INFORMATICA, 2013, Vol. 24, No. 2, 181–197

A New Pseudo-Random Number Generator Based on Two Chaotic Maps

Michael FRANCOIS[1], Thomas GROSGES[1] *, Dominique BARCHIESI[1], Robert ERRA[2]

[1]*Project Group for Automatic Mesh Generation and Advanced Methods, Gamma3 Project, University of Technology of Troyes*
12 rue Marie Curie, CS 42060, 10004 Troyes Cedex, France
[2]*Network & Information Security, Ecole Supérieure d'Informatique, Electronique, Automatique*
9 rue Vésale, 75005 Paris, France
e-mail: thomas.grosges@utt.fr

Received: September 2011; accepted: September 2012

Abstract. A new pseudo-random number generator (PRNG) is proposed. The principle of the method consists in mixing chaotic maps produced from an input initial vector. The algorithm uses permutations whose positions are computed and indexed by a chaotic function based on linear congruences. The performance of this scheme is evaluated through statistical analysis. Such a cryptosystem lets appear significant cryptographic qualities for a good security level.

Keywords: chaotic function, permutation, linear congruence, pseudo-random, cryptography.

1. Introduction

Nowadays, telecommunication networks and especially the internet have become common tools in everyday of life. From simple data transmission to a high sensitive data storage, the need to develop methods to secure these transactions has become a concern for many researchers. Thus, all these constraints made that the randomness in data processing has become a key element for security. The generation of pseudo-random numbers is a subject which continue to impassion the researchers since a few decades. Indeed, a pseudo-random number generator (PRNG) is defined as an algorithm enabling to generate a sequence of numbers with properties of randomness. The design of the generator is important in the choice of its application. One way to design such a pseudo-random number generator is connected to the chaos theory (Alvarez and Li, 2006; Zheng *et al.*, 2008; Patidar and Sud, 2009a). Since the nineties, many researchers have shown that an interesting relationship between cryptography and chaos exists. These chaotic systems are characterized by their high sensitivity to initial conditions and some properties like ergodicity and random behaviors. This extreme sensitivity to the initial conditions

*Corresponding author.

makes chaotic system very important for cryptographic applications and developments of pseudo-random number/bit generator (Po-Han *et al.*, 2004; Cecen *et al.*, 2009; Orúe *et al.*, 2010; Patidar and Sud, 2009b; Pareek *et al.*, 2010; Guyeux *et al.*, 2010). Indeed, sequences of numbers presenting properties of randomness can always be produced but a rigorous mathematical analysis is necessary to evaluate the level of randomness and the efficiency of the generator.

In this paper an algorithm using a chaotic function is developed for the generation of multiple pseudo-random sequences. The algorithm uses permutations whose positions are computed and indexed by a chaotic function based on linear congruences. These chaotic permutations are achieved iteratively on this inital vector to produce two chaotic maps. These maps are xored in order to produce one sequence which is the output of the algorithm. The choice of producing and xoring two chaotic maps enlarges the complexity of the system and increases the difficulty for an attacker to extract sensitive informations from the outputs. The outputs produced by our generator can be used in several secure cryptographic applications or even as simple random sequenses for numerical simulations.

This paper is structured as follows. The description of the method as well as the chaotic function analysis are given in Section 2. Section 3 presents the statistical analysis applied on a set of generated pseudo-random sequences. The security analysis of the generator is achieved in Section 4, before concluding.

2. The Proposed Cryptosystem

The core of the PRNG algorithm is based on the contruction of two chaotic maps obtained by permuting and shuffling the positions of an initial input vector. The permutation function is inspired by recurrence functions used for linear congruential generators (Janke, 2002). Here, the chaotic function is constructed and used to compute the positions according to an initial vector. These positions are shuffled during the generation process through an internal modulo operator which is extended to the whole size of the vector. In order to initate the permutation process, an input vector V_{in} of size N and a starting seed are necessary. The function uses a degressive modulo C related to the size of the input vector and is defined by the following recurrence relation:

$$X_{n+1} = \big[[X_n^2 \bmod C] \times X_n + X_{\text{init}}\big] \bmod C, \tag{1}$$

where $X_0 = g$, $X_{\text{init}} = g^2$, the seed value $g \in \{1, \dots, M\}$ with $M = N \times \log_2 N$. The space of the seed values g is limited to M in order to avoid any problem of periodicity. The value of C is initialized to $M - 1$ and decremented after each iteration. The positions are computed with a memory effect. Indeed the new computed positions X_n represent the values of the old positions already shuffled. So, the value of a position can move several times before fixing. The principle of the proposed chaotic function is to shuffle the initial positions $1, \dots, M$ by starting of an initial seed value X_0. Before giving the general algorithm which constitutes the PRNG, we characterize the chaotic behavior of the function of (1) in the following (see Section 2.1).

2.1. *Chaotic Behaviour of the Function of Recurrence*

The chaotic behaviour of the chaotic function given by (1) is characterized and analysed through their Lyapunov exponents. For dynamical systems, the Lyapunov exponent characterizes the velocity of evolution between two near trajectories and is given, for discrete dynamical system, by Wolf *et al.* (1985), Aurell *et al.* (1997):

$$\lambda(x_0) = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \ln |f'(x_i)|, \tag{2}$$

with $x_{n+1} = f(x_n)$, x_0 and $x_0 + \varepsilon$ being two near initial conditions. A quantitative measurement of chaotic behaviour of the function is given by the positive values of Lyapunov exponent (i.e., $\lambda > 0$). Here, the initial conditions are corresponding to the seed values g and the Lyapunov exponents between the suites of positions generated for two consecutive seeds are computed. In fact, the function in (1) is not differentiable and an evaluation of its derivative is obtained by finite difference. With such an approximation, the value of Lyapunov exponent λ is given by:

$$\lambda(g) = \frac{1}{M} \sum_{i=1}^{M} \ln \left| \frac{V_g[i] - V_{g'}[i]}{g - g'} \right|, \tag{3}$$

where V_g and $V_{g'}$ correspond to position vectors generated by the seeds g and g', respectively. The seeds are $g_i = i$, $1 \leqslant i \leqslant M$, $M = 10\,240$, the Lyapunov exponents $\lambda(g_i)$ are computed for two near seed values (g_{i-1}, g_i) and shown in Fig. 1(a). All computed Lyapunov exponents are positive and belong to the interval [7.69, 7.77]. Figure 1(b) shows an example of computed positions, for two near seeds $g_1 = 9717$, $g_2 = 9718$, as function of the position n. One can remark that the two trajectories are very different and exhibit the sensitivity related to the initial seeds. The value $\log |X_n - X'_n|$ as function of the position n for these two trajectories, is also presented (see Fig. 1(c)). Finally, for a chosen seed value $g = 977$, the evolution of the Lyapunov exponent values as function of M is also computed and presented in Fig. 1(d). All the Lyapunov exponents are positive and confirm the chaotic behaviour of the analysed function based on linear congruences. Let us turn to the whole algorithm of pseudo-random number generation.

2.2. *Description of the Generator*

Such a chaotic function is integrated in the pseudo-random number generator and the algorithmic principle of the method consists in four steps:

1. An initial vector V_{in} of size N is selected (this choice is free). This vector V_{in} is transformed into a binary vector $V_{\text{in}}^{\text{bin}}$ obtained by the binary sequences of V_{in} taken in sequential order. Thus $V_{\text{in}}^{\text{bin}}$ contains only the values 0 and 1 and its size is $M = N \times \log_2 N$ (preferably take $N = 2^x,\ x \gg 1$).
2. A seed $g \in \{1, \dots, M\}$ is chosen as initiator of the relation of recurrence given by (1).

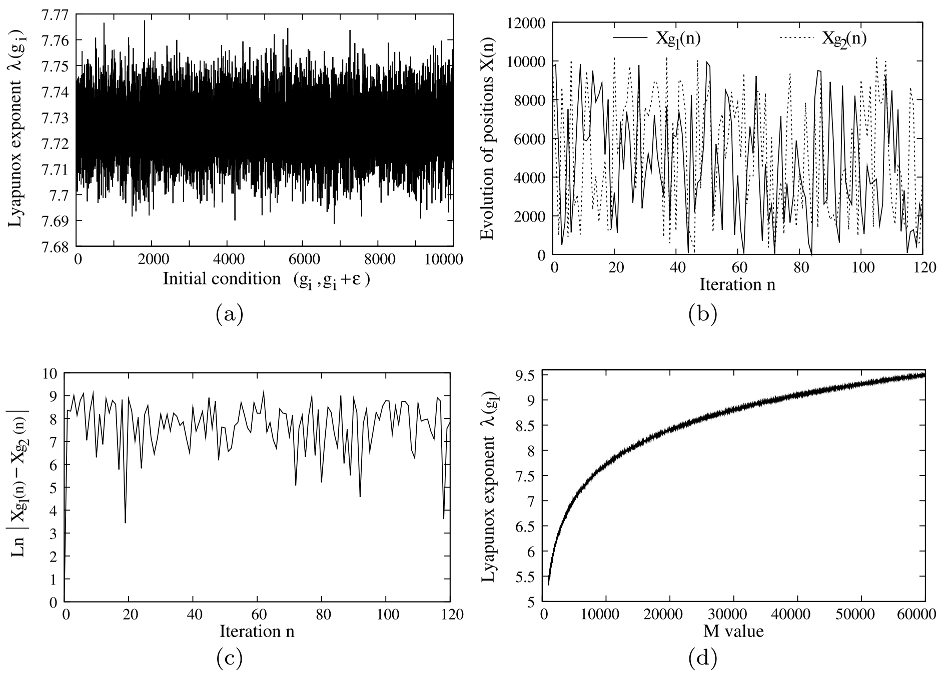


Fig. 1. Behaviour of the function of (1): (a) Lyapunov exponent between generated suites for two consecutive seeds $(g_{i-1}, g_i), 1 \leqslant i \leqslant M = 10\,240$. (b) Sensitivity on the initial conditions as function of the iteration step. (c) log difference between two sequences generated by the seeds $g_1 = 9717$, $g_2 = 9718$ and (d) Lyapunov exponent as function of M for a fixed seed value $g = 977$.

3. Do loop for $V_{\text{in}}^{\text{bin}}[i]$, where i is the current position in $V_{\text{in}}^{\text{bin}}$ and construction of a second vector component $V_{\text{in}}^{\text{bin}}[j]$ in a new chaotic position $j = i + 1 + X_{i+1}$ with (1). The elements of $V_{\text{in}}^{\text{bin}}$ are transformed to $V_{\text{in}}^{\text{bin}}[i] = Z_3$ and $V_{\text{in}}^{\text{bin}}[j] = Z_1$ with

$$Z_1 = V_{\text{in}}^{\text{bin}}[i], \tag{4}$$

$$Z_2 = V_{\text{in}}^{\text{bin}}[j] = V_{\text{in}}^{\text{bin}}(i + 1 + X_{i+1}), \tag{5}$$

$$Z_3 = Z_1 \oplus Z_2, \tag{6}$$

where the symbol $\oplus$ represents the exclusive OR operation bit-by-bit. This process is achieved until the end of the loop.

4. The bits of $V_{\text{in}}^{\text{bin}}$ are gathered per package of $\log_2 N$ in order to obtain the new vector V_1 of size N. This constitutes the steps for one round in the vector V_{in} using the seed g.

The construction of the two chaotic maps consists in applying iteratively L and T times the algorithm on the vector V_{in} in order to transform it into V_L and V_T, respectively. The vectors V_L (with $L = \text{Floor}[T/2]$) and V_T (i.e., produced by applying $T - L$ rounds from V_L) are two different chaotic vectors which are combined to produce the output vector:

$$V_{\text{out}} = V_L \oplus V_T, \tag{7}$$

which is the output pseudo-random sequence of size N. It should be noted that the generator consists in mixing permutation of positions through a xoring operation on the binary elements of V_{in} for positions i and j. Moreover, in order to improve the security of the algorithm, the sequence V_{out} is obtained by xoring V_L and V_T making the algorithm irreversible. As we should store two vectors of size N at most, the memory space complexity is $\mathcal{O}(N)$.

In general case, the algorithmic processes for generating pseudo-random numbers are interesting for the following reasons:

1. Repeatability: for chosen (and stored) seeds, a fully reproducing sequence of pseudo-random numbers can be assured. That may be important in simulation codes.
2. Usefulness: fast generation without limitation for a vast majority of implementations.
3. Standardization: reliability for quality and efficiency.

2.3. *Round Number Selection T*

A fundamental advantage of any kind of PRNG is the quality of keys. According to the Kerckhoffs' principle (Menezes *et al.*, 1996), the security of a cryptosystem only depends on its keys. In any cryptosystem, a poor key or a limited key space $\mathcal{K}$ induces a weakness of the cryptosystem (i.e., which can be easily broken). Indeed, given today's computer speed, it is generally accepted that a key space of size smaller than 2^{128} is not secure enough (Janke, 2002). The present algorithm consists in producing pseudo-random vectors from keys. Each key is corresponding to two input data: (1) an initial vector V_{in} of size N (or M in bits) and (2) a set of seed values $g_i, 1 \leqslant i \leqslant T$. At each round, each seed g_i has a value in the interval $\{1, \ldots, M\}$. Therefore, the total number of possibility for the seed space is M^T. In order to satisfy the condition $M^T > 2^{128}$, avoiding any brute-force attack, the minimum number of rounds T_1 is:

$$T_1 = \text{Floor}\left[\frac{128}{\log_2 M}\right] + 1, \tag{8}$$

with Floor$[x]$ is the largest integer not greater than x. This condition $T = T_1$ assures the minimum entropy limit for the seed space but not necessarily the randomness quality of the output vector V_{out}. Indeed, an initial vector V_{in} with very low Shannon's entropy can necessitate $T \geqslant T_1$. Therefore, the number of rounds T is also related to the distribution of bits '0' and '1' in V_{in}. Assuming that in V_{in}, the occurency of the bit '0' has a probability $p_0(0)$ (resp. $p_0(1) = (1 - p_0(0))$ for the bit '1'), then at each new round t, the probability $p_t(0)$ is iteratively modified by:

$$p_t(0) = [p_{t-1}^2(0) + (1 - p_{t-1}(0))^2] \quad \forall t \geqslant 1, \tag{9}$$

and the limit of the suite $p_t(0)$ must converge to 0.50 to assure a maximum Shannon's entropy in the output vector V_{out}. The goal is to find the number of rounds T_2 satisfying the relation:

$$\lim_{t \to T_2} p_t(0) = 0.50 - \varepsilon_1, \tag{10}$$

with ε_1 a fixed numerical tolerance (here $\varepsilon_1 = 0.001$). The value T_2 can be large for V_{in} with very low Shannon's entropy or small for V_{in} with Shannon's entropy closed to its maximum (i.e., 1 in base 2). Finally, the algorithm must also assure high sensitivity to the inital input vector V_{in}. Indeed, by using the same set of seed values with two near initial vectors V_{in} and V'_{in}, the security level can not be maximum if sensitivity is not guaranteed and will produce high correlation between the output vectors V_{out} and V'_{out}. In order to avoid this situation, an additional hypothesis must be taken into account. By considering two initial input vectors V_{in} and V'_{in} of size M (in bits) and differing by only one bit, the probability s_0 of identical elements between these two vectors is equal to $s_0 = (M-1)/M$. This probability decreases according to the number of rounds as:

$$s_r = s_{r-1}^2 \quad \forall r \geqslant 1, \tag{11}$$

and must satisfy:

$$\lim_{r \to T_3} s_r \leqslant \varepsilon_2, \tag{12}$$

where ε_2 is the acceptable criterion of similitude between binary sequences (e.g., ε_2 is fixed to 0.005, assuming a rate of identical bits smaller than 0.5%). The minimum number of round T_3 satisfying the (12) is:

$$T_3 = \text{Floor}\left[\log_2\left(\frac{\ln(\varepsilon_2)}{\ln(s_0)}\right)\right] + 1. \tag{13}$$

With these three indicators T_1, T_2 and T_3, the number of rounds is given by:

$$T = \max\{T_1, T_2, T_3\}. \tag{14}$$

This number of rounds T is automatically computed from the knowledge of the input vector V_{in} and permits to satisfy simultaneously the criteria of key space entropy, maximum Shannon's entropy and sensitivity to initial conditions (initial input vector and seeds). As an example, for an initial vector of size $N = 1024$ (i.e., $M = 10\,240$, $T = 16$), the cryptosystem permits to produce exactly $10\,240^{16} (\simeq 2^{213})$ different pseudo-random sequences. For the generation of an unspecified pseudo-random sequence, the parameters which must remained secret are:

(1) the arbitrary initial input vector V_{in} of size N;
(2) the set of T seed values $\{g_1, \dots, g_T\}$ with each seed value $g_i \in \{1, \dots, M\}$.

3. Statistical Analysis

The purpose of this section is to present the approaches which are used to analyse the qualities of the produced pseudo-random sequences. These qualities are investigated following both aspects: randomness properties of each individual sequence and correlation between multiple sequences. In order to achieve this objective, two approaches are proposed.

3.1. *Approach 1*

This first approach consists in evaluating the randomness quality of the sequences V_{out} produced by the algorithm. Therefore, the sequences are evaluated through statistical tests suite NIST (National Institute of Standards and Technology of the U.S. Government). This suite consists in a statistical package of fifteen tests developped to quantify and to evaluate the randomness of (arbitrarily long) binary sequences produced by either hardware or sotware based cryptographic random or pseudorandom number generators (Rukhin *et al.*, 2010). For each statistical test, a set of p_{value} is produced and is compared to a fixed significance level $\alpha = 0.01$ (i.e., only 1% of the sequences are expected to fail). Therefore, a sequence passes a statistical test for $p_{\text{value}} \geqslant \alpha$ and fails otherwise. In case of testing multiple sequences at the same time, each test define a proportion η as the ratio of sequences passing succesfully the test relatively to the total number of sequences N_k (i.e., $\eta = n[p_{\text{value}} \geqslant \alpha]/N_k$). This proportion η is compared to an acceptable proportion η_{accept} which corresponds to the ratio of sequences which should pass the test (Rukhin *et al.*, 2010). These NIST tests are achieved on the three following kind of sequences: individual sequences, concatened sequence, and modified sequences.

1. Individual sequences: The randomness quality of each sequence belonging to a subset of sequences is analysed directly by NIST tests. The N_k sequences V_{out}^k of binary size M (with $1 \leqslant k \leqslant N_k$) are individually tested and the results are given as ratio of success compared to a fixed threshold. The provided information is the randomness of each sequence.
2. Concatened sequence: A new sequence is constructed by concatening all the individual sequences: $V_{\text{cat}} = \{V_{\text{out}}^1, \dots, V_{\text{out}}^{N_k}\}$ of binary size $N_k \times M$. The randomness quality of this new sequence is also analysed directly by the NIST tests. The provided information is the randomness of the concatened sequence and the binary internal correlation.
3. Modified sequences: M modified sequences V_{mod} of size N_k are constructed by collecting for each position $1 \leqslant j \leqslant M$ the binary value of each sequence V_{out}^k. Therefore, $V_{\text{mod}}^j = \{V_{\text{out}}^1[j], \dots, V_{\text{out}}^{N_k}[j]\}$. The NIST tests are applied on such modified sequences in order to analyse the correlation between the N_k produced sequences. This analyse provides informations on the hidden linear structure of a group of sequences produced with near seed values.

3.2. *Approach 2*

The purpose of this second approach is to check the correlation between the produced pseudo-random number sequences. In contrast to previous approach (Approach 1.2), the correlation between sequences are analysed globally by computing the Pearson's correlation coefficients of each pair of sequences (Cheng *et al.*, 2004). Let the two sequences $k_x = [x_1, \dots, x_N]$ and $k_y = [y_1, \dots, y_N]$, we have:

$$C_P = \frac{\sum_{i=1}^{N}(x_i - \overline{x}) \cdot (y_i - \overline{y})}{[\sum_{i=1}^{N}(x_i - \overline{x})^2]^{1/2} \cdot [\sum_{i=1}^{N}(y_i - \overline{y})^2]^{1/2}}, \tag{15}$$

where $\overline{x} = \sum_{i=1}^{N} x_i/N$ and $\overline{y} = \sum_{i=1}^{N} y_i/N$ are the mean values of k_x and k_y, respectively. A strong correlation occurs between two sequences for $C_P \simeq \pm 1$ and no correlation or absence of monotonic association corresponds to $C_P \approx 0$. These Pearson's correlation coefficients are computed for each pair of sequences and the distribution of their values is presented by a histogram. The Pearson's correlation coefficients are also compared with the nonparametric correlation estimators as the Kendall and Spearman correlation which are known to be resistant to outlying observations and non normal distribution data (Taylor, 1987; Croux and Dehon, 2010). Spearman's correlation coefficient C_S varies from -1 to $+1$ and the absolute value of C_S describes the strength of the monotonic relationship. Similar to Spearman's rank-order correlation coefficient, Kendall's C_K correlation coefficient is designed to capture the association between two ordinal (not necessarily interval) variables.

3.3. *Analysis of a Subspace of Pseudo-Random Sequences*

In this section, a subspace of sequences is considered and analysed. These sequences are produced from consecutive seed values belonging to the total seed space $\mathcal{K}$. This analysis puts forward the quality of the outputs produced following a change of consecutive seeds. For the generation, we consider the two following input data: one initial input vector $V_{\text{in}} = [0, \dots, 1023]$ of size $N = 1024$ (i.e., $M = 10\,240$, and $T = 16$) and the sequences of T seed values $\{g_1, \dots, g_{16}\}$ with $\{g_1, \dots, g_{14}\} = \{$10, 1984, 3001, 47, 10 115, 489, 3420, 5019, 137, 9612, 2089, 737, 7437, 89$\}$, $g_{15} \in \{1, \dots, 4\}$ and $g_{16} \in \{1, \dots, 10\,240\}$. The total number of generated sequences is $N_k = 4 \times 10\,240 = 40\,960$, each sequence has a size of 10 240 bits and the seed values g_i for $1 \leqslant i \leqslant 14$ are chosen in $\{1, \dots, 10\,240\}$.

3.3.1. *Analysis with Approach 1*

The results obtained by Approach 1 (1–3) on the $40\,960$ sequences, are given in Table 1. For the tests "NonOverlappingTemplate", "RandomExcursions" and "RandomExcursionsVariant" the smallest percentage of all subtests are presented. Due to the unadapted size of the individual sequences (only $10\,240$ bits per sequence in contrast to $419\,430\,400$ bits for the concatened sequence and $40\,960$ bits for each modified sequence), these last

Table 1

Results of the NIST tests using Approach 1 on the 40 960 generated sequences for individual, concatened and modified sequences. The ratio η of p_{value} concerns individual and modified sequences while the p_{value} concerns the concatened sequences

Test name	Indiv. η	Seq. V_{out} result	Concat. p_{value}	Seq. V_{cat} result	Modif. η	Seq. V_{mod} result
Frequency	99.03	Success	0.6936	Success	98.82	Success
Block-Frequency	99.01	Success	0.7740	Success	99.02	Success
Cumulative Sums (1)	99.06	Success	0.4846	Success	98.14	Success
Cumulative Sums (2)	99.02	Success	0.2366	Success	98.04	Success
Runs	98.94	Success	0.7489	Success	99.21	Success
Longest Run	98.99	Success	0.1637	Success	98.33	Success
Rank	99.10	Success	0.7278	Success	98.92	Success
FFT	98.95	Success	0.6470	Success	98.82	Success
Non-Overlapping	97.93	Success	0.0401	Success	98.40	Success
Overlapping	97.32	Success	0.1916	Success	99.02	Success
Universal	–	Success	0.3965	Success	98.43	Success
Approximate Entropy	98.58	Success	0.2101	Success	98.73	Success
Random Excursions	–	Success	0.2938	Success	98.10	Success
Random E-Variant	–	Success	0.0633	Success	98.10	Success
Serial (1)	98.85	Success	0.1567	Success	99.02	Success
Serial (2)	98.96	Success	0.2624	Success	99.02	Success
Linear Complexity	98.46	Success	0.2187	Success	98.73	Success

two tests and the "Universal" test are not applicable. We notice that the results of the tests are satisfactory for the whole set of tested outputs. The sequences pass successfully the NIST tests for individual sequences, for the constructed concatened sequence and for the modified sequences. These results show the quality of the produced sequences with the pseudo-random number generator.

3.3.2. *Analysis with Approach 2*

The coefficients of correlation between each pair of the $40\,960$ generated sequences are computed with (15) and the distribution of these coefficients C_P is presented in Fig. 2(a). The histogram shows that the computed coefficients are very close to 0 and included in the interval $[-0.04,\ 0.04]$. The absolute value of each coefficient is smaller than 0.04 and most than 98.48% of the coefficients have an absolute value smaller than 0.025. That clearly shows a weak correlation between the sequences.

Moreover, these Pearson's correlation coefficients are also compared with the non-parametric Kendall's C_K and Spearman's C_S correlation estimators (see Fig. 2(b)). We can remark that these three indicators (Pearson C_P, Spearman C_S and Kendall C_K) have similar values and the absolute values of Kendall's coefficients are smaller than Pearson's coefficients.

The outputs of PRNG must have both strong quality of randomness and strong in-

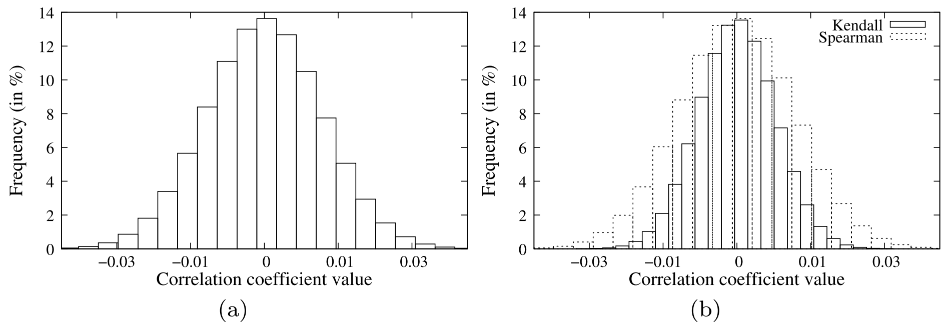


Fig. 2. Distribution of (a) the Pearson's coefficients C_P on the interval $[-0.04,\ 0.04]$, for the tested keys and (b) comparison with Spearman C_S and Kendall C_K correlation coefficients.

dependence between these outputs. The results of the analysis obtained with the two approaches, show the randomness level of the pseudo-random sequences and the quasi independence that may exist between a group of sequences produced with the PRNG.

4. Security Analysis

The security analysis of any PRNG should also be evaluated against attacks as well as its application domain. Therefore, the analysis must take into account all the critical points of the cryptosystem and must meet cryptographic requirements (Alvarez and Li, 2006). In the present case, the investigated points are: the size of the key space, the key sensitivity, the key choice, the randomness quality of the ouputs and weak key generation. These points are investigated through the three following attacks: Guess-and-Determine Attack (Ahmadi and Eghlidos, 2009), Distinguishing Attack (Coppersmith *et al.*, 2002) and Differential Attack (Biham and Shamir, 1993).

4.1. *Key Space*

A good generator of (pseudo-)random sequences should have a large key space in order to make brute-force attacks infeasible. It is generally accepted that a key space of size smaller than 2^{128} is not secure enough (Janke, 2002). Here, the key space is constructed by taking into account this constraint and permits to generate M^T distinct key sequences, with $T = \max\{T_1, T_2, T_3\}$ (see (14)). By example, for $N = 1024$ (i.e., $M = 10\,240$ and $T = 16$), the algorithm enables to produce about 2^{213} pseudo-random sequences. Therefore, the key space size is large enough to resist brute-force attacks. Such a large space of keys is a necessary condition, but not sufficient. Indeed, all the produced keys must also be cryptographically strong and uncorrelated.

4.2. *Key Sensitivity*

The sensitivity on the key is an essential factor for the pseudo-random generation based on chaos. Indeed, only a small deviation in the input should cause a large change in the output. Here, the key is given by two kinds of inputs: a set of seed values k_{in} and one initial vector V_{in}. Therefore, the key sensitivity analysis must be achieved on these two inputs.

4.2.1. *Seed Value Sensitivity*

The first part of key sensitivity concerns the seed values. Actually in the test of correlation (Section 3), this seed sensitivity was already tested due to the selected near seed values. To bring an additional response, a large pseudo-random sequence V^a_{out} of size $N = 4\,194\,304$ (i.e., $M = 92\,274\,688$ and $T = 29$) is produced. For this output pseudo-random sequence V^a_{out}, the input initial vector is $V_{\mathrm{in}} = [0, \dots, 4\,194\,303]$ and the set of seed values $k_a = \{g_1, \dots, g_{28}, g^a_{29}\}$ is $\{1984, 194, 21\,294, 1\,299\,314, 12\,314, 10, 74\,120, 1\,230\,014, 951\,210,\ 194, 70\,553, 2835, 19\,800,\ 9\,299\,314,\ 83\,721,\ 610\,990,\ 2120,\ 65\,521,\ 39, 1\,239\,094,\ 9\,230\,014,\ 16\,630\,010, 324, 19\,201, 75\,245, 365\,257, 820\,0014, 10\,000, 745\,309\}$. Moreover, with the same input initial vector V_{in}, two slightly differing sets of seeds k_b and k_c are also considered with $k_b = \{g_1, \dots, g_{28}, g^b_{29}\}$ and $k_c = \{g_1, \dots, g_{28}, g^c_{29}\}$ to produce V^b_{out} and V^c_{out}, respectively. These two sets differ by only one bit in the last seed value $g^a_{29} = 745\,309$ (i.e., $g^b_{29} = 745\,308$ and $g^c_{29} = 745\,310$ in the sets k_b and k_c, respectively). The analysis of the sensitivity to the seed value consists in computing the correlation coefficients between the produced sequences V^a_{out}, V^b_{out} and V^c_{out}. The results are given in Table 2. These coefficients are closed to 0 and the tested sequences are very different. Moreover, in order to illustrate such a sensitivity to the seed value, a new sequence is also constructed and is corresponding to the difference between two pseudo-random sequences: $V^{\mathrm{diff}}_{\mathrm{out}} = |V^a_{\mathrm{out}} - V^b_{\mathrm{out}}|$. This $V^{\mathrm{diff}}_{\mathrm{out}}$ is projected in 256 gray-levels and the associated image is presented in Fig. 3(a). If the two sequences V^a_{out} and V^b_{out} are random, the new one $V^{\mathrm{diff}}_{\mathrm{out}}$ should not present any linear structure. Therefore, the correlation (Approach 2 applied on the lines of the image) is evaluated and presented in Fig. 3(b). All the coefficients are in the interval $[-0.08,\ 0.08]$ and 99.04% of coefficients have a value smaller than 0.055. The correlation between the lines of the image is weak. The results obtained here show that the sequences seem to be very different. This illustrates the sensitivity of the cryptosystem to the initial seeds.

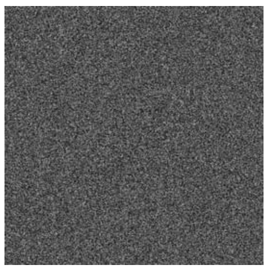
4.2.2. *Initial Vector Sensitivity*

The second part of the key sensitivity concerns the input initial vector V_{in} and the cryptosystem, to be efficient, must also be very sensitive to this last one. Here, a complete

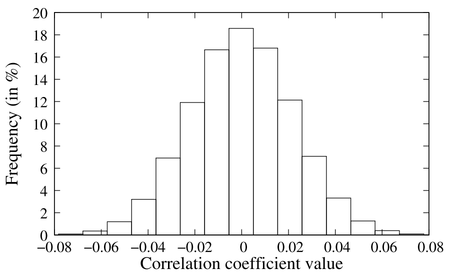
Table 2

Correlation coefficients between the three pseudo-random sequences produced with slightly different seed sets k_a, k_b and k_c

Outputs 1/2	$V^a_{\mathrm{out}}/V^b_{\mathrm{out}}$	$V^a_{\mathrm{out}}/V^c_{\mathrm{out}}$	$V^b_{\mathrm{out}}/V^c_{\mathrm{out}}$
Pearson's correlation coef.	0.000787	0.000175	0.000248
Spearman's correlation coef.	0.000786	0.000170	0.000240
Kendall's correlation coef.	−0.000010	−0.000026	−0.000005



(a)



(b)

Fig. 3. Illustration of (a) the corresponding image in gray-level of the sequence $V_{\text{out}}^{\text{diff}}$ obtained from the difference of two pseudo-random sequences produced with two slightly different seed sets k_a, k_b. (b) Correlation histogram between the lines of the image.

knowledge of the used seed values g_i $(1 \leqslant i \leqslant T)$ and only partial information on the initial vector V_{in} are considered. Let the initial vector of size $N = 4\,194\,304$ to be $V_{\text{in}}^a = [0, 1, 2, \ldots, 4\,194\,303]$ and the set of seed values $k_{\text{in}} = k_a$ given previously. With such a fixed k_{in}, two additional initial vectors V_{in}^b and V_{in}^c differing from V_{in}^a by only one bit are also considered (e.g., $V_{\text{in}}^b = [1, 1, 2, \ldots, 4\,194\,303]$ and $V_{\text{in}}^c = [0, 0, 2, \ldots, 4\,194\,303]$). If the cryptosystem is sensitive to the initial input vector then the produced outputs V_{out}^a, V_{out}^b and V_{out}^c should also be very different and not correlated. The correlation coefficients between these three produced sequences are computed and presented in Table 3. With such a difference of only one bit in the initial vector, the produced sequences V_{out}^a, V_{out}^b and V_{out}^c are very different and not correlated. To illustrate the sensitivity to the initial vector, a sequence corresponding to the difference between the two sequences V_{out}^a and V_{out}^b is also constructed and is given by $V_{\text{out}}^{\text{diff}} = |V_{\text{out}}^a - V_{\text{out}}^b|$. This sequence $V_{\text{out}}^{\text{diff}}$ is projected in 256 gray-levels and the associated image is presented in Fig. 4(a). Moreover, the correlation (Approach 2 applied on the lines of the image) is computed and presented in Fig. 4(b). All the coefficients are into the interval $[-0.08,\ 0.08]$ and 99.16% of them have an absolute value smaller than 0.057. The lines of the image present a weak correlation. The results show that the sequences are very different and the high sensitivity of the cryptosystem related to the initial input vector.

Table 3

Correlation coefficients between the three pseudo-random sequences produced with slightly different initial input vectors V_{in}^a, V_{in}^b and V_{in}^c with the set $k_{\text{in}} = k_a$

Outputs 1/2	$V_{\text{out}}^a/V_{\text{out}}^b$	$V_{\text{out}}^a/V_{\text{out}}^c$	$V_{\text{out}}^b/V_{\text{out}}^c$
Pearson's correlation coef.	0.000050	0.000066	-0.000155
Spearman's correlation coef.	0.000051	0.000086	-0.000105
Kendall's correlation coef.	0.000038	0.000002	0.000020

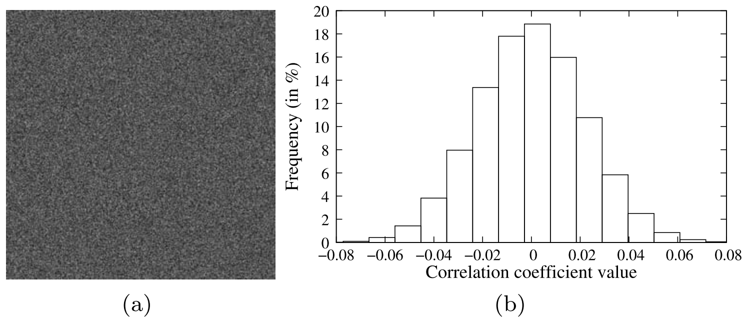


Fig. 4. Illustration of (a) the corresponding image in gray-level of the sequence $V_{\text{out}}^{\text{diff}}$ obtained from the difference of two pseudo-random sequences produced with two slightly different initial vectors V_{in}^{a}, V_{in}^{b} and (b) correlation histogram between the lines of the image.

4.3. *Choice of the Key: Initial Vector and Seeds*

The choice of the input set of seeds k_{in} and initial vector V_{in} is free but must not be neglected. As an example, the initial vector $V_{\text{in}} = [0, \dots, N-1]$ was chosen to illustrate that a high randomness quality can be obtained even with a simple vector. For a maximum security level, both the set of seeds $k_{\text{in}} = \{g_1, \dots, g_T\}$ and V_{in} are parts of the secret key and can come from a complex "source" (e.g., complex function, complex image, complex physical process,..). The idea is that the set k_{in} and the vector V_{in} can not be easily founded without any information about how it was created. The choice of the inputs (complex and secret) for the generation of pseudo-random sequences is an asset to the cryptosystem but the vector V_{in} and the set k_{in} need to be stored securely.

4.4. *Quality of Pseudo-Random Sequences*

A rigorous analysis would be necessary to determine the quality of a PRNG. Indeed, whichever way the cryptosystem is designed, the produced output must be strong (i.e., random, decorrelated and sensitive). In the literature, various statistical tests are available to evaluate the randomness of binary sequences. Reference test suites for PRNGs are TestU01 (Ecuyer and Simard, 2007), the NIST suite (Rukhin *et al.*, 2010), and the DieHARD suites (Marsaglia, 1996). Here, the NIST tests are used to quantify and to evaluate the randomness level of sub-sets of produced pseudo-random sequences. The correlation between such pseudo-random sequences were evaluated (see Approaches 1.2 and 2). The sensitivity to the key (seed and initial vector) was also tested. All the produced pseudo-random sequences pass successfully all the statistical analysis.

4.5. *Weak Key Generation*

It is essential to have a large space of keys but it is also necessary to wonder if all the keys are cryptographically strong and valid. It is mentioned in the suggested Rule 5 of Alvarez and Li (2006) that, the key space from which valid keys must be chosen, should be precisely determinated in order to avoid non-chaotic regions. Therefore, a careful study of the chaotic regions in the space of parameters is necessary in order to avoid weak or degenerate keys. In our case, this study is not really useful indeed, the space of seeds is already limited and optimized (per round) to avoid any problem of periodicity. Moreover, at each round, the seed value g can be selected randomly from the set $\{1, \dots, M\}$ and then achieving all these values. The different statistical tests clearly show the quality of tested sequences. Moreover, this chaotic region is very homogeneous. Therefore, the proposed PRNG does not present weak or degenerate key region.

4.6. *Guess-and-Determine Attack*

Guess-and-determine attack is a general attack on stream ciphers (Ahmadi and Eghlidos, 2009). The strategy of this attack is to guess firstly the value of few unknown variables of the cipher. Next, the remaining unknown variables are deduced by iterating the system a few times and by comparing the produced pseudo-random sequence with the original pseudo-random sequence. If these two sequences are identical, then the guessed values are correct and the cryptosystem is broken, else the attack should be repeated with new guessed values. Here, if the attacker does not possess the full initial vector V_{in} and the complete set of seed values, he can not apply the attack. By considering that V_{in} is known by the attacker, it seems that the attack discussed in reference (Ahmadi and Eghlidos, 2009) can not be applied on the proposed cryptosystem which is not of the same family of involved stream ciphers. Indeed, the internal structure of the cipher algorithm is completely different. Here the vector V_{in} is an input, after a few rounds it becomes a vector V_{out} of same size. An alternative way to apply this attack would be to guess and to fix the L chosen seeds in order to create the map V_L and to iterate the algorithm by searching the $T - L$ remaining seeds to produce V_T before V_{out}. Once all the comparisons made without success, the L input seeds are guessed again and the process is repeated until success. This process has almost the same complexity than a classic brute-force attack.

4.7. *Distinguishing Attacks*

Any output sequence of a stream cipher (or PRNG) designed for cryptographic applications, should not be statistically distinguished from a truly random sequence. In fact, distinguishing attacks described in reference (Coppersmith *et al.*, 2002), try to find traces of the distinguishing property by exploiting the weaknesses of the algorithm related to the linear and non-linear combinations. Here, the generated sequences pass successfully the standard statistical tests for randomness. In the algorithm, the permutations, which are done by using the function of (1), are crossed with a xor operator, which kills any information on linear dependance. Moreover, the only linear masking occurs when $V_{\text{in}}^{\text{bin}}[j]$ is replaced by Z_1, but as the index goes to the end of the vector then all positions are concerned by the internal xor operation. By knowing that the process is repeated T times on a long enough initial vector (which is secret), the attack becomes ineffective even inapplicable.

4.8. *Differential Attack*

This method of cryptanalysis was introduced by Biham and Shamir (Biham and Shamir, 1993). Similarily to the chosen-plaintext attack, its principle is to analyze and to exploit the effect of a small difference in input pairs on the difference of their corresponding output pairs. This technique allows to target the most probable key that was used to produce the output. Given two inputs x and x' (e.g., V_{in} and V'_{in}) and their corresponding outputs y and y' (e.g., V_{out} and V'_{out}), the most commonly used differences are:

1. The subtraction modulus which is the differences related to both inputs and outputs: $\Delta x_1 = |x - x'|$ and $\Delta y_1 = |y - y'|$;
2. The xor difference which is defined by: $\Delta x_2 = x \oplus x'$ and $\Delta y_2 = y \oplus y'$.

The diffusion aspect or the sensitive dependence on the initial condition is then measured by a differential probability. However, the proposed PRNG is designed to avoid this kind of cryptanalysis. Indeed, during the computation of the number of rounds T, this problem is solved by including the hypothesis T_3 (see Section 2.3). Therefore, the process of generation of pseudo-random sequences assures the high sensitivity to the initial vector. This is also illustrated by the example given in Section 4.2.2, highlighting the sensitivity related to three closed initial input vectors.

5. Conclusions

In this paper, a new PRNG using two chaotic maps to generate multiple key sequences was presented. Such a generator has shown its ability to produce a very large number of pseudo-random sequences which can be usefull in several cryptographic applications. The coupling of chaotic function with the xor operation, drastically disrupts the internal structure of the initial vector and induces progressively an unpredictable randomness effect. The advantages of the generator are the adaptive size of the key space, the sensitivity to the initial inputs (keys), the quality of pseudo-random sequences, the security level against several attacks, the simplicity of implementation and the portability architecture (only integers are used).

Acknowledgments. Authors thank the Centre de Calcul Intensif ROMEO2 for computational facilities, the Région Champagne-Ardennes and the Conseil Régional de l'Aube for financial supports.

References

Ahmadi, H., Eghlidos, T. (2009). Heuristic guess-and-determine attacks on stream ciphers. *Information Security, IET*, 3(2), 66–73.

Álvarez, G., Li, S. (2006). Some basic cryptographic requirements for chaos-based cryptosystems. *International Journal of Bifurcation and Chaos*, 16(8), 2129–2151.

Aurell, E., Boffetta, G., Crisanti, A., Paladin, G., Vulpiani, A. (1997). Predictability in the large: an extension of the concept of Lyapunov exponent. *Journal of Physics A: Mathematical and General*, 30(1), 1–26.

Biham, E., Shamir, A. (1993). *Differential Cryptanalysis of the Data Encryption Standard*. Springer, Berlin.

Cascais, J., Dilao, R., Noronha da Costa, A. (1983). Chaos and reverse bifurcations in a RCL circuit. *Physics Letters A*, 93(5), 213–216.

Cecen, S., Demirer, R.M., Bayrak, C. (2009). A new hybrid nonlinear congruential number generator based on higher functional power of logistic maps. *Chaos, Solitons and Fractals*, 42(2), 847–853.

Cheng, G., Mao, Y., Chui, C. (2004). A symmetric image encryption scheme based on 3D chaotic cat maps. *Chaos, Solitons and Fractals*, 21(3), 749–761.

Coppersmith, D., Halevi, S., Jutla, C. (2002). Cryptanalysis of stream ciphers with linear masking. In: *22nd Annual International Cryptology Conference, Santa Barbara, California*. *Advances in Cryptology – CRYPTO*, Vol. 2442, pp. 515–532.

Croux, C., Dehon, C. (2010). Influence functions of the Spearman and Kendall correlation measures. *Statistical Methods and Applications*, 19(4), 497–515.

L'ecuyer, P., Simard, R. (2007). TestU01: a C library for empirical testing of random number generators. *ACM Transactions on Mathematical Software*, 33(4).

Guyeux, C., Wang, Q., Bahi, J.M. (2010). A pseudo random numbers generator based on chaotic iterations: application to watermarking. *Web Information Systems and Mining*, 6318, 202–211.

Janke, W. (2002). Pseudo random numbers: generation and quality checks. *Quantum Simulations of Complex Many-Body Systems*, 10, 447–458.

Lee, P.-H., Chen, Y., Pei, S.-C. Chen, Y.-Y. (2004). Evidence of the correlation between positive Lyapunov exponents and good chaotic random number sequences. *Computer Physics Communications*, 160(3), 187–203.

Marsaglia, G. (1996). Diehard: a battery of tests of randomness. `http://stat.fsu.edu/geo/diehard.html`.

Menezes, A.J., Oorschot, P.C.V., Vanstone, S.A. (1996). *Handbook of Applied Cryptography*. CRC Press, Boca Raton.

Orúe, A.B., Álvarez, G., Guerra, A., Pastor, G., Romera, M., Montoya, F. (2010). Trident, a new pseudo random number generator based on coupled chaotic maps. *Computational Intelligence in Security for Information Systems, Advances in Intelligent and Soft Computing*, 85, 183–190.

Pareek, N.K., Patidar, V., Sud, K.K. (2010). A random bit generator using chaotic maps. *International Journal of Network Security*, 10(1), 32–38.

Patidar, V., Sud, K.K. (2009a). A pseudo random bit generator based on chaotic logistic map and its statistical testing. *Informatica*, 33, 441–452.

Patidar, V., Sud, K.K. (2009b). A novel pseudo random bit generator based on chaotic standard map and its testing. *Electronic Journal of Theoretical Physics*, 6(20), 327–344.

Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, Vangel, M., Banks, D., Heckert, A., Dray, J., Vo, S. (2010). A statistical test suite for random and pseudorandom number generators for cryptographic applications. *NIST special publication*, Revision 1a.

Taylor, J.M.G (1987). Kendall's and spearman's correlation coefficients in the presence of a blocking variable. *Biometrics*, 43, 409–416.

Wolf, A., Swift, J.B., Swinney, H.L., Vastano, J.A. (1985). Determining Lyapunov exponents from a tile series. *Physica*, 16(3), 285–317.

Zheng, F., Tian, X., Song, J., Li, X. (2008). Pseudo-random sequence generator based on the generalized Henon map. *The Journal of China Universities of Posts and Telecommunications*, 15(3), 64–68.

M. Francois received in 2009 a master of science degree in applied mathematics and cryptography from the Caen University. He is currently working toward the PhD degree from the University of Technology of Troyes. His current research interests are numerical and optimization methods with application to engineering and cryptographic systems.

T. Grosges is an associate professor in numerical and theoretical physics and applied mathematics at the University of Technology of Troyes. Prof. dr. Grosges conducts active research on numerical developments related to modelling the interaction of matter and light and to the optimization of numerical solvers in computer sciences and their applications. His work was published in a number of physics, applied physics, engineering and computer science academic journals, conference publications and books.

D. Barchiesi is full professor in theoretical physics, applied mathematics and statistics at the University of Technology of Troyes. He conducts active research in numerical modelling, optimization and advanced methods with application to engineering of nanotechnologies and plasmonics. The result of his research has been published in over one hundred eighty-five articles, conferences, and book chapters since 1993, in the fields of optics, electromagnetism, signal processing, operational research and finite element methods.

R. Erra is professor in computer science at the ESIEA (École supérieure d'informatique, électronique, automatique) of Paris. His area of research concerns cryptography and in particular the domains of information security, protocols attack, algorithmic number theory and the theory of complexity. His work has been published in several international papers, books and have been presented in international conferences.

Naujas pseudo atsitiktiniųskaičių generatorius besiremiantis dviem chaotiniais atvaizdavimais

Michael FRANCOIS, Thomas GROSGES, Dominique BARCHIESI, Robert ERRA

Apibrėžtas ir naudojamas pseudo atsitiktinių skaičių generatorius (pseudo-random number generator, PRNG). Metodas grindžiamas chaotinių atvaizdavimų maišymu. Algoritme naudojami statistinės analizės metodai. Tokia kriptosistema leidžia pasiekti gerą saugumo lygį.