Commun Nonlinear Sci Numer Simulat 19 (2014) 887–895

Contents lists available at ScienceDirect

Commun Nonlinear Sci Numer Simulat

journal homepage: www.elsevier.com/locate/cnsns

Pseudo-random number generator based on mixing of three chaotic maps

M. François [a,b], T. Grosges [a,*], D. Barchiesi [a], R. Erra [c]

[a] Group for Automatic Mesh Generation and Advanced Methods (Gamma3 UTT-INRIA), University of Technology of Troyes, 12 rue Marie Curie, CS 42060, F-10004 Troyes Cedex, France
[b] Equipe-projet Digits, Architectures et Logiciels Informatiques (DALI), Université de Perpignan, 52 avenue Paul Alduy, F-66860 Perpignan cedex 9, France
[c] Department of Network & Information Security, Ecole Supérieure d'Informatique, Electronique, Automatique (ESIEA), 9 rue Vésale, F-75005 Paris, France

ARTICLE INFO

Article history:
Received 27 February 2013
Received in revised form 19 August 2013
Accepted 20 August 2013
Available online 31 August 2013

Keywords:
Nonlinear chaotic function
Pseudo-random
Permutation
Cryptography
Statistical analysis

ABSTRACT

A secure pseudo-random number generator three-mixer is proposed. The principle of the method consists in mixing three chaotic maps produced from an input initial vector. The algorithm uses permutations whose positions are computed and indexed by a standard chaotic function and a linear congruence. The performance of that scheme is evaluated through statistical analysis. Such a cryptosystem lets appear significant cryptographic qualities for a high security level.

1. Introduction

The generation of pseudo-random numbers plays a critical role in large number of applications such as, numerical simulations, gaming industry, communications or cryptography [1]. A pseudo-random number generator (PRNG) is defined as an algorithm enabling to generate sequences of numbers with some properties of randomness. The main advantages of such generators are the rapidity and the repeatability of the produced pseudo-random sequences. In practice, the generation of pseudo-random numbers is not trivial and the randomness quality of the produced sequence can be essential in the choice of the application. One way to design such PRNG is connected to chaos theory [2]. That theory focuses primarily on the description of these systems that are often very simple to define, but whose dynamics appears to be very confused. Chaotic systems are characterized by their high sensitivity to initial conditions and some properties like ergodicity, pseudo-random behavior and high complexity [2]. The extreme sensitivity to the initial conditions (i.e. a small deviation in the input can cause a large variation in the output) makes chaotic system very attractive for cryptographic applications, specially for pseudo-random number generators. Therefore, several chaotic systems have been applied successfully to produce pseudo-random sequences [3–7]. Many of these systems use the same standard chaotic function to generate efficiently pseudo-random numbers [8–12].

In this paper, a new PRNG using a standard chaotic function is presented. The algorithm uses a degressive modulo to index progressively the positions of an initial vector before permuting their associated elements through the use of a xor operator. The chaotic permutations are achieved iteratively on the initial vector in order to produce three chaotic maps. These

* Corresponding author. Tel.: +33 3 25 71 84 30; fax: +33 3 25 71 56 49.
E-mail address: thomas.grosges@utt.fr (T. Grosges).

1007-5704/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.cnsns.2013.08.032

maps are xored and the resulting sequence is the output of the algorithm. The choice of producing and xoring three chaotic maps enlarges the complexity of the system and increases the difficulty for an attacker to extract sensitive informations from the outputs. The advantage of the proposed PRNG is the free choice of the input vector and the randomnes quality of the produced outputs. The paper is structured as follows, the description of the method is given in Section 2. Section 3, presents the statistical analysis applied on a set of generated pseudo-random sequences. The security analysis of the PRNG is achieved in Section 4, before concluding.

2. The generator

The PRNG algorithm is based on the construction of three chaotic maps obtained by permuting and shuffling the elements of an initial input vector. The permutations are performed using a chaotic function to scramble the used positions. The chaotic function is given by the logistic function $F(X) = \lambda X(1 - X)$ (with $3.5699 < \lambda < 4.0$) and was already used in pseudo-random number generation [8–12]. Here, we use the chaotic function with $\lambda = 3.9999$ which corresponds to a highly chaotic case [13]. To initiate the permutation process, an input initial vector I_{in} of size N and a starting seed value X_0 are necessary. The iterative form of the used chaotic function is:

$$X_{n+1} = 3.9999X_n(1 - X_n) \quad \forall n \geqslant 0, \tag{1}$$

where the starting seed X_0 is a real number belonging to $]0, 1[$. All the output elements X_n are also real numbers belonging to $]0, 1[$. Due to the term $(1 - X)$ in the Eq. (1), the distributions are symmetric about the mid point of the interval $]0, 1[$. To avoid redundancy, the starting seed X_0 must be chosen in one of the two half-intervals. Here we consider that X_0 belongs to $]0.0, 0.5[$.

2.1. Description of the generator

The proposed cryptosystem integrates the chaotic function in the core of the PRNG and the algorithmic description of the generator consists in the four following steps:

1. An initial vector I_{in} of size N is chosen (the choice is free). The vector I_{in} is transformed into the binary vector I^b_{in} (i.e., by taking the binary components in sequential order). The components of I^b_{in} have only the values 0 or 1 and the vector size is $M = N \times \log_2 N$ (with $N = 2^x$ and $x \gg 1$).
2. A seed value X_0 is chosen in $]0.0, 0.5[$ and initiates the iterative relation of Eq. (1). The choice of the seed value is arbitrary.
3. A loop is started on the index i of the vector I^b_{in}, with $0 \leqslant i < M - 2$. With the current position i in I^b_{in}, a new position j is computed using the chaotic function:

$$j = i + 1 + [\text{Floor}[\beta X_{i+1}] \bmod S], \tag{2}$$

with $\beta = 10^c$ and the value of S is initialized to $M - 1$ and decremented after each iteration. Due to the modular operation, the value $\text{Floor}[\beta X_{i+1}]$ is chosen to be greater than S. Therefore the value of c is related to the size M and is given by:

$$c = \text{Floor}[\log_{10} M] + 3. \tag{3}$$

At each iteration i, the new position j has a value $i < j < M$. The elements of I^b_{in} are transformed into $I^b_{in}[i] = Q_3$ and $I^b_{in}[j] = Q_1$ with

$$Q_1 = I^b_{in}[i], \tag{4}$$
$$Q_2 = I^b_{in}[j] = I^b_{in}[i + 1 + [\text{Floor}[\beta X_{i+1}] \bmod S]], \tag{5}$$
$$Q_3 = Q_1 \oplus Q_2, \tag{6}$$

where the symbol $\oplus$ represents the exclusive OR operation bit-by-bit. That process is achieved until the end of the loop where the last iteration corresponds to $i = M - 3$. One can remark that the new computed positions j represent the value of the old positions i already shuffled. Therefore, the value of a position can move several times before fixing.

4. The bits of I^b_{in} are gathered per package of $\log_2 N$ to construct a new vector I_1 of size N. That constitutes the steps for one round (i.e. $T = 1$) in the vector I_{in} using the seed X_0.

The construction of the three chaotic maps consists in transforming the initial vector I_{in} into I_k by applying iteratively the algorithm (i.e., $k = T/3$ and $T \bmod 3 = 0$). From I_k, a second application of k rounds is achieved to produce I_{2k} and finally I_{3k} is obtained from I_{2k} with k supplementary rounds. The three constructed vectors I_k, I_{2k} and I_{3k} are chaotic vectors and are mixed to produce the sequence I_{out} of size N:

$$I_{out} = I_k \oplus I_{2k} \oplus I_{3k}. \tag{7}$$