have a very sensitive dependence on their initial. This sensitive dependence is shown in Fig. 1. After a few iterations, the two iterative sequences are completely different. Thus, there are enormous numbers of chaotic pseudorandom sequences that satisfy the requirements of cryptography [24]. Parameters and initial value of chaotic maps also can be used as an encryption key.

3. Empirical Test

For the randomness analysis of a pseudorandom number generator, we adopted the strategy illustrated in Fig. 2.

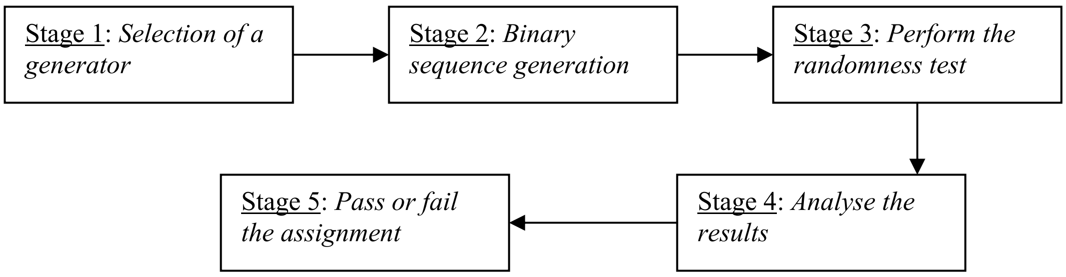
Stage 1: *Selection of a generator*
Stage 2: *Binary sequence generation*
Stage 3: *Perform the randomness test*
Stage 4: *Analyse the results*
Stage 5: *Pass or fail the assignment*

Fig. 2 : The strategies for the randomness analysis

3.1. Generating PRBS

Selection of PRBG and binary sequence generation are two processes involved in PRBS generation. In this paper we select three PRBG for conventional PRBS and chaotic PRBS. For conventional PRBS, we implement Linear Congruential method, Mersenne Twister method and Marsaglia's Ziggurat algorithm which have been implemented using built-in function *rand()* in Matlab and Visual C++. As mentioned earlier, we produce chaotic-PRBS based on three chaotic maps. They are chebychev map, logistic map and sawtooth-like map.

The logistic map can be denoted by the following equation: $x_{n+1} = \mu x_n(1-x_n)$ (2)

where, x_0 is the initial state, and $x_0 \in [0,1)$; μ is a cryptographic key, x_n is a random number generated by the random number generator, and x_n+1 is an iterated result. Here, we select $\mu = 4$, where the logistic map is chaotic [17, 21].

The sawtooth-like map can be expressed as equation here: $x_{n+1} = cx_n \bmod 1$ (3)

The chebychev map which is employed as follow: $x_{n+1} = \cos(4\arccos(x_n)) \quad -1 \le x_n \le 1$ (4)

After n iterations, a pseudorandom number x_n is obtained. In order to construct a bit sequence, the pseudorandom number generated is converted to '0' and '1' using the following tossing coin formula:

$$b_n = \begin{cases} 0 & if x_n < 0.5 \\ 1 & if x_n \ge 0.5 \end{cases} \tag{5}$$

Finally a bit sequence B = $\{b_1, b_2, b_3..., b_n\}$ is produced.

3.2. The Randomness Test

To verify the randomness of pseudorandom bit sequences, we apply the standard randomness test FIPS 140-2 [25]. Any PRBS can be claimed as a good PRBS if passed all the tests specified. For analysis purposes, the randomness test has been performed which consist of four tests. A single bit sequence of 20,000 consecutive bits of output from the generator is subjected to each of the following tests:

- *The Monobit Test*: In this test, the number of ones is counted and denotes this quantity by x. The test is considered pass if $9,654 < x < 10,346$.
- *The Poker Test*: We divide 20,000 bit sequence into 5,000 contiguous 4 bit segments. The number of occurrences of each of the 16 possible 4 bit values is counted and stored. Denote f(i) as the number of each 4 bit value i where $0 < i < 15$. Then, we evaluate the following:

  $$x = (16/5000) * (\sum[f(i)]^2)-5000 \tag{6}$$

  The test is passed if $1.03 < x < 57.4$.
- *The Runs Test*. A run is defined as a maximal sequence of consecutive bits of either all ones or all zeros, which is part of the 20,000 bit stream. The incidences of runs of all lengths should be counted and stored.

The test is passed if the number of runs that occur is each within the specified interval as stated in Table II.

TABLE II. VALIDITY INTERVAL OF THE RUN TEST

Run length	Required Interval
1	*2,267-2,733*
2	*1,079-1,421*
3	*502-748*
4	*223-402*
5	*90-223*
≥ 6	*90-223*

- *The Long Runs Test*: A long run is a run that formed by '0' or '1' with length 34 or more. The test is passed if there are no long runs on the 20,000 consecutive bits.

4. Result and Analysis

In this section, we present the testing result of the conventional PRBS and chaotic-based PRBS. Table III shows the output of the randomness test that performed on conventional PRBS. Two different results of Linear congruential method are obtained. As mentioned earlier, it relies to the parameter setting of built-in PRBG implemented in the compiler Visual C++ and Matlab respectively. However, both passed the randomness test followed by Twister and Marsaglia method.

TABLE III. RANDOMNESS TEST OF THE CONVENTIONAL PRBS

Method	Monobit	Poker	Runs						Long runs
			r=1	*r=2*	*r=3*	*r=4*	*r=5*	*r>=6*	
LCG	9831	16.10	2536	1240	626	312	144	136	None
LCG	9978	14.96	2464	1232	631	331	138	162	None
Twister	10129	10.84	2549	1207	628	330	169	161	None
Marsaglia	9887	16.36	2516	1269	630	297	156	144	None

TABLE IV. RANDOMNESS TEST OF THE CHAOS-BASED PRBS

Chaotic map	Monobit	Poker	Runs						Long runs
			r=1	*r=2*	*r=3*	*r=4*	*r=5*	*r>=6*	
Chebychev	*6685*	*2429.70*	*1666*	*1278*	*824*	*530*	*315*	*405*	None
Logistic	9867	22.33	2549	1239	625	300	139	153	None
Sawtooth-like	9853	30.11	2391	1210	601	303	163	173	None

As comparison, the same randomness test was repeated on the chaos-based PRBS using chebychev map, logistic map and sawtooth-like map. The results of statistical test are depicted in Table IV. Unexpectedly, the PRBS based on chebychev map does not satisfy three tests except the long run test. The results of the long run test show that there are NO long runs in all four pseudorandom numbers. Thus, all methods and functions have passed the Long runs test.

5. Conclusion

All the pseudorandom sequences generated in this paper passed the randomness test, FIPS PUB 140-2 test except PRBS generated using chebychev map. Thus, the analysis results show that conventional PRBG and chaotic PRBG able to generate a good pseudorandom bit sequence for encryption key. In addition, chaotic PRBGs are better for multiple key generations which slightly difference in their initial value will results completely difference set of pseudorandom numbers. A further study can be carried out on other 2-D, 3-D and discrete chaotic-based pseudorandom bit sequence that possible to generate secure PRBS.

6. Acknowledgements

This research is partially supported by Ministry of Higher Education, Malaysia. Our thanks also to MOSTI and Universiti Teknologi Malaysia.

7. References

[1] R. Mita, G. Palumbo, S. Pennisi, and M. Poli. A novel pseudo random bit generator for cryptography applications. *Proc. Int. Conf. on Electronics, Circuits and Systems*. 2002, vol. 2, pp. 489-492.

[2] W. Stallings. *Cryptography and Network Security Principles and Practices*. Third Edition, Prentice Hall, 2003.

[3] N. Ruggeri. Principles of Pseudo-Random Number Generation in Cryptography, VIGRE Program, The University of Chicago, August 2006. Available online at http://www.math.uchicago.edu/~may/ VIGRE/VIGRE2006/PAPERS/Ruggeri.pdf

[4] Y. L. Chung, , S. C Jiung, Y. C. Tsin. A chaos-based pseudo random number generator using timing-based reseeding method. *Proc. IEEE International Symposium on Circuits and Systems, ISCAS 2006.*, 2006, vol. no. 4, pp.-3280.

[5] P. Alfke. Efficient Shift Register, LFSR Counters, and Long Pseudo-Random Sequence Generators. Xilinx Application Note, 1996.

[6] D. Kay, S. Mourad. Controllable LFSR for BIST. *Proc. Intrumentation and Measurement Technology Conference*, 2000, pp. 223-228.

[7] H. L.Bin, H. S.Shao, W. W.Cheng. A fast signature computation algorithm for LFSR and MISR," *IEEE Trans. CAD of Int. Circuits and Systems.* 2000, **19** (9): 1031-1040.

[8] M. Y. Seung, O. J. Seoung, , M. K. Sung. Low cost and high efficiency BIST scheme with 2-level LFSR and ATPT. *Proc. IEEE Int. Symp. Circuits and Systems*. 2001, vol. 4, pp. 1-4.

[9] X. Wang, J. Zhang, Y. Fan and W. Zhang. Chaotic Pseudorandom Bit Generator Using n-dimensional Nonlinear Digital Filter. *Proc. Int. Conf. on Communication Technology(ICCT '06)*. 2006, pp. 1-4, 27-30.

[10] Haldir. How to crack a Linear Congruential Generator. The Reverse Engineering Team, December 2004. Available online at http://www.reteam.org/papers/ e59.pdf

[11] B. Schneier. *Applied Cryptography – Protocols, algorithms, and souce code in C*. John Wiley & Sons, Inc., New York, second edition, 1996.

[12] C. Moler, *Numerical Computing with MATLAB,Electronic* edition: The MathWorks, Inc., Natick, MA, 2004. Available online at http://www.mathworks.com/moler. Print edition: SIAM, Philadelphia, 2004.

[13] D. E. Knuth, *Seminumerical Algorithms*, 3rd edition, vol. 2 of: The Art of Computer Programming, Addison-Wesley, Reading, MA (1998).

[14] M. Matsumoto and T. Nishimura, Mersenne Twister Home Page, Available at http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt .html

[15] G. Marsaglia and W. W. Tsang, The ziggurat method for generating random variables, *Journal of Statistical Software*. 2000, **5**: 1-7. Available online at http://www.jstatsoft.org/v05/i08

[16] W. Zhang, J. Peng, H. Q. Yang. A digital image encryption scheme based on the hybrid of cellular neural network and logistic map. *LNCS 3497*, 2005, pp. 860 -867.

[17] N. S. Philip and K. B. Joseph. Chaos for stream cipher. Cryptography and Security, arXiv:nLin.CD/0102012 v1,16 Feb.2001. [Online]. Available online at http://arxiv.Org/abs/cs.CR/0102012.

[18] S. C. Phatak and S. S. Rao. Logistic map: a possible random-number generator. *Physical Review E*. 1995, 51 (4): 3670-3678.

[19] X. F. Liao, X. M. Li, J. Peng, et al., A digital secure image communication scheme based on the chaotic Chebyshev map, *Int. J. Commun. Syst*em. 2004, **17** (5): 437-445.

[20] H. Jian, Y. Mao and Z. Wang. A Novel Chaos-Based Video Encryption Algorithm. *Proc. of Trim Size: 9in x 6in*, 2004.

[21] D. Erdmann and S. Murphy, "H´enon stream cipher," *Electron. Letter.* 1992, **28** (9): 893-895.

[22] S. Lian, G. Chen, A. Cheung, Z. Wang. A chaotic-neural-network-based-encryption algorithm for JPEG2000 encoded images. *Proc. ISNN 2004-II. LNCS 3174* , 2004, pp. 627–632.

[23] M. K. Khan and J. S. Zhang. Investigation on pseudorandom properties of chaotic stream cipher. *Proc. IEEE International Conference on Engineering of Intelligent Systems*, 2006, pp. 1-5.

[24] J. Peng, X. F. Liao, and Z. Wu. Digital image secure communication using Chebyshev map sequences. *Proc. IEEE Int. Conference on Comm., Circuits and Systems and West Sino Expositions (ICCCS'2002)*, 2002, pp. 492–496.

[25] National Institute of Standard and Technology, Security Requirements for Cryptographic Modules", FIPS PUB 140-2, May 2001.