STATISTICAL ANALYSIS OF PSEUDORANDOM BINARY SEQUENCES GENERATED BY USING TENT MAP

Azeem ILYAS[1], Adriana VLAD[2], Adrian LUCA[3]

The paper presents a detailed analysis of two types of pseudorandom binary sequence generators obtained by using tent map. The test is performed using a NIST statistical test suite which is widely used for testing the randomness of any random number generator. The binary sequences under investigation are obtained either by considering all the successive iterations of the tent map and choosing a threshold equal to the tent map parameter or by applying a periodical sampling on the tent map values and by choosing a threshold equal to 0.5. Additionally, the paper comes up with a new discussion concerning the elements of the secret key for both of the generators. Based upon the results presented in the paper, both generators can be used for designing a new cipher where the pseudorandom binary sequence is the main ingredient of the cipher.

Keywords: pseudorandom binary sequences, tent map, NIST statistical test suite

1. Introduction

Producing cryptographically secure ***P**seudo **R**andom **N**umber **G**enerators* (PRNGs) is a well known debate for the last 50 years. A special interest is paid in the literature for the construction of pseudorandom binary sequences generators based on chaotic systems, [1]-[4]. The features of the chaotic map such as sensitivity to initial condition and/or control parameter, the stationarity and the ergodicity, all of them make chaotic maps a worthy choice of achieving the Shannon's concept of good practical secrecy systems. If pseudorandom binary sequences are generated for cryptographic purposes, usually a simple inspection in not enough, it is always better to put the binary sequences under some of the known rigorous statistical test suites before using them. The most common tests suites used in the literature to analyze the random number generators are NIST,

[1] PhD Student, Faculty of Electronics, Telecommunications and Information Technology, University POLITEHNICA of Bucharest, 1-3, Iuliu Maniu Bvd., Bucharest 6, Romania.
email: azeemilyas@gmail.com
[2] Prof., Faculty of Electronics, Telecommunications and Information Technology, University POLITEHNICA of Bucharest, 1-3, Iuliu Maniu Bvd., Bucharest 6, Romania.
The Research Institute for Artificial Intelligence, Romanian Academy, 13, Calea 13 Septembrie, Bucharest 5, Romania. email: avlad@racai.ro, adriana_vlad@yahoo.com
[3]Lecturer, Faculty of Electronics, Telecommunications and Information Technology University POLITEHNICA of Bucharest, 1-3, Iuliu Maniu Bvd., Bucharest 6, Romania
email: adrian.luca@upb.ro

[5] and DIEHARD, [6]. The both test suites are considered the most stringent test among others.

The aim of this paper is to perform an in-depth analysis of two pseudorandom binary generators (PRBG) constructed starting from the tent map, [3]. In [3], the quality of the two PRBG was theoretically proved and authors have performed a statistical evaluation by using probability tests of *m*-grams (letter, bigram, trigram probabilities). This time, NIST statistical test suites are the primary tools for an analysis of the proposed PRBG.

The paper is formulated in two parts. Section 2 summarizes the method for generating the pseudorandom binary sequences. Additionally, Section 2 provides a discussion concerning the length of the binary sequences and assists in defining the secret key elements from cryptographical point of view. Section 3 presents a detailed analysis of PRBG using NIST test suite.

2. An overview of the pseudorandom binary generators

The tent map used to generate pseudorandom binary sequences in [3] is given by the following equation:

$$x_{k+1} = f(x_k) = \begin{cases} \dfrac{x_k}{a}, & 0 \le x_k \le a \\ \dfrac{1-x_k}{1-a}, & a < x_k \le 1 \end{cases} \tag{1}$$

where $a \in (0;1)$ is the tent map parameter. The tent map defined in (1) has uniform invariant density in $[0;1]$ interval. Binary sequences are obtained from the real values x_k of tent map (1) by a comparison with a *c* threshold as in Fig. 1 and relation (2). The $[0;1]$ range of the tent map is divided into two subintervals associated to two discrete values 0 and 1. The *c* threshold is chosen either 0.5 or equal to the tent map parameter , depending on the generator type proposed in [3]:

A generator. A practical fair coin generator starting from successive iterations of the tent map (*c* threshold is chosen equal to the tent map parameter *a*);

B generator. A fair coin generator based on the statistical independence sampling distance of the tent map (*c* threshold is chosen equal to 0.5).

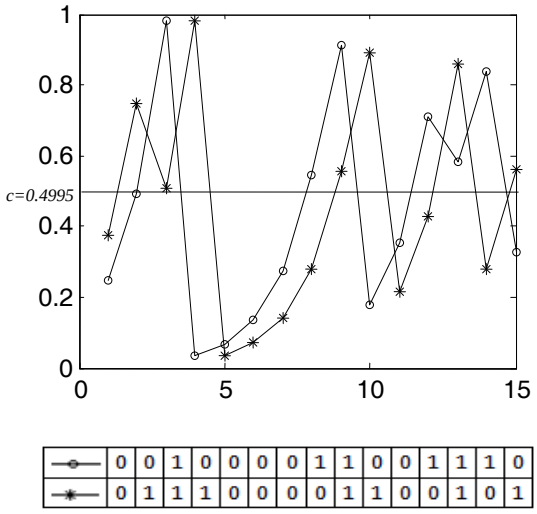

Fig. 1 Generation of binary sequences using the c threshold. Illustration for two trajectories indicated by $x_0 = 0.2457$ and $x_0 = 0.3728$, $c = a = 0.4995$

$$Z_k = \begin{cases} 0, & 0 \le x_k \le c \\ 1, & c < x_k \le 1 \end{cases} \tag{2}$$

In what follows, the two methods mentioned above for generating the binary sequences from the tent are presented.

A practical fair coin generator starting from successive iterations of the tent map

The tent map is iterated (starting from an initial condition randomly chosen) and all successive x_k values in (1) are considered; the x_k values are transformed into binary values with the c threshold equal to the tent map parameter. As already proved in [3], if the threshold is chosen equal to the tent map parameter (*i.e.* $c = a$), then the binary sequences coming from successive x_k values of the tent map are *i.i.d* (data coming out from *independently and identically distributed* random variables), having probabilities $P_0 = a$ and $P_1 = 1 - a$. In order to have binary sequences that comply to the fair coin model, the threshold and the tent map parameter has to be chosen as $c = a = 0.5(1 - \delta)$, [3]. The δ value is evaluated using the type II statistical error probability, (3), for the hypothesis that the probability of each of the binary symbol is 0.5.

Note. In (3), $z_{\alpha/2}$ is $\alpha/2$-point value of the standard gaussian law for a significance level α (for example, if $\alpha = 0.05$, then $z_{\alpha/2} = 1.96$).

$$\beta(N,\delta)=\int_{0.5-\varepsilon}^{0.5+\varepsilon}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\widetilde{P})^2}{2\sigma^2}\right)dx \tag{3}$$

$$\widetilde{P}=0.5(1\pm\delta),\ \sigma^2=\widetilde{P}(1-\widetilde{P})/N,\ \varepsilon=z_{\alpha/2}\sqrt{1/(4N)}$$

Comments concerning the length of the binary sequences and choice of tent map parameter.

In [3], the maximum deviation considered for the balanced case was $\delta=0.001$, meaning $a\in[0.4995;\,0.5005]$ with $a\neq 0.5$, for the sequences of length $N\leq 65536$. In this case, type II statistical error probability defined in (3) is $\beta>0.94$. That means, the binary sequences of a length $N\leq 65536$ coming out from the successive values of the tent map with $a\in[0.4995;\,0.5005]$ comply with the fair coin model (*i.e.* if a probability test is performed, the test will accept in about 95% cases that binary sequences comply with the throwing of the fair coin).

This generator is relying on type II statistical error but type II statistical error depends upon two parameters, N and δ. In order to be sure that the binary sequences always obey the fair coin model, δ needs to be reevaluated for every length N (for example, if a binary sequence is having length $N\leq 10^6$, it results $\delta=0.0001$ and the tent map parameter $a\in[0.49995;\,0.50005]$).

A fair coin generator based on the statistical independence sampling distance of the tent map

We iterated the tent map (starting from a randomly chosen initial condition and a fixed tent map parameter chosen in $[0.4;\,0.6]$) and we preserved only the x_k values extracted by a periodical sampling of (1) with a distance $d=15$ iterations; then, the extracted x_k values are transformed into binary values by choosing a threshold $c=0.5$. Note that the value of the sampling distance enables to have statistical independence between the extracted values of the tent map, [7].

Note. We can have *i.i.d.* data which comply with the fair coin model for any value of the tent map parameter by choosing a threshold $c=0.5$ and using a periodical sampling of the tent map values (the sampling distance value which ensures statistical independence depends on the tent map parameter). For example, a sampling distance value of 30 iterations ensures the statistical independence for a large range of values of the tent map parameter, [7].

Comments concerning cryptographic features of the two type generators.

For type **A** generator, the interval for the control parameter is practically known, so one cannot consider to put the control parameter as an element of the secret-key, resulting only to have the initial condition as a secret key. If the tent map is realized on a finite precision system and taking into account the double precision of floating point presentation, the effective key size of the type **A** generator approximately can be thought of 53 bits (bearing in mind that we considered only the mantissa part). Although the key size looks good from practical point of view, it is also a known fact that if $c = a$ the cryptanalyst can easily get back the initial condition, [8]. Thus, type **A** key-generator is good only from the binary sequence generation point of view, as binary sequences are generated from successive values of the tent map, but from the security point of view it should be used only as a part of mixing transformation in some cipher scheme.

For type **B** generator, we can select any value for the tent map parameter, so tent map parameter can be put as part of the secret key alongside with the initial condition. Again, considering the mantissa part of 64-bit floating point representation, we can think of the key size about 106 bits, which is well according to the today standards. Type **B** generator is good from the key-space point of view and the way the binary sequences are obtained ensures that the method resists the parameter estimation techniques, but it lacks the computational efficiency (generation of binary sequences) compared to type **A** generator. It is because of the fact that for the generation of each single bit, tent map has to be iterated with a specific sampling distance (distance value depending on the tent map parameter).

It is usually recommended to perform an analysis using NIST statistical tests or DIEHARD. The basic purpose of these statistical test suites is to detect the non-random behavior which looks random on the initial inspection. For the purpose of this study we have considered NIST for analyzing the two above mentioned generators.

3. Analysis by means of NIST test suite

NIST 800-22 was issued by the National Institute of Standards and Technology (NIST) (an agency of the U.S. Department of Commerce) in 2001 and has been revised in 2008, namely, NIST SP800-22 rev. 1a, [5]. The revised test suite contains 15 statistical tests based on hypothesis testing; each test tries to detect anomalies if present in the random sequences. A hypothesis test is a formal procedure used in the statistics to accept or reject an assumption. In NIST test suite, this assumption refers to the idea that if a particular test is passed, we consider that the sequence under investigation is random. For each test, a relevant

randomness statistic is chosen and used to determine the acceptance or rejection of the null hypothesis. Under an assumption of randomness, such a statistic has a distribution of possible values. A theoretical reference distribution of this statistic under the null hypothesis is determined by mathematical methods and corresponding probability value (*p*-value) is computed, which summarizes the strength of the evidence against the null hypothesis. For each test a *p*-value is calculated with a specified significance level α. A *p*-value less than α would mean that the sequence is non-random and if a *p*-value is greater than α, the sequence is accepted as random (when $\alpha = 0.01$, we accept the random binary sequences with a confidence level of 99%), [5].

To gain confidence that the binary sequences have all the characteristics required from a PRNG from cryptographical point of view, an investigation is made on $m = 1000$ sequences where the length of each binary sequence is $N = 10^6$ (in other words we disposed of 1000 independent binary sequences of length $N = 10^6$). NIST SP800-22 rev. 1a, [5], provides a guideline involved in the statistical testing to make a decision on the empirical calculated results of each test. We here present the steps in brief, for more detail one is referred to [5].

Step1. *Construct m-binary sequences from the random number generator*
Step2. *Execute the statistical test suite to get the p-values for each of the 15 tests.*
Step3. *Examine the p-values: an output file will be generated by the test suite with relevant intermediate values, such as test statistics, and p-values for each statistical test. Based on these p-values, a conclusion regarding the quality of the sequences can be made.*
Step4. *Assessment based upon the p-values either test passes or fails: For each statistical test, a set of p-values (corresponding to the set of sequences) is produced. For a fixed significance level* α*, a certain percentage of p-values are expected to indicate failure (i.e. if* $\alpha = 0.01$ *then about 1% of the sequences are expected to fail). For each statistical test, the proportion of sequences that pass is computed and analyzed.*

To generate the samples for type **A** generator, parameter values are chosen randomly in $[0.49995; 0.50005]$ and for type **B** generator the tent map parameter values are randomly chosen in the interval $[0.4; 0.6]$ and a periodical sampling $d = 15$ is applied to get the binary sequences. For both generators, the $m = 1000$ sequences are generated starting from initial conditions randomly chosen in $(0; 1)$ interval.

To assess the *p*-values for each of the test present in test suite, NIST has adopted two approaches which are mentioned next in detail.

i) Passing ratio of each test

To determine the passing ration of each test, the significance level for each test is set to 0.01 meaning that 99% test samples should pass the test. We

considered a significance level of 1% . By estimation theory and resuming 1000 times each test, the range of acceptable proportions for each of the individual tests can be determined by using the confidence interval defined as $\hat{p} \pm 3\sqrt{\hat{p}(1-\hat{p})/m}$, where $\hat{p} = 1 - \alpha = 1 - 0.01 = 0.99$ and $m = 1000$. The acceptance region in our case will be $0.99 \pm 3\sqrt{0.99(1-0.99)/1000}$, namely $[0.9806; 0.9994]$.

Table 1 presents results for 13 tests from the NIST suite for each of the two generators under discussion; if the passing ratio belongs to the acceptance region, the decision is SUCCES (the test is passed).

Table 1

Proportions of the sequences passing against each test

	type **A** generator		type **B** generator	
Test Name	Passing Ratio	Decision	Passing Ratio	Decision
Frequency(Mono) Test	0.991	SUCCESS	0.992	SUCCESS
Frequency test within a block	0.993	SUCCESS	0.987	SUCCESS
Cumulative Sums (Forward)	0.992	SUCCESS	0.993	SUCCESS
Cumulative Sums (Reverse)	0.989	SUCCESS	0.991	SUCCESS
Runs Test	0.987	SUCCESS	0.989	SUCCESS
Test for the longest runs of 1's	0.988	SUCCESS	0.99	SUCCESS
Rank Test	0.994	SUCCESS	0.992	SUCCESS
FFT	0.994	SUCCESS	0.989	SUCCESS
Non-overlapping Template[1]	0.99	SUCCESS	0.989	SUCCESS
Overlapping Template	0.991	SUCCESS	0.983	SUCCESS
Universal	0.99	SUCCESS	0.992	SUCCESS
Approximate Entropy	0.987	SUCCESS	0.997	SUCCESS
Serial Test	0.983	SUCCESS	0.994	SUCCESS
Linear Complexity	0.988	SUCCESS	0.991	SUCCESS

[1]the result is presented only for template 000000001 from all other templates (template length is 9).

ii) p-values uniformity of each test

It is generally suggested examining the uniformity of *p*-values for each of the individual tests using χ^2 test. The χ^2 test value is defined in (4) where: F_i is the number of *p*-values in the *i* class (the *p*-values are put into 10 classes between 0 and 1, thus the degree of freedom in this case for χ^2 test is 9); m is number of samples (here $m = 1000$).

$$\chi^2 = \sum_{i=1}^{10} (F_i - m/10)^2 / (m/10) \quad (4)$$

The significance level is again set to $\alpha = 0.01$. Table 2 presents the results for each test for both **A** and **B** generators**.** If the χ^2 test value (4) is less than 33.72, the test is assumed as SUCCESS, otherwise FAILURE (33.72 is the $\alpha/2$-point value of the χ^2 law of 9 degrees of freedom).

Table 2

***p-values* uniformity of each test**

	type **A** generator		type **B** generator	
Test Name	Test value	Decision	Test value	Decision
Frequency(Mono) Test	4.64	SUCCESS	8.96	SUCCESS
Frequency test within a block	14.42	SUCCESS	14.34	SUCCESS
Cumulative Sums	6.58	SUCCESS	15.28	SUCCESS
Cumulative Sums	9.64	SUCCESS	2.48	SUCCESS
Runs Test	4.92	SUCCESS	4.02	SUCCESS
The longest runs of 1's	9.04	SUCCESS	10.28	SUCCESS
Rank Test	3.32	SUCCESS	10.18	SUCCESS
FFT	8.96	SUCCESS	10.66	SUCCESS
Non-overlapping Template[1]	13.16	SUCCESS	11.96	SUCCESS
Overlapping Template	5.26	SUCCESS	18.06	SUCCESS
Universal	7.74	SUCCESS	14.1	SUCCESS
Approximate Entropy	9.82	SUCCESS	11.92	SUCCESS
Serial	11.3	SUCCESS	11.32	SUCCESS
Linear Complexity	9.16	SUCCESS	8.52	SUCCESS

[1]the result is presented only for template 000000001 from all other templates (template length is 9)

Note. The experimental study included other two tests from NIST suite, namely Random Excursion Test and Random Excursion Variant Test, Table 3; both tests have considered only 633 and 634 sequences out of 1000 samples under investigations. It is due to the fact that the probability of stopping the test is 38%. Although for both generators the two tests were passing, these tests were not included in Table 1 and Table 2, to keep the symmetry concerning the number of binary sequences under investigations.

Table 3

Detailed results concerning the Random Excursion and Random Excursion Variant tests

		type **A** generator		type **B** generator	
		χ^2 test value	*Passing ratio*	χ^2 test value	*Passing ratio*
Random Excursion	-4	16.5893	0.9921	18.6498	0.987382
	-3	14.5039	0.9842	21.9306	0.988959
	-2	7.7109	0.9921	7.1987	0.993691
	-1	12.1975	0.9826	10.4795	0.987382
	1	9.1643	0.9889	13.5710	0.984227
	2	7.9953	0.9952	5.7476	0.976341
	3	3.3507	0.9873	9.7539	0.985804
	4	4.0774	0.9826	12.7192	0.981073
Random Excursion Variant	-9	8.2164	0.988942	9.5962	0.998423
	-8	8.6272	0.990521	8.2713	0.981073
	-7	5.4992	0.992101	9.8801	0.981073
	-6	9.0063	0.995261	6.3155	0.984227
	-5	10.0490	0.993681	16.2208	0.992114
	-4	5.2780	0.996840	13.8233	0.995268
	-3	8.2164	0.987362	9.1861	0.993691
	-2	17.6951	0.985782	15.5268	0.993691
	-1	19.1485	0.990521	13.9495	0.993691
	1	16.4629	0.993681	17.1356	0.993691
	2	3.6667	0.992101	11.2050	1.000000
	3	8.0585	0.995261	12.1830	0.996845
	4	4.7725	0.993681	6.7571	0.993691
	5	8.2164	0.993681	12.0883	0.996845
	6	5.9731	0.988942	16.4732	0.995268
	7	15.3886	0.993681	13.3502	0.995268
	8	4.0458	0.993681	17.1356	0.996845
	9	9.3223	0.987362	7.1356	0.993691

4. Conclusions

The proposed pseudorandom binary sequences based on tent map are obtained in such a way that they comply with the fair coin model (if certain conditions are met for both of the generators). In this paper, we have rigorously tested the generated binary sequences using the NIST suite to be sure that they have all the specific properties which can be anticipated from a random sequence. The results of statistical testing are in concordance with the theoretical support provided.

The paper also brings into discussion and provides enough details concerning the key elements of the secret key for both generators. For type **A** generator, both tent map parameter and initial condition cannot be considered as elements in the secret key. On the other hand, the type **B** generator can have tent map parameter as well as the initial condition as part of the secret key. However,

for applications in cryptography, the suggestion is to use binary sequences (either from type **A** or from type **B**) only in product ciphers to avoid the chosen text attack.

REFERENCES

[1] *Li, S., Mou, X., Cai, Y.*, "Pseudo-Random Bit Generator Based on Couple Chaotic Systems and its application in Stream-Ciphers Cryptography". Lecture Notes in Computer Science (Progress in Cryptology – INDOCRYPT 2001), **vol. 2247**, 2001, pp. 316–329.

[2] *V. Patidar, K.K. Sud, N.K. Pareek*, "A Pseudo Random Bit Generator Based on Chaotic Logistic Map and its Statistical Testing", Informatica, **vol. 33**, 2009, pp. 441–452.

[3] *A. Luca, A. Ilyas, A. Vlad*, "Generating Random Binary Sequences Using Tent Map". Proc. IEEE Int. Symposium on Signals, Circuits and Systems (ISSCS), Iasi, Romania, June 30-July 1, 2011, pp. 81-84.

[4] *A. Kanso, N. Smaoui*, "Logistic Chaotic Maps for Binary Numbers Generations". Chaos, Solitons and Fractals, **vol. 40**, Issue 5, June 2009, pp. 2557-2568.

[5] *Runkin et al*, "A Statistical test suite for random and pseudo random number generators for cryptographic applications", NIST special publication 800-22, Rev. 1a, 2010.

[6] *G. Marsaglia*, DIEHARD: a battery of tests of randomness, 1997. http://stst.fsu.edu/geo/diehard.html

[7] *A. Luca, A. Vlad, B. Badea, M. Frunzete*, "A Study on Statistical Independence in the Tent Map", in Proc. IEEE Int. Symposium on Signals, Circuits and Systems (ISSCS), Iasi, Romania, July 9-10, 2009, pp. 481-484.

[8] *D. Arroyo*, Framework for the Analysis and Design of Encryption Strategies based on Discrete-Time Chaotic Dynamical Systems, Ph.D. thesis, Universidad Politécnica de Madrid,Escuela Técnica Superior de Ingenieros Agrónomos, Spain, 2009.