THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 14, Special Issue 2013, pp. 295–302

GENERATING CHAOTIC SECURE SEQUENCES USING TENT MAP AND A RUNNING-KEY APPROACH

Adriana VLAD[1,2], Adrian LUCA[1], Octavian HODEA[1], Relu TATARU[1]

[1]Faculty of Electronics, Telecommunications and Information Technology,
POLITEHNICA University of Bucharest, 1-3, Iuliu Maniu Bvd. Bucharest 6, Romania
[2]The Research Institute for Artificial Intelligence,
Romanian Academy, 13, Calea 13 Septembrie, Bucharest 5, Romania
Corresponding author: Adriana VLAD, E-mail: avlad@racai.ro

This paper completes recent results obtained by extending the running-key cipher procedure from natural language to applications over chaotic systems. We apply the new running-key approach on the chaotic tent map and prove its utility in obtaining practically zero-redundant pseudo-random number generators, alongside with the possibility to consider the initial condition and the tent map control parameter as elements in the secret key – a desideratum in chaos-based cryptography. The results are both theoretically and experimentally supported by combining concepts from information theory and statistical methods in the context of the chaotic system. The statistical evaluation includes NIST test suite for testing the randomness of the proposed binary generator. Based upon the results presented in this paper, the provided generator can be used for designing new cryptosystems where the pseudo-random binary sequences can be a chief support.

Key words: tent map, pseudorandom binary sequences, running-key cipher, noisy channel, NIST test suite

1. INTRODUCTION

The running-key procedure advanced in [1] and [2] for the logistic function is here extended and adapted for tent map. Mathematically speaking, tent map is a discrete time chaotic system described by relation (1):

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

where, $a \in [0,1]\backslash\{0.5\}$ is the control parameter and x_n is the current state of the system. Tent map defined in (1) has uniform invariant probability density in [0, 1] interval. Binary sequences (further denoted by Z) are obtained from the real x_n values of tent map by a comparison with a c threshold equal with the control parameter a as in Fig. 1 and relation (2).

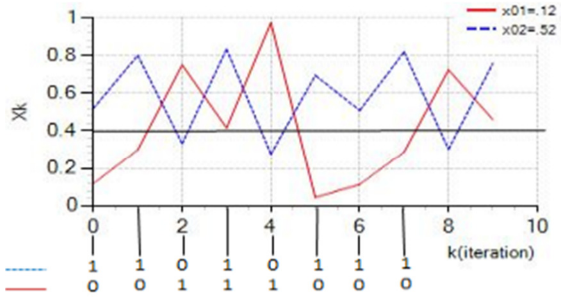


Fig. 1 – Two binary sequences generated by tent map for two different initial conditions. Illustration for $c = a = 0.4$.

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

The working versions of running-key procedure are illustrated in Fig. 2.

$$\begin{array}{ll}
Z_0 + Z_1 & \Longrightarrow Y_1 \\
Z_0 + Z_1 + Z_2 & \Longrightarrow Y_2 \\
Z_0 + Z_1 + Z_2 + Z_3 & \Longrightarrow Y_3 \\
\vdots & \\
Z_0 + Z_1 + Z_2 + Z_3 + \dots + Z_k & \Longrightarrow Y_k
\end{array}$$

Fig. 2 – A running-key approach applied to the chaotic system. Z_i stands for the binary typical sequences.

The method has the particularity that all the summed sequences are typical binary sequences generated by tent map (1) considering successive iterations for a fixed a control parameter and a binarization threshold $c = a$. The summations in the Fig. 2 are bit by bit modulo 2.

The extending of running-key procedure for chaotic systems is due to the ergodicity property of these systems. For fixed a parameter in (1) we have an ergodic random process. Each parameter change will lead to another random process. Thus, future discussions will be made for fixed a value of the parameter. Each typical sequence will be defined by a randomly chosen initial condition in [0,1] and a fixed control parameter $c = a$, the same for all the summed sequences.

Our particular concern when applying the running-key approach (see Fig. 2) is to evaluate the number of summed typical binary sequences that can lead to Y_k sequence compatible with the fair coin model. In other words, we aim to determine the k value, so that the output information source that produces Y_k sequences becomes a binary memory-less source of zero-redundancy. In this case, the Y_k sequence will be statistically independent of its components (sequences that are summed up) and it would not be possible to be decomposed into its components.

This way of proceeding, that we call the running-key approach, includes a representation of variants in Fig. 2 by using a cascade of information channels that allows a simple way to determine the value of k when reaching the Y_k sequence compatible with the fair coin model. Using the cascade of information channels, all evaluations will be done directly inspecting the output sequence Y_k. The requirement that Y_k is compatible with the fair coin model implicitly ensures the fact that Y_k cannot be separated into its components. This condition is a desideratum that can be reached (verified) by various statistical tests carried on Y_k, but avoiding more complicated assessments of "cryptanalysis" type.

Note: The term "cryptanalysis" is misused here because what we get in the end is a good quality statistical pseudo-random binary generator, where Y_k sequences could be used as enciphering sequences (keys) in a cryptosystem. For a better understanding, recall that the evaluation of running-key cipher applied to natural language was based on cryptanalysis. Thus, in versions 1 and 2 (corresponding to Fig. 2) the cipher was broken (*i.e.* the cryptograms Y_1 and Y_2 could be decomposed into the natural texts that participated to the respective summation), but in the variant 3 (*i.e.* Y_3, meaning four summed natural texts) the cipher could not be broken, result which allowed assessment of relative redundancy (about 75%) for English language, [1] - [6].

In what follows, we show that the running-key procedure applied on tent map will provide a binary pseudo-random number generator compatible with the fair coin model for a large range of values for a control parameter.

Section 2 presents the theoretical evaluation of the generator obtained by the running-key approach. Section 3 shows the experimental results obtained when catching the k value for which Y_k type sequence corresponds to fair coin model. The investigation was made using probability tests supported by a Monte-Carlo analysis and by the NIST test suite. Section 4 presents final remarks and conclusions.

2. THEORETICAL EVALUATION

By successively iterating the tent map (1) and choosing binarization threshold equal to tent map parameter, the obtained binary sequence obeys to the *i.i.d.* statistical model (data coming up from independently and identically distributed random variables), result shown in [7]. The result is valid for any control parameter value (except $a = 0.5$, when the tent map statistical behavior is no more chaotic). Although binary data are statistically independent even in successive iterations, in order to comply with the fair coin model a numerical restriction concerning the parameter range of values is required [7] and [8]. Thus, if the parameter is chosen in the interval (0.49995, 0.50005), a binary pseudorandom generator of good statistical quality can be obtained for typical sequences of length at most equal to 10^6 binary symbols (see section 3). Although being of a good statistic quality, the generator has the disadvantage that the initial condition can be easily recovered based on a binary sequence. In this case, the initial condition and the control parameter from (1) may not be included in the secret key in cryptographic applications, [8] and [9].

By the running-key procedure, Fig. 2, we get Y_k type binary sequences compatible with the fair coin model for a wide range of values of the control parameter. According to the running-key procedure, the summed sequences will not be recovered from Y_k. In this way a major obstacle against the recovery of the control parameter value and of the initial conditions will be encountered, which could lead to their inclusion in the secret key.

In Fig. 2, Z_i are *i.i.d.* binary sequences of *N* size, obtained by considering all successive iterations of (1), fixed a value and $c = a$.

Fig. 3 introduces a new view of the running-key procedure from Fig. 2, through a cascade of information channels. The cascaded information channels allow to catch the moment when the running-key procedure stops, by evaluating the entropies of the secondary sources $Y_1, Y_2, \cdots Y_k$. All the implied information sources $Z_0, Y_1, Y_2, \cdots Y_k$, are binary and memory-less; Fig. 3 allows a decision concerning the degree of dependence/independence between the Z_0 input source (which corresponds to the tent map and to its typical sequences Z_i) and the Y_k output source.

Note: The typical sequence $Y_1 = Z_0 + Z_1$ (from Fig. 2) can be viewed as a particular realization of the Y_1 source in Fig. 3 (output of the first information channel). Similarly, Y_k typical sequence from Fig. 2 can be viewed as a particular realization of the Y_k source in Fig. 3 (Y_k source in Fig. 3 and Y_k typical sequence from Fig. 2 are denoted by the same letter).

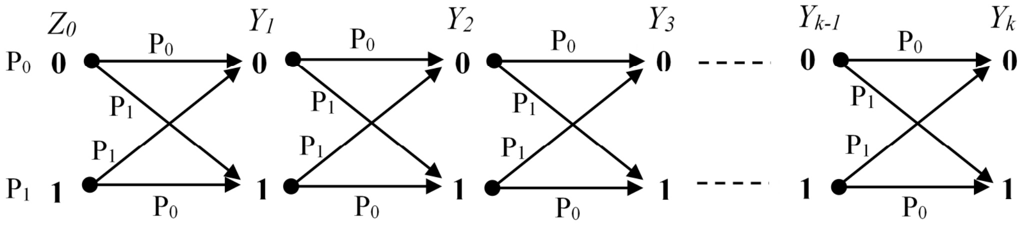


Fig. 3 – An information channel depiction of running-key approach.

The cascaded information channels are binary symmetric channels, each of them having the noise matrix from relation (3). P_{ij} elements of the noise matrix are actually the probabilities of the Z_0 information source, P$_0$=$P(Z_0 = 0) = a$ and P$_1$=$P(Z_0 = 1) = 1 - a$ [7]. The Y_1 secondary source has the probabilities: $P(Y_1 = 0) = a^2 + (1 - a)^2 = 1 - 2a(1 - a)$ and $P(Y_1 = 1) = 2a(1 - a)$. It follows that the entropy assigned to Y_1 binary source can be obtained by using the entropy function $H(p) = -p\log_2 p - (1 - p)\log_2(1 - p)$. So $H(Y_1) = H(2a(1 - a))$.

$$\begin{array}{c|c|cc} & & \multicolumn{2}{c}{Y_1} \\ & P_{ij} & 0 & 1 \\ \hline & 0 & a & 1-a \\ Z_0 & 1 & 1-a & a \end{array} \qquad P_{ij} = P(Y_1 = j / Z_0 = i), \quad i, j \in \{0,1\} \tag{3}$$

Table 1 presents the entropy values corresponding to Z_0 and Y_k information sources, evaluated for two control parameter values: $a = 0.2$ and $a = 0.4$.

Table 1

Entropies assigned to the cascaded information channels

Entropy	$a = 0.2$	$a = 0.4$
$H(Z_0) = H(a)$	0.7243	0.9699
$H(Y_1) = H(2a(1-a))$	0.9074	0.9987
$H(Y_3) = H(2u(1-u))$; $u = 2a(1-a)$	0.9915	0.9994
$H(Y_5) = H(2w(1-w))$; $w = 2u(1-u)$	0.99978	0.9999999

Based on Fig. 3 and fundamentals from information theory and cryptography [4], [5], [6] and [10], the cascaded binary symmetric channels successively convey information from the Z_0 input source, which is a redundant source, to the Y_k which is practically a non-redundant source. Thus we have the following relationship between the entropies of the binary successive sources:

$$H(Z_0) < H(Y_1) < H(Y_2) < H(Y_3) < \cdots < H(Y_k) \tag{4}$$

The running-key procedure determines the k value for which $H(Y_k) \cong 1$. To decide when $H(Y_k) \cong 1$, a probability test on the Y_k binary sequence will be used. The decision that the sequence is compatible with the fair coin model will be taken based on type II statistical error [11].

The probability test has the following hypotheses:

✓ null hypothesis, H_0: $p = p_0$

✓ alternative hypothesis, H_1: $p \neq p_0$

where p is the true probability of the investigated event.

The statistical significance level is $\alpha = 0.05$, thus $z_{\alpha/2} = 1.96$ ($z_{\alpha/2}$ is $\alpha/2$-point value of the standard Gaussian law). The H_0 hypothesis is accepted if $|\hat{p} - p_0| \leq \varepsilon = z_{\alpha/2}\sqrt{p_0(1-p_0)/N}$ where $\hat{p}$ is the estimated value of p probability and N is the size of the $i.i.d.$ sequence. The type II statistical error probability for the test is expressed by (5).

$$\beta(N,\delta) = \int_{p_0-\varepsilon}^{p_0+\varepsilon} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\tilde{p})^2}{2\sigma^2}\right) \mathrm{d}x$$
$$\varepsilon = z_{\alpha/2}\sqrt{\frac{p_0(1-p_0)}{N}},\ \tilde{p} = p_0(1 \pm \delta), \sigma^2 = \tilde{p}(1-\tilde{p})/N \tag{5}$$

The decision on k value ($H(Y_k) \cong 1$) relies on type II statistical error probability and depends upon two parameters, N and δ (here $p_0 = 0.5$). In order to be sure that the binary sequence always obeys fair coin model (i.e. the final expected result), δ needs to be reevaluated for any N length (required by the application).

The β type II statistical error probability of accepting wrong data as good data (*i.e.* to accept H_0 even if the coin is slightly unbalanced) is evaluated according to relation (5) where $\alpha = 0.05$ and $p_0 = 0.5$. For example if $\delta = 0.001$ and $N \leq 65536$, the type II statistical error probability is greater than 0.94. For a data volume $N \leq 10^6$ (needed by NIST tests) and aiming that the probability test will pass in about 95% of cases ($\beta \approx 0.95$), it results $\delta \approx 10^{-4}$. If follows that a parameter value chosen in the interval [0.49995, 0.50005] will enable to obtain a typical Y_k binary sequence complying with the fair coin model without any summation.

By applying the running-key procedure for a choice of the a parameter value in a larger interval, *e.g.* $a \in [0.4, 0.6]$, the decision that the output sequence complies with the fair coin model is taken when $k = 4$ respectively for Y_4 (summing 5 typical sequences generated by tent map), see *Table 1* and Section 3.

Comments on Table 1:

For $a = 0.2$ parameter value, the resulting entropy of the input binary source is $H(Z_0) = 0.7243$ bits. If we sum two typical Z_i sequences emitted by tent map it results Y_1 information source of entropy $H(Y_1) = 0.9074$ bits. If we sum six typical sequences we obtain Y_5 sequence emitted by a practically non-redundant information source namely, $H(Y_5) = 0.99978$.

Following the reasoning applied for the evaluation of natural language redundancy, [2], [3] and [6], the recovery of the 6 sequences of N length that participated to the summation would imply to determine $6 \cdot N \cdot H(Z_0)$ unknown binary symbols from N known binary symbols (representing Y_k).

Obviously, we cannot decompose Y_5 into the 6 components. If we could decompose it into the 6 components, we should have $6 \cdot H(Z_0) < 1$. Judging from this, it would be enough to make a summation of 2 typical sequences produced by tent map with $c = a = 0.2$. In order to obtain a non-redundant generator for $a = 0.2$ and $N = 10^6$, we need more than 5 summations for the needed accuracy. The 5 summations implicitly ensure statistical independence between the Y_5 sequence and any of the Z_i sequences that participated to the summation, result which is valid only for N values (*e.g.* $N < 65536$) smaller than the requirement of the NIST test suite.

To conclude: Finally we obtain a non-redundant sequence, completely independent of any typical sequence participating in the summation. So, it is difficult to believe that someone could easily recover, without exhaustive trials, the $k + 1$ initial conditions and the control parameter value, based on Y_k.

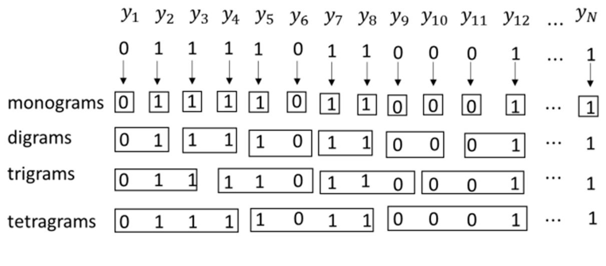
3. EXPERIMENTAL RESULTS

Here we present experimental results on the proposed generator. The running-key procedure is evaluated by successively investigating the typical sequences $Y_1, Y_2, \cdots, Y_k$, aiming to catch the moment when Y_k complies with the fair coin model. Section 3.1 presents the experimental study by using basic statistical tests concerning m-gram probability (m successive binary symbols) supported by the Monte-Carlo analysis. Section 3.2 presents experimental studies on the same generator using the NIST test suite.

3.1. Basic statistical methods

For this generator, the control parameter is equal to the binarization threshold, in which case the Z_i sequences obey to the *i.i.d.* model, however their probability law is not uniform. By successive summations we aim to determine the k value when Y_k sequence complies with the fair coin model.

Y_k sequences are obtained by a bit by bit modulo 2 summation of the Z_i sequences. Sequences denoted by Z_i are of size $N = 10^6$ bits and are generated by (1) for a fixed a control parameter value and randomly chosen initial conditions according to the uniform law in the (0, 1) interval. For example, summing up the two data sequences $Z_0 \leftrightarrow (z_{01}, z_{02}, \cdots z_{0N})$ and $Z_1 \leftrightarrow (z_{11}, z_{12}, \cdots z_{1N})$ we obtain $Y_1 \leftrightarrow (y_{11}, y_{12}, \cdots y_{1N})$, where $y_{1j} = z_{0j} + z_{1j}$.



y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8 y_9 y_{10} y_{11} y_{12} … y_N
0 1 1 1 1 0 1 1 0 0 0 1 … 1
monograms
digrams
trigrams
tetragrams

Fig. 4 – The m-gram experimental data sets.

Based on the obtained Y_k binary sequences we evaluated the m-gram ($m = 1,2,3,4$) probability by applying the usual probability test with the following hypotheses: H_0, meaning that the investigated probability is equal to the p_0 expected theoretical value; H_1, the investigated probability differs from p_0. The H_0 hypothesis is accepted if $|\hat{p} - p_0| \leq \varepsilon = z_{\alpha/2}\sqrt{p_0(1-p_0)/N}$ where $\hat{p}$ is the estimated value of the investigated m-gram probability. The m-gram data sets submitted to test are extracted from Y_k as shown in Fig. 4. Note that the data set volume for monogram is $N = 10^6$, for digram is $N/2$, for trigram $N/3$ and for tetragram $N/4$. The corresponding p_0 theoretical probabilities are: 0.5 for monogram, 0.25 for digram, 0.125 for trigram and 0.0625 for tetragram. We made a detailed analysis for each possible m-gram ($m = 1,2,3,4$).

For each and every investigated m-gram, we applied a Monte-Carlo analysis by resuming the probability test 500 times and recording the proportion of the H_0 hypothesis acceptance. The decision that Y_k sequence complies with the fair coin model was taken if the recorded proportion was within the interval $[0.93, 0.97]$, for each and every investigated m-gram.

In *Table 2* we show this kind of results for $a = 0.4$ and for the k summation number equal to 3 and 4. It can be noticed that the proportions of the H_0 hypothesis acceptance is inside $[0.93, 0.97]$ for all m-grams, when $k = 4$.

Table 2

Proportion of H_0 acceptance for m-gram probability test for $a = 0.4$

m-gram	Y_3	Y_4	m-gram	Y_3	Y_4
0	0.622	0.96	011	0.936	0.958
1	0.622	0.96	100	0.944	0.948
00	0.742	0.974	101	0.93	0.934
01	0.95	0.95	110	0.95	0.942
10	0.942	0.94	111	0.912	0.96
11	0.716	0.96	0000	0.884	0.96
000	0.834	0.948	0001	0.926	0.962
001	0.946	0.936	0010	0.926	0.942
010	0.942	0.956	1111	0.864	0.966

We made similar investigations as in *Table 2* for various a control parameter values. The overall study was summarized in *Table 3*. Namely, for each k value we determined a range of values for the choice of a control parameter. If the a control parameter is chosen inside the respective interval, the corresponding Y_k sequence complies with the fair coin model. For example, for $k = 3$ the a parameter value should be chosen within $(0.43, 0.57)$ interval.

Table 3

Tent map control parameter intervals that enable to generate the Y_k non-redundant sequences

k	Y_k	Assigned interval for a
0	$Y_0 = Z_0$	(0.49995, 0.50005)
1	Y_1	(0.497, 0.503)
2	Y_2	(0.48, 0.52)
3	Y_3	(0.43, 0.57)
4	Y_4	(0.39, 0.61)

3.2. NIST test suite

The performances of the proposed chaotic generator were evaluated using one of the most popular standards for investigating the randomness of binary data, namely NIST (National Institute of Standards and Technology) test suite. The NIST statistical test suite was specially designed for randomness examination of hardware or software data generated from cryptographic random or pseudo-random generators.

NIST tests were primarily applied to verify the results obtained by the running key approach, testing the Y_k sequences from sub-section 3.1. Thus, we checked the ranges of the control parameter a for which the chaotic generator provides pseudo-random and non-redundant binary sequences Y_k.

For our investigation we have generated a number of $m = 1000$ different Y_k binary sequences for each k value. Each Y_k sequence was generated using $k + 1$ initial conditions for tent map, uniformly chosen in the range $(0,1)$, each of length 10^6 bits (fixed a parameter and $c = a$).

For each Y_k sequence, all 15 variants of NIST statistical tests (divided in 188 tests) were applied. Each test provides a p-value [12]. Similarly to other statistical tests, the NIST tests are based on hypotheses testing. Hypotheses testing is a procedure to determine whether an assumption about a particular theory is reasonable. In this case, the assumption is that a certain sequence of 0 and 1 is random [12].

To validate each statistical test and to determine its passing ratio, a significance level (α) is firstly chosen. A value $\alpha = 0.01$ of the significance level assumes that 99% of the sequences should pass the statistical test. In our experiments the significance level was fixed to 1%. By the probability estimation theory and resuming each test 1000 times, the range of acceptable proportions for each type of test can be determined by using the confidence interval defined as $(p \mp 3\sqrt{p(1-p)/m}$, where $p = 1 - \alpha = 0.99$ and $m = 1000$. Thus, the passing ratio for the Y_k sequences is restricted to the acceptance interval $[0.9806, 0.9994]$.

Note: In this paper only 15 of the 188 NIST statistical tests are illustrated. For tests existing in several variants only the results for one randomly chosen test were provided for illustration. These categories include: Cumulative Sums, Non Overlapping Template, Random Excursions, Random Excursions Variant and Serial statistical tests.

The randomness of Y_k sequences generated using the running-key approach was investigated using the NIST statistical test suite in order to verify/validate the range of a control parameter according to *Table 3*. For these intervals we claim that Y_k sequences follow closely the fair coin model. The margins (left/right limit) of each interval were tested along with several values inside the range. The results confirmed the high quality for the provided intervals. *Table 4* illustrates one of the NIST analysis for $k = 4$ (corresponding to Y_4) and the control parameter $a = 0.4210$ and threshold $c = a$.

Table 4

NIST results for the proposed generator ($c = a = 0.4210$)

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-val.	P.R.%	Statistical Test
111	107	107	100	96	95	101	88	102	93	0.86	0.989	**Frequency**
82	92	111	110	100	90	110	101	103	101	0.51	0.989	**BlockFrequency**
115	100	103	84	117	83	103	87	110	98	0.14	0.986	**CumulativeSums**
99	80	106	85	98	122	83	104	108	115	0.04	0.992	**Runs**
94	90	104	102	97	92	98	108	112	103	0.88	0.987	**LongestRun**
110	114	88	91	96	93	106	93	100	109	0.59	0.986	**Rank**
106	90	101	113	87	107	110	88	94	104	0.51	0.989	**FFT**
100	91	109	98	116	91	103	109	92	91	0.60	0.987	**NonOverlappTemplate**
117	103	101	88	90	108	100	86	105	102	0.50	0.985	**OverlappTemplate**
122	97	108	84	100	95	88	110	103	93	0.25	0.986	**Universal**
96	100	88	102	109	107	93	117	87	101	0.53	0.987	**ApproximateEntropy**
58	54	71	64	67	51	64	65	52	59	0.65	0.998*	**RandomExcursion**
58	61	61	57	50	60	69	49	73	67	0.45	0.995*	**RandomExcursionV**
105	91	103	88	117	103	87	103	107	96	0.53	0.989	**Serial**
124	96	96	85	94	100	93	88	111	113	0.14	0.987	**LinearComplexity**

* RandomExcursions and RandomExcursionsVariant tests used fewer sequences to complete the tests

Interpretation of Table 4

The proposed chaotic generator was analyzed using the latest version of the NIST statistical test suite published on the official website of the National Institute of Standards and Technology. The result provided by NIST consists in a test report for each binary file of length $N \times m$ bits under investigation. The resulting test report contains a table that includes the following columns:

✓ STATISTICAL TEST column presents the names of the tests

✓ PROPORTION (Passing Ratio – P.R.) column represents the passing ratio of the Y_k sequences from a total of m sequences. This value should be in the range provided in this section in case of success

✓ $C_1, C_2, C_3, \cdots, C_{10}$ columns represent the number of sequences (from m) for which the p-value is located in one of the 10 disjoint intervals of length 0.1 which cover the $[0,1]$ range

✓ P-value column indicates the uniformity of the p-values obtained for each statistical test [11]. This can be considered as a P-value of p-values and along with the proportion column it represents an indicator of the statistical test success. This P-value is obtained as follows:

– The chi-square (χ^2) test is applied to measure the spreading of the p-values over $[0,1]$ using the results provided by columns C_i, ($i = 1,2, \dots 10$)

– The value $P\_value_T = igamc(\frac{9}{2}, \frac{\chi^2}{2})$ is evaluated. If $P\_value_T \geq 0.0001$, the p-values can be considered uniformly distributed in $[0,1]$.

According to NIST statistical test suite, regarding the passing ratio and the uniformity of p-values obtained after applying the statistical tests, the proposed chaotic generator is eligible to be used in pseudo-random binary data generation, with direct applications in cryptography.

4. CONCLUSIONS

In this study we applied the new running-key method on the chaotic tent map and demonstrated its utility in designing non-redundant pseudo-random number generators, alongside with the possibility to consider the initial condition and the tent map control parameter as elements in the secret key. The theoretical and experimental study was done on binary sequences generated by tent-map, for a binarization threshold equal to the control parameter. This choice of the binarization threshold (equal to tent map parameter) implied that all typical binary sequences involved in the running-key method comply with the *i.i.d.* model required by the statistical inferences used in this evaluation.

The running-key method was experimentally supported by the usual statistical methods completed with the NIST tests suite. All assessments (theoretical and experimental) provided numerical results that allow an immediate application. These quantitative results consist in providing the range of choice for the control parameter as a function of the number of summations (requested by the application) involved in the running-key method.

The running-key procedure, firstly advanced for the logistic map and considered generally valid for ergodic sources, is here completed by this new exploration on tent map.

As a final remark, the provided generator can be used in cryptographic applications where the pseudo-random binary sequences can be an important component in the enciphering key designing.

We mention that, although the running-key study on tent map was done for a particular choice of the binarization threshold, this method can be extended, with similar results, for other choices of binarization threshold.

REFERENCES

1. A. Vlad, A. Ilyas and A. Luca, Unifying running-key approach and logistic map to generate enciphering sequences, *Annals of Telecommunications,* **vol. 68**, p. 179–186, 2013.
2. A. Vlad, A. Ilyas and A. Luca, A closer view of running-key cipher on natural languages, *Proceedings of the Romanian Academy, Series A*, **vol. 13**, Number 2/2012, pp. 157–166, 2012
3. C. E. Shannon, Prediction and Entropy of Printed English, *Bell Syst. Tech. J.,* **vol. 30**, p. 50–64, 1951.
4. C. E. Shannon, Communication Theory of Secrecy Systems, *Bell Syst. Tech. J.,* **vol. 28**, p. 656–715, 1949.
5. C. E. Shannon, A Mathematical Theory of Communication, *Bell Syst. Tech. J.,* **vol. 27**, p. 379–423, 623–656, 1948.
6. W. Diffie and M. Hellman, Privacy and Authentication: An Introduction in Cryptography, *Proc. IEEE,* **vol. 67**, p. 397–426, 1979.
7. A. Luca, A. Ilyas and A. Vlad, Generating Random Binary Sequences using Tent Map, in Proc. 10th International Symposium on Signals, Circuits and Systems, (ISSCS), Iasi, Romania, pp. 81–84. June 30–July 1, 2011.
8. A. Ilyas, A. Luca and A. Vlad, A study on binary sequences generated by tent map having cryptographic view, in Proc. 9th International Conference on Communications (COMM), Bucharest, pp. 23–26, June 2012.
9. D. Arroyo, G. Alvarez, J. M. Amigo and S. Li, Cryptanalysis of a family of self-syncronizing chaotic stream cipher, Commun Nonlinear Sci Numer Simul **16***,* pp. 805–813, 2011.
10. A. Spataru, Fondements de la theorie de la transmission de l'information, Lausanne: Presses Polytechniques Romandes, 1987.
11. J. Devore, Probability and Statistics for Engineering and the Sciences, 2nd ed., Monterey, California: Brooks/Cole Publishing Company, 1987.
12. Runkin *et al.*, Statistical test suite for random and pseudo random number generators for cryptographic applications, *NIST special publication,* **Vols. 800-22**, 2010.

Received May 20, 2013