A NEW HYBRID PSEUDORANDOM NUMBER GENERATOR

Adrian-Viorel DIACONU[1,2], Ion SIMA, Valeriu Manuel IONESCU
[1]Lumina – The University of South – East Europe, IT&C Department, Bucharest 021187, Romania
[2]University Politehnica of Bucharest, ETTI Faculty, Bucharest 061071, Romania
[1]adrian.diaconu@lumina.org

Keywords: pseudorandom number generator, logistic map, RANROT generators, NIST Statistical Test Suite for Random and Pseudorandom Number Generators

Abstract: *The pseudorandom number generators are an essential component of any cryptosystem because the security of many cryptographic applications depends on the generation of good pseudorandom binary sequences. In the last decade, the development of pseudorandom numbers generator based on chaotic maps has been widely study. In this sense, we propose a new pseudorandom generator which has a hybrid structure, i.e., based on classical RANROT generators of type B3 and the chaotic logistic map. Using NIST Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications we prove that the proposed generator has very good cryptographic qualities.*

1. INTRODUCTION

The core of any cryptographic application resides within the process of generating arbitrary numbers, apriori unknown and unpredictable. Keystreams, initialization vectors and signature protocols used in encryption schemes are all based on generation of sequences of numbers that approximates quality of random variables specific to stochastic process, known in the literature as pseudorandom number generators (PRNG).

Designing of reliable PRNGs remains an open problem in cryptography. Some *de facto* standards, regarded as secure in the past, have recently failed [1 – 4]; other generators (*e.g.*, the BBS generator [5], one of the few PRNGs with proven security) are rarely used due to their slowness. One of most studied and used class of PRNGs is one of Lagged Fibonacci Generators (LFGs), which are based on a generalization of the Fibonacci sequence [6 – 8]. Still, these types of generators have some drawbacks [9], *e.g.*, dependence from the values used as seed and the correlation of generated values.

In order to improve those drawbacks some perturbation techniques of Fibonacci recurrence have been proposed [6], [8].

In last decade, one of the most promising directions for research and development of new PRNGs is the usage of chaotic maps [10 – 12]. Although chaos implies an unpredictable time behavior of a system, its dynamics, whose evolution seems to be random, can be expressed by a deterministic rule. This is one of the main properties of chaotic dynamical systems which have encouraged the idea to design new PRNGs and also to develop some robust encryption schemes based on chaos. In addition, some specific properties of chaotic dynamical systems, *e.g.*, sensitivity to initial conditions, ergodicity and mixing properties can be connected with notions of confusion and diffusion, introduced by Shannon [13].

In this paper we propose a new PRNG, which has a hybrid structure, based on three classical RANROT generators of type B3, along with four chaotic logistic maps. Using the NIST statistical test suite and other specific tools [14], we proved that the proposed generator has very good cryptographic qualities.

In section 2 we present the RANROT generators, in section 3 logistic map is described, in section 4 the proposed PRNG is presented, in section 5 PRNG's performances are assessed, while the last section is devoted to conclusions.

2. THE RANROT GENERATORS

A pseudorandom number generator is a deterministic algorithm defined by a finite space of states S, a transition function $f: S \rightarrow S$ and an initial state s_0, called the seed of the generator. The transition between states of the generator is based on a recurrence:

$$s_i = f(s_{i-1}), \qquad i = 1,2,3, \ldots \tag{1}$$

and current state s_i are converted to a real number from $[0,1]$ by the relation:

$$U_i = g(s_i) \tag{2}$$

where $g: S \rightarrow (0,1)$ is the output map.

Pseudorandom numbers generators, *abbr.* PRNG, which are based on Fibonacci recurrence, have become more and more popular, both in serial and parallel applications, in the last years. This is because it is easy to implement it and responds quite well to randomness tests. These generators, named in the specialized literature Lagged Fibonacci Generators (*abbr.*, LFG), have been widely studied. George Marsaglia has made an extensive study on these generators in order to determine their maximum period and how to select the parameters for the initial state. The general form of a Lagged Fibonacci Generator is $LFG[r, s, m, \circ, x_t\{0, \ldots, r-1\}]$, where $r > s > 0$ are the lag parameters, $\circ$ is a binary operation, m is the modulo and $x_t\{0, \ldots, r-1\}$ is a sequence of r initial values (seed). For $n \geq r$ the recurrence may be defined as follows:

$$x_n = x_{n-r} \circ x_{n-s} \pmod{m} \tag{3}$$

The regular operations are addition, subtraction, multiplication (modulo m) and XOR if m is a power of two. Usually, $m = 2^b$ where b is the length of the memory word [3]. The maximum repetition period can be achieved only if certain conditions are satisfied by the initial values and by the lag parameters. Brent [15] has recently showed that if $m = 2^b$ and the polynomial $x^r + x^s + 1$ is irreducible primitive over $GF(2)$, the maximum repetition period is $p = 2^{b-1}(2^r - 1)$.

In practice, the LFG generators still have some lacks. First LFGs' drawback is that their output is sensitive to initial conditions. Another problem is related to their random feature, being well known the fact that LFGs don't pass certain randomness statistic tests, *e.g.*, Birthday Spacing test from Diehard standard randomness testing battery proposed by Marsaglia [16]. Moreover, due to designing of binary operations, there is a leak of information from the least significant bits to the most significant ones, information which does not transfer in contrary [9].

To eliminate/improve these shortcomings, various options have been tried, *e.g.*, transport to the most significant bit, which improves length of the period, but not the random character. In 1997, danish scholar Agner Fog proposes a new class of PRNGs based on Fibonacci recurrence. Known as the RANROT generators, they realize a disturbance of the insignificant bits with the bit rotation operations [6]. Initially proposed for Monte Carlo applications, RANROT generators can be successfully used in cryptography if the lag and rotation parameters are secret. Unlike conventional generators, RANROT generators have random lengths of the cycle, *i.e.*, inclusive their period is random. Also, disturbance created by the rotations of the bits greatly increases the period of the LFGs.

Several types of generators RANROT are presented below (4). In the case of the type A generator the bits are rotated after addition, while in the case of type B the bits are rotated turned before addition. There is a case when there are more than two terms, as the type B3 below, and there is a case in which parts of the bits strings can be rotated, separately, as the generator of type W.

$$\begin{aligned}
\text{Type A: } & x_n = \left((x_{n-j} + x_{n-k}) \bmod 2^b\right) \gg p \\
\text{Type B: } & x_n = \left((x_{n-j} \gg p_1) + (x_{n-k} \gg p_2)\right) \bmod 2^l \\
\text{Type B3: } & x_n = \left((x_{n-i} \gg p_1) + (x_{n-j} \gg p_2) + \right. \\
& \left. + (x_{n-k} \gg p_3)\right) \bmod 2^b \\
\text{Type W: } & z_n = \left((y_{n-j} \gg p_3) + (y_{n-k} \gg p_1)\right) \bmod 2^{\frac{l}{2}} \\
& y_n = \left((z_{n-j} \gg p_4) + (z_{n-k} \gg p_2)\right) \bmod 2^{\frac{b}{2}} \\
& x_n = y_n + z_n \cdot 2^{\frac{b}{2}}
\end{aligned} \tag{4}$$

Each x_n represents an unsigned integer represented on b bits, y_n and z_n are unsigned integers represented on $b/2$ bits, *resp.*, i, j and k are unsigned integers subject to the following rule: $0 < i < j < k$. The operation $\gg$ represents

a circular rotation of the bits to the right. The values are calculated in a buffer with k elements, named S_n. Initial state is $S_1 = (x_1, x_2, \dots, x_n)$ and the transition from one state to another is accomplished by a shift to the left of the form $(x_{n-k}, \dots, x_{n-1}) \rightarrow (x_{n-k+1}, \dots, x_n)$, where x_n is calculated according to the used recurrence. The seed of the generator W is given by the lag parameters j and k, circular rotation parameters p_1, p_2, p_3, p_4, the length of the buffer k and the initial state $S_1 = (x_1, x_2, \dots, x_n)$.

In order to obtain the best performance, the parameters must to fulfill few rules:

- lag parameters, *i.e.*, j and k, must be prime numbers, namely between them and their odd difference;
- to eliminate possible symmetries in the generated string, circular rotation parameters *i.e.*, p_1, p_2, p_3, p_4, must be distinct, non-zero integers, prime with the length of the buffer k, *resp.*, with length of the memory word b;
- the results are poor if the rotation parameters p_1, p_2, p_3, p_4 are too small or too close to the value $b/2$, so the circular rotation parameters must be generated as random numbers from the interval $[0, b/2)$.

The statistical tests presented in [17], [18] shows that perturbation increases dramatically the randomness and the period of the generator.

3. THE LOGISTIC MAP

The logistic map is a very simple mathematical model often used to describe the growth of biological populations. In 1976 May [19] showed that this simple model shows bewildering complex behavior. Because of its mathematical simplicity, this model continues to be useful in chaos theory as well as application of chaos in cryptography [20 – 25]. The logistic map is given by:

$$x_{n+1} = r x_n (1 - x_n) \tag{5}$$

where $r \in (0,4]$ is the control parameter and $x_0 \in [0,1]$ is the initial condition.

The time long behavior of logistic map depends by the control parameter r and by the initial condition. For $r < 1$, the origin $x_0 = 0$ is a fixed point, but while the parameter's value increase ($1 < r < 3$), the origin lose its stability and another fixed point occurs, $x_1 = \frac{r-1}{r}$. For r between 3 and 3.57 the fixed point also lose its stability and an attractor with period 2 occurs. Starting from $r \cong 3.57$ the behavior of logistic map becomes complex, and for parameter's values close to 4 the map is in a chaotic regime. This can be seen from the bifurcation diagram plotted in Fig.1.

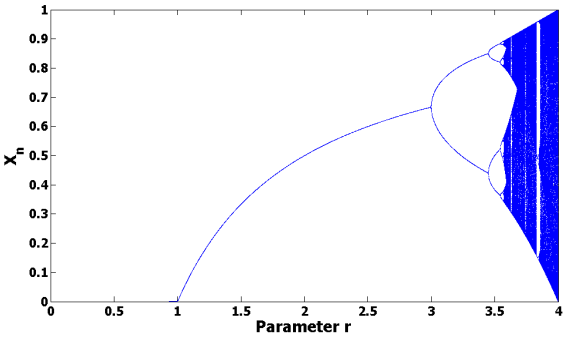

Fig. 1 Bifurcation diagram of the logistic map

Another powerful instrument that can be used to determine the sensitivity of a dynamical system to the initial conditions is the *Lyapunov exponent*, defined as it follows [26,27]:

$$\lambda = \lim_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n-1} \ln \left| f'(x_i) \right| \tag{6}$$

A dynamical system is in a chaotic regime if its Lyapunov exponent is positive [26], [27]. In Fig. 2, Lyapunov exponent's values, *i.e.*, for the logistic map, which were numerically calculated using Wolf's algorithm [28], are plotted and it can be observed that for values of the parameter $r \geq 3.57$ the logistic map is chaotic.

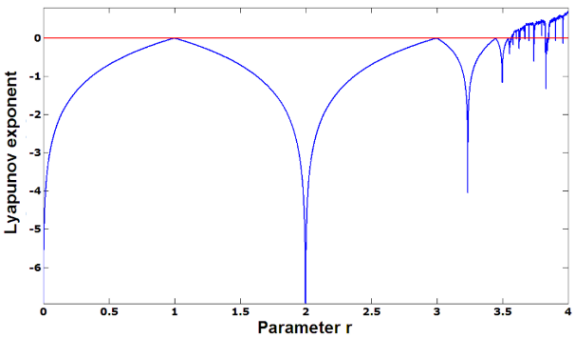

Fig. 2 Logistic map's Lyapunov exponent

4. THE PROPOSED PRNG

As we mentioned at the end of Section 3, performances of RANROT generators depend on the selection of lag and rotation parameters. Since a proper selection of these parameters requires compliance with several conditions simultaneously, their generation process can be quite complicated and, moreover, the set of all their possible values can be very small, which decreases the security of the generator. In order to ameliorate this drawback of the RANROT generators, we propose the usage of fixed, but random, lag parameters, along with random rotation parameters, which are changed in each step of the generation process.

The proposed PRNG uses three coupled RANROT generators of type B3, denoted by x_n, y_n and z_n. The output t_n combines the outputs of the generators x_n, y_n and z_n, using a formula similarly to equation of a RANROT generator of type A, as it follows:

$$\begin{aligned} x_n &= \left(\left(x_{n-i_1} \text{ rotr } p_n^{1,1}\right) + \left(x_{n-j_1} \text{ rotr } p_n^{1,2}\right) + \right. \\ &\quad \left. + \left(x_{n-k_1} \text{ rotr } p_n^{1,3}\right)\right) \bmod 2^b \\ y_n &= \left(\left(y_{n-i_2} \text{ rotr } p_n^{2,1}\right) + \left(y_{n-j_2} \text{ rotr } p_n^{2,2}\right) + \right. \\ &\quad \left. + \left(y_{n-k_2} \text{ rotr } p_n^{2,3}\right)\right) \bmod 2^b \\ z_n &= \left(\left(z_{n-i_3} \text{ rotr } p_n^{3,1}\right) + \left(z_{n-j_3} \text{ rotr } p_n^{3,2}\right) + \right. \\ &\quad \left. + \left(z_{n-k_3} \text{ rotr } p_n^{3,3}\right)\right) \bmod 2^b \\ t_n &= \left((x_n + y_n + z_n) \bmod 2^b\right) \text{rotr } p_n^4 \end{aligned} \quad (7)$$

Proposed PRNG also uses four logistic maps f_1, f_2, f_3 and f_4, each with its own control parameter, *i.e.*, r_1, r_2, r_3 and r_4 (randomly chosen from [3.9,4] interval), *resp.*, each with its own initial conditions x_0^1, x_0^2, x_0^3 and x_0^4 (randomly chosen from the interval (0,1)), which ensure a chaotic regime for all the four logistic maps. Moreover, a good chaotic regime is assured pre-iterating all the four maps. The number of pre-iterations m_1, m_2, m_3 and m_4 of each map are randomly chosen, but greater than 100. The real values of the logistic maps are discretized multiplying them by 10^{15} and taking the integer part of the result modulo 2^b. The logistic maps f_1, f_2, f_3 and f_4 are used in the proposed PRNG as it follows:

- each logistic map f_l is pre-iterated by m_l times ($l = \overline{1,4}$);
- the next 3 values of each logistic map f_l become i_l, j_l and k_l ($l = \overline{1,3}$);
- the next $\max\{i_l, j_l, k_l\}$ values of each logistic map f_l are used to initialize the RANROT generators of type B3 denoted by x_n, y_n and z_n ($l = \overline{1,3}$);
- in each step of the generation process, three consecutive values, for each logistic map, *i.e.*, f_l ($l = \overline{1,3}$), become the current rotation parameters $p_n^{l,1}$, $p_n^{l,2}$ and $p_n^{l,3}$, while the value of the logistic map f_4 becomes p_n^4.

The seed of the proposed PRNG consists from the 8 real values $r_1, r_2, r_3, r_4, x_0^1, x_0^2, x_0^3$ and x_0^4 and from the 4 unsigned integers m_1, m_2, m_3 and m_4.

If the implementation of the proposed PRNG is done using a programming language that complies with *IEEE Standard 754-2008*, then it's recommended to use the double data type, *i.e.*, which stores real numbers on 8 bytes with an accuracy of 15 decimals, *resp.*, unsigned integers, which are stored using 4 bytes. In this case, the length of the seed, which becomes secret key, if the PRNG is to be used in some cryptographic applications, will be 2560 bits.

Obviously, size of subsequent key space will be equal to 2^{2560}, large enough to prevent guessing the secret key in a reasonable time, using brute force.

5. RANDOMNESS ANALYSIS

In the third step of performances' testing, we analyzed the randomness of binary streams generated by the proposed PRNG, using the well known NIST Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. The NIST suite is a statistical package consisting of 15 tests that were developed to test the randomness of (arbitrarily long) binary sequences produced by either hardware or software based cryptographic random or pseudorandom number generators. These tests focus on a variety of different types of non-randomness that could exist in a sequence.Each test produces a $P - value$, which is a real number from the interval [0,1]. If the $P - value$ of a test is greater than a significance

level α, by default equal to 0.01, then the analyzed binary sequence passes the test successfully, so the sequence would be considered to be a random one with a confidence of 99%. If the tested binary sequences are truly random, then the $P-values$ is expected to appear uniform in the interval $[0,1)$. The $P-value_T$ corresponding to the uniformity of the $P-values$ must to be greater than 0.0001 to ensure that the $P-values$ could be considered uniformly distributed [14].

For this analysis, we have generated $m = 2000$ different binary streams from 500 randomly chosen seeds, each sequence having a length of $n = 1000000$ bits. Binary streams were obtained from the 32-bit binary unsigned representation of the output t_n. For each binary stream we computed $P-value$ corresponding to all the 15 tests of the NIST suite. The obtained results are summarized in Table 1. From the second column of Table 1 we can observe that the computed proportion for each test lies inside the confidence interval, which is $[0.983, 0.996]$ in this case, and from the third column of Table 1 we can observe that the $P-values$ for each statistical test are uniformly distributed. Hence, we can state that the tested binary streams are random with respect to all tests of the NIST suite.

Table 1. The results of the NIST tests

Test Name	Passing Ratio of the Test	Uniformity p-value	Result
Frequency	0.991	0.346453	SUCCESS
Block Frequency	0.994	0.650652	SUCCESS
Cumulative Sums	0.992	0.030297	SUCCESS
Runs	0.992	0.961039	SUCCESS
Longest Run	0.989	0.519103	SUCCESS
Rank	0.991	0.947805	SUCCESS
FFT	0.988	0.664168	SUCCESS
NonOverlapping Template	0.987	0.033362	SUCCESS
Overlapping Template	0.991	0.491459	SUCCESS
Universal	0.988	0.985339	SUCCESS
Approximate Entropy	0.991	0.925888	SUCCESS
Random Excursions	0.985	0.986305	SUCCESS
Random Excursions Variant	0.989	0.718590	SUCCESS
Serial	0.991	0.405613	SUCCESS
Linear Complexity	0.988	0.015982	SUCCESS

6. CONCLUSIONS

We have proposed a new PRNG which has a hybrid structure, based on three classical RANROT generators of type B3, along with four chaotic logistic maps iterated independently. The pseudorandom number sequence was obtained combining the outputs of three RANROT generators of type B3 using a RANROT type A expression. We have also tested rigorously the randomness of generated sequences using the NIST suite, which consists of 15statistical tests designed to detect the specific characteristics expected of truly random bit sequences. The results of statistical testing are encouraging and show that the proposed PRBG has very good cryptographic properties.

7. REFERENCES

[1]. Dorrendorf, L., Gutterman, Z., Pinkas, B., "*Cryptanalysis of the random number generator of the Windows operating system*", ACM Transactions on Information Systems, vol. 13(1), pp. 1-32, 2009.

[2]. Goldberg, I., Wagner, D., "*Randomness and the Netscape browser*", Dr.Dobb's Journal, pp. 66-70, 1996.

[3]. Gutterman, Z., Pinkas, B., Reinman, T., "*Analysis of the Linux Random Number Generator*", Proceedings of the 2006 IEEE Symposium on Security and Privacy, pp. 371-385, 2006.

[4]. Klein, A., "*Attacks on the RC4 stream cipher*", Designs, Codes and Cryptography, vol. 48(3), pp. 269-286, 2008.

[5]. Blum, L., Blum, M., Shub, M., "*A simple unpredictable pseudorandom number generator*", SIAM Journal on Computing, vol. 15, pp. 364-383, 1986.

[6]. Fog, A., "*Chaotic Random Number Generators with Random Cycle Lengths"*, http://www.agner.org/ random/ theory/chaosran.pdf, 2001.

[7]. Mascagni, M., Cuccaro, S.A., Pryor, D.V., Robinson, M.L., "*A fast, high quality, and reproducible parallel lagged-Fibonacci pseudorandom number generator*", Journal of Computational Physics, vol. 119, pp. 211-219, 1995.

[8]. Orue, A.B., Montoya, F., Hernández, E.L., "*Trifork, a New Pseudorandom Number Generator Based on Lagged Fibonacci Maps*", Journal Of Computer Science and Engineering, vol. 2(2), pp. 46-51, 2010.

[9]. Marsaglia, G., Tsang, W.W.,*"Some difficult-to-pass tests of randomness"*, Journal of Statistical Software, vol. 7(03), pp. 1-8, 2002.

[10]. Pellicer-Lostao, C. , Lopez-Ruiz, R., "P*seudo-Random Bit Generation Based on 2D Chaotic Maps of Logistic Type and Its Applications in Chaotic Cryptography*", Journal of Computational Science and Its Applications, vol. 5073, pp. 784-796, 2008.
[11]. Stojanovski, T., Kocarev, L.C., "*Chaos-Based Random Number Generators*", IEEE Transactions on Circuits and Systems, vol. 48(3), pp. 281-288, 2001.
[12]. Wang, Y., Shen, H., Yan, X., "*Design of a Chaotic Random Number Generator*", Chinese Journal of Semiconductors, vol. 26(12), pp. 2433-2439, 2005.
[13]. Shannon, C., "*Communication theory of secrecy systems*", The Bell System Technical Journal, vol. 28, pp. 656–715, 1949.
[14]. 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 the validation of random number generators and pseudorandom number generators for cryptographic applications*", NIST Special 800-22, 2010.
[15]. Brent, R.P., *"Fast and reliable random number generators for scientific computing"*, Proceedings of PARA'04 Workshop on the State-of-the-Art in Scientific Computing, pp. 1–10, 2004.
[16]. Marsaglia, G., *"Diehard Battery of Tests of Randomness"*, http://www.stat.fsu.edu/pub/diehard/, last accesses: October 2013.
[17]. Răcuciu, C., Grecu, D., Boriga, R., Dăscălescu, A.C., "*Analysis of a Pseudorandom Number Generator with Random Cycles*", Megabyte, vol. 9, pp. 95-100, 2010.
[18]. Boriga, R., Dăscălescu, A.C., "*A Method for Increasing the Randomness of Lagged Fibonacci Generators*", Annals of the Ovidius University - Economic Sciences Series, vol. 10, pp. 51-55, 2010.
[19]. May, R.M., "*Simple mathematical models with very complicated dynamics*", Nature, vol. 261, pp. 459-467, 1976.
[20]. Dăscălescu, A.C., Boriga, R., "*A Novel Fast Chaos-Based Method for Generating Random Permutations with High Shift Factor Suitable for Image Scrambling*", Nonlinear Dynamics, vol. 74(1-2), pp. 307-318, 2013.
[21]. Baptista, M.S., "*Cryptography with Chaos*", Physics Letters A, vol. 240, pp. 50–54, 1998.
[22]. Huang, F., Guan, Z.H., "*Cryptosystem using chaotic keys*", Chaos, Solitons and Fractals, vol. 23(3), pp. 851–855, 2005.
[23]. Wong, W.K., Lee, L.P., Wong, K.W., "*A modified chaotic cryptographic method*", Computer Physics Communications, vol. 138, pp. 234–236, 2001.
[24]. Patidar, V., Pareek, N.K., Sud, K.K., "*A Pseudo Random Bit Generator based on Chaotic Logistic Map and its Statistical Testing*", Informatica, vol. 33, pp. 441–452, 2009.
[25]. Pareek, N.K., Patidar, V., Sud, K.K., "*Image Encryption using Chaotic Logistic Map*", Image and Vision Computing, vol. 24(9), pp. 926–934, 2006.
[26]. Alligood, K.T., Sauer, T.D., Yorke, J.A., "*Chaos: An Introduction to Dynamical Systems*", Springer Verlag, 1996.
[27]. Şerbănescu, A., Rîncu, C.I., "*Systemes et signaux face au chaos. Applications aux communications*", MTA Press, 2008.
[28]. Wolf, A., Swift, J.B., Swinney, H.L., Vastano, J.A., "*Determining Lyapunov exponents from a time series*", Physica D, vol. 16, pp. 285-317, 1985.