Multimed Tools Appl (2018) 77:13391–13417
DOI 10.1007/s11042-017-4953-x

Design and analysis of two stream ciphers based on chaotic coupling and multiplexing techniques

Ons Jallouli[1] · Safwan El Assad[1] · Maryline Chetto[2] · René Lozi[3]

Received: 4 October 2016 / Revised: 3 April 2017 / Accepted: 14 June 2017 /
Published online: 29 June 2017

Abstract In this paper, we design and implement two new stream ciphers based on Pseudo Chaotic Number Generators (PCNGs) which integrate discrete chaotic maps, namely, Piecewise Linear Chaotic Map (PWLCM), Skewtent and Logistic map. They are weakly coupled by a predefined matrix **A** for the first PCNG and they are coupled by a binary diffusion matrix **D** for the second one. Each PCNG includes a chaotic multiplexing technique that allows the enhancement of the robustness of the system. The structure is implemented with finite precision $N = 32$ bits in C language. Security performance of the proposed stream ciphers is analysed and several cryptanalytic and statistical tests are applied. Experimental results highlight robustness as well as efficiency in terms of computation time of these two stream ciphers.

Keywords Chaos-based cryptography · Stream cipher · Pseudo-chaotic number generator · Discrete chaotic maps · Coupling and multiplexing techniques

1 Introduction

Nowadays, the increasing pervasiveness of technologies concerning the Internet of Things (IoT) and the fast development of digital technologies and communication networks, have given rise to dense traffic of information (documents, images, audio, videos...). Therefore,

✉ Ons Jallouli
ons.jallouli@univ-nantes.fr

[1] IETR UMR CNRS 6164, Polytech Nantes Rue Christian Pauc, 44306 Nantes, France

[2] IRCCyN, Ecole Centrale de Nantes 1 Rue de la Noë, 44321 Nantes, France

[3] Université côte d'azur, Laboratoire J.A. Dieudonné UMR CNRS 7351, Nice Cedex 02, France

it is particularly important and essential to protect data transmission against attackers. Consequently, security of data transmission has been gaining more and more importance in the last decade and has been a subject of intense research.

In this context, a growing number of cryptographic techniques to secure transmitted information have been developed. Chaos in cryptography was introduced by Matthews in 1990s [22]. Since then, investigation on chaotic image encryption has become an active field of research due to the interesting properties of chaos such as ergodicity, sensitivity to initial conditions and parameters of the system, similarity to random behavior, and broad-band power spectrum [16]. During the last 20 years, many chaotic image encryption methods have been proposed in the literature.

Chaotic image encryption which is symmetric encryption, is classified into two types: chaotic block ciphers and chaotic stream ciphers. Chaotic block ciphers are the main symmetric key cryptosystems, as their design is related to Shannon's theory of information security based on confusion and diffusion operations. Their security properties are well studied and these ciphers encrypt the plaintext block by block.

Some block ciphers have proven to be vulnerable to certain types of attack [23, 29, 33], and others have proven to be very efficient in terms of security and computing time [7, 9, 11, 12, 35, 36].

Stream ciphers encrypt bits or samples continuously. This is achieved by masking the plaintext (XOR operation) using the keystream output of the key stream random generator, as a one-time-pad. Therefore, stream ciphers, unlike block ciphers, are suitable for applications where the plaintext length is either continuous or unknown, such as network communications. Also, apart from the security aspect, the main characteristic of a stream cipher is its speed performance on different platforms and power consumption.

In recent years, several research efforts have investigated secure stream cipher designs. Many of these have been proposed in software form, e.g., A5/1 [5], LEVIATHAN (Cisco), MUGI (Hitachi-K.U. Leuven), RC4 [14], SNOW [8], SOBER (Qualcomm) and [26]. These stream ciphers have proven to be very weak and insecure. This has incited researchers to search for new methodologies that are immune to many attacks that can be applied.

In 2004, a project under the Information Societies Technology (IST) Program of the European Network of Excellence for Cryptology (ECRYPT), called "eStream" was tasked with seeking a strong stream cipher [25]. Its goal was to give rise to a standardization of fast and secure stream ciphers. Thirty-four candidate ciphers were submitted. Only a few proposals were chosen to belong to the current official "eStream" project and the others were rejected because of security vulnerabilities or lower overall performance. Two profiles of ciphers for software and hardware implementations were defined. The first profile is oriented to software-ciphers with high throughput and is faster than the 128-bits AES-CTR. The finalist include Salsa20/12, Rabbit, HC-128, and SOSEMANUK. The second profile is oriented to hardware ciphers that are suitable for highly constrained environments and are more compact than the 80-bits AES. Finalist ciphers include Grain, Trivium and MICKEY 2.0. These ciphers were found to be secure against known attacks. However, some tangible results have been reported by newer cryptanalysis attempts for some of these ciphers (Rabbit, Salsa12, SOSEMANUK, Grain, Trivium and MICKEY2.0) [21].

A new class of chaos-based stream ciphers has emerged and seems to be robust against known attacks.

Ahmed et al., [1] published a chaos-based feedback stream cipher (ECBFSC) for image cryptosystems. The proposed stream cipher is based on the use of a logistic map and an external secret key of 256-bit. The initial conditions for the logistic map are derived using the external secret key by providing weight to its bits corresponding to their position in the key.

Liu et al., [17] designed a stream-cipher algorithm based on one-time keys and robust chaotic maps, in order to obtain high security and improve the dynamic degradation. They used the piecewise linear chaotic map as the generator of a pseudo-random key stream sequence. The initial conditions were generated by the true random number generators, the Message-Digest algorithm 5 (MD5) of the mouse positions.

In [31] Vidal et al., proposed a new fast and light stream cipher based on a hyper-chaotic dynamic system, a codifying method with a whitening technique and a non linear transformation. This stream cipher has been implemented in video-conference applications for smart phones.

In this paper, we propose and realize in an effective way two stream ciphers, based on two robust Pseudo-Chaotic Numbers Generators (PCNGs). The proposed systems are very secure, due to the use of chaotic coupling and multiplexing techniques, while having a high speed performance.

The paper is organized as follows: we describe the general scheme of a stream cipher in Section 2. The proposed structure of the two PCNGs is detailed in Section 2.1. In Section 2.1.1, we describe the architecture of the first proposed PCNG and in Section 2.1.2 the architecture of the second proposed PCNG. In Section 2.1.3, we analyze the obtained results of mapping and approximated invariant values of the two PCNGs. Section 2.1.4 presents the performance measures of the two PCNGs in terms of average generation time, average Bit Rate (BR), and average Number of Cycles needed to generate one Byte (NCpB) according to the data size. Section 3 introduces the security analysis of the two proposed stream ciphers and their speed performance. Finally, Section 4 concludes the paper and gives some perspectives for our future work.

2 Proposed chaos-based stream ciphers

Stream ciphers, as shown in Fig. 1, are a class of symmetric cryptography, along with block ciphers [15]. In order to obtain a cipher text C_i, the plain-text P_i is combined, using a XOR operation, with a random keystream used only once and having the same length as the plaintext. The keystream is produced by a PCNG having as input a secret shared key K and an initial vector IV, which must be different for each encryption round. As the PCNG is deterministic, the same keystream can be generated in the decryption. Then, one can recover the original plaintext P_i, by XORing the same keystream with the cipher text C_i.

The keystream must be random enough to ensure that if an attacker knows the keystream, he cannot recover the secret key or derive the internal state. Thus, the security of any stream cipher depends on the randomness of the keystream, therefore on the robustness of the used PCNG which is the main element of a stream cipher. Note that the same secret key and IV must be shared by the emitter and the receiver in order to encrypt/ decrypt the message sent through the communication channel and must be protected from access by others.

Several techniques have been proposed for the distribution of keys and IV. Concerning our algorithms, a symmetric key distribution is used in the generation and management of the secret keys and IV, in order to provide confidentiality and integrity of the keys. This technique is based on the use of a master key, which is infrequently used and is long lasting, and session keys which are generated and distributed for each communication between emitter and receiver [28].

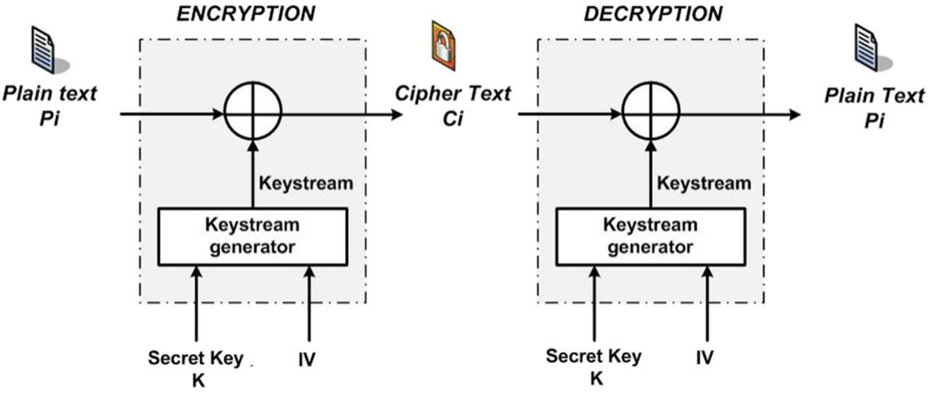


Fig. 1 General scheme of a stream cipher

In the following, we will describe in detail the general structure of the proposed PCNGs and their architectures.

2.1 Description of the general proposed structure for both PCNGs

The general structure of the proposed PCNGs is presented in Fig. 2. It takes the parameters of the system (N and the number of samples Ns), a secret key "K" and a 32-bit initial vector "IV" as input, and as output, it generates pseudo-chaotic samples $X(n)$, n=1, 2, ..., each quantified on N = 32 bits.

The structure consists of four function blocks: IV-setup, Key-setup, Internal State and Output function. Both proposed PCNGs have the same general structure but completely differ in their internal state and slightly change in their Key-setup and IV-setup. Each function block will be detailed in the architectural description of the proposed PCNGs.

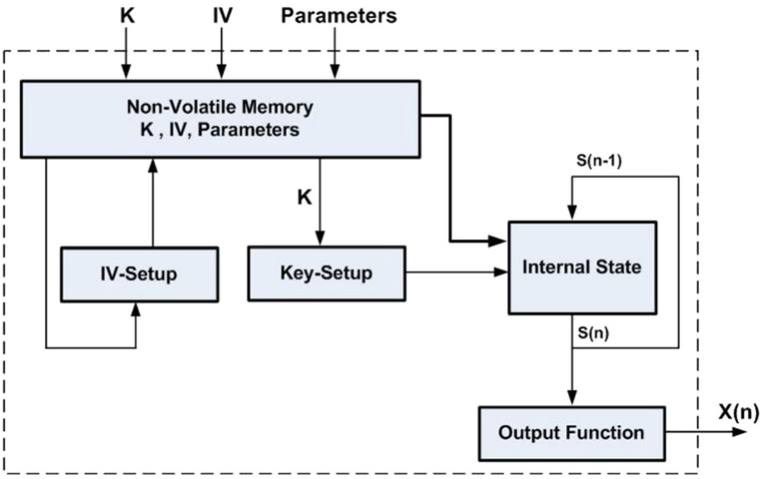


Fig. 2 General structure of the proposed PCNGs

2.1.1 Architecture of the first proposed PCNG

In this section, we describe in detail the architecture of the first proposed chaotic generator. It is given in Fig. 3. The architecture uses three weakly coupled chaotic maps: PWLCM, Skew Tent and Logistic and includes a multiplexing chaotic technique [3, 18, 19, 24, 30].

The Key-setup function consists of two main parts. It takes the secret key K and the initial vector IV as input and calculates the initial values $Xp(0)$, $Xs(0)$ and $Xl(0)$ of the three chaotic maps: PWLCM, Skewtent and Logistic respectively.

The secret key of the system is formed by:

- the initial conditions Xp, Xs and Xl of the three chaotic maps: PWLCM, Skewtent and Logistic respectively, ranging from 1 to 2^N-1,
- the control parameter Pp and Ps of PWLCM and Skewtent maps, in the range $[1, 2^{N-1} - 1]$ and $[1, 2^N - 1]$ respectively,
- the parameters of the coupling matrix **A**, ε_{ij}, ranging from 1 to 2^k with k$\leq$ 5.

All the initial conditions, parameters and initial vector are chosen randomly from Linux generator: `"/dev/urandom"`.

The initial values $Xp(0)$, $Xs(0)$ and $Xl(0)$ are calculated as follows:

$$\begin{cases} Xp(0) = Xp \oplus IVp \\ Xs(0) = Xs \oplus IVs \\ Xl(0) = Xl \oplus IVl \end{cases} \tag{1}$$

Where

$$\begin{cases} IVp = lsb(IV) \\ IVs = L_{cir}[lsb(IV), 3] \\ IVl = L_{cir}[lsb(IV), 2] \end{cases} \tag{2}$$

with $\oplus$ denotes the XOR operator, $lsb(IV)$ is the 32 least significant bits of IV and $L_{cir}[S, q]$ performs the q-bits left circular shift on the binary sequence S.

The internal state function achieves the weak coupling of the chaotic maps and produces the future samples $Xp(n)$, $Xs(n)$ and $Xl(n)$ from which the output function, by using a chaotic switching technique, produces the output sequence $X(n)$ (see Fig. 3).

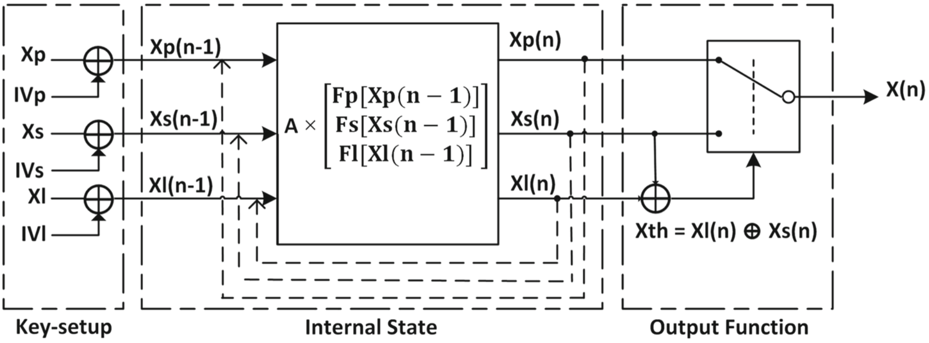


Fig. 3 Architecture of the first proposed PCNG

The system is governed by the following equation :

$$\begin{bmatrix} Xp(n) \\ Xs(n) \\ Xl(n) \end{bmatrix} = \mathbf{A} \times \begin{bmatrix} Fp[Xp(n-1)] \\ Fs[Xs(n-1)] \\ Fl[Xl(n-1)] \end{bmatrix}. \tag{3}$$

where **A** represents the weak coupling matrix:

$$\mathbf{A} = \begin{bmatrix} (2^N - \varepsilon_{12} - \varepsilon_{13}) & \varepsilon_{12} & \varepsilon_{13} \\ \varepsilon_{21} & (2^N - \varepsilon_{21} - \varepsilon_{23}) & \varepsilon_{23} \\ \varepsilon_{31} & \varepsilon_{32} & (2^N - \varepsilon_{31} - \varepsilon_{32}) \end{bmatrix}. \tag{4}$$

with ε_{ij} are the weakly coupling parameters, ranging from 1 to 2^k and k≤5.

And $Fp[Xp(n-1)]$, $Fs[Xs(n-1)]$ and $Fl[Xl(n-1)]$ are the discrete functions of the chaotic maps PWLCM, Skew Tent and Logistic respectively defined as follows:

$$Fp[Xp(n-1)] = \begin{cases} \left\lceil 2^N \times \frac{Xp[n-1]}{Pp} \right\rceil & \text{if } 0 < Xp[n-1] \leq Pp \\ \left\lceil 2^N \times \frac{Xp[n-1]-Pp}{2^{N-1}-Pp} \right\rceil & \text{if } Pp < Xp[n-1] \leq 2^{N-1} \\ \left\lceil 2^N \times \frac{2^N-Pp-Xp[n-1]}{2^{N-1}-Pp} \right\rceil & \text{if } 2^{N-1} < Xp[n-1] \leq 2^N - Pp \\ \left\lceil 2^N \times \frac{2^N-Xp[n-1]}{Pp} \right\rceil & \text{if } 2^N - Pp < Xp[n-1] \leq 2^N - 1 \\ 2^N - 1 - Pp & otherwise \end{cases} \tag{5}$$

$$Fs[Xs(n-1)] = \begin{cases} \left\lfloor \frac{2^N \times Xs[n-1]}{Ps} \right\rfloor & \text{if } 0 < Xs[n-1] < Ps \\ 2^N - 1 & \text{if } Xs[n-1] = Ps \\ \left\lfloor \frac{2^N \times (2^N - Xs[n-1])}{2^N - Ps} \right\rfloor & \text{if } Ps < Xs[n-1] < 2^N \end{cases} \tag{6}$$

$$Fl[Xl[n-1]] = \begin{cases} \left\lfloor \frac{Xl[n-1] \times [2^N - Xl[n-1]]}{2^N - 1} \right\rfloor & \text{if } Xl[n-1] \neq [3 \times 2^{N-2}, 2^N] \\ 2^N - 1 & \text{if } Xl[n-1] = [3 \times 2^{N-2}, 2^N] \end{cases} \tag{7}$$

The obtained multiplexed samples of the sequence $X(n)$ are controlled by the chaotic sample $Xth(n)$ and a threshold T, as shown in Fig. 3, and are defined as follows:

$$X(n) = \begin{cases} Xp(n), & \text{if } 0 < Xth(n) < T \\ Xs(n), & \text{otherwise} \end{cases} \tag{8}$$

Where $Xth(n) = Xl(n) \oplus Xs(n)$.

After the generation of all needed samples $X(n)$, the IV-setup function computes a new IV that will be used for the next running of the PCNG. The new IV is generated from the Linux generator: `"/dev/urandom"`.

2.1.2 *Architecture of the second proposed PCNG*

The architecture of the second proposed PCNG is presented in Fig. 4. In comparison with the previous architecture, the main difference lies in the internal-state function, which is based on a binary diffusion matrix **D**.

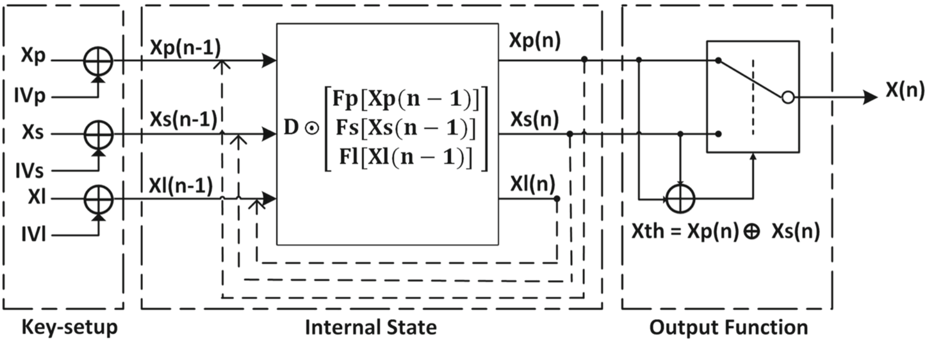

Xp
IVp
Xs
IVs
Xl
IVl
Xp(n-1)
Xs(n-1)
Xl(n-1)
$\mathbf{D} \odot \begin{bmatrix} \mathbf{Fp[Xp(n-1)]} \\ \mathbf{Fs[Xs(n-1)]} \\ \mathbf{Fl[Xl(n-1)]} \end{bmatrix}$
Xp(n)
Xs(n)
Xl(n)
X(n)
Xth = Xp(n) ⊕ Xs(n)
Key-setup
Internal State
Output Function

Fig. 4 Architecture of the second proposed PCNG

The initial values $Xp(0)$, $Xs(0)$ and $Xl(0)$ are initialized throughout the key-setup function, as in Eqs. (1) and (2).

The equation of the system is given by:

$$\begin{bmatrix} Xp(n) \\ Xs(n) \\ Xl(n) \end{bmatrix} = \mathbf{D} \odot \begin{bmatrix} Fp[Xp(n-1)] \\ Fs[Xs(n-1)] \\ Fl[Xl(n-1)] \end{bmatrix}. \tag{9}$$

where D is the binary diffusion matrix:

$$D = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}. \tag{10}$$

And $\odot$ is the operator defined as follows :

$$\begin{bmatrix} Xp(n) \\ Xs(n) \\ Xl(n) \end{bmatrix} = \begin{bmatrix} Fp[Xp(n-1)] \oplus Fs[Xs(n-1)] \\ Fs[Xs(n-1)] \oplus Fl[Xl(n-1)] \\ Fp[Xp(n-1)] \oplus Fl[Xl(n-1)] \end{bmatrix}. \tag{11}$$

The choice of the output samples $X(n)$ is governed, as in Eq. (8) by a threshold T and the chaotic sample Xth, with $Xth(n) = Xp(n) \oplus Xs(n)$.

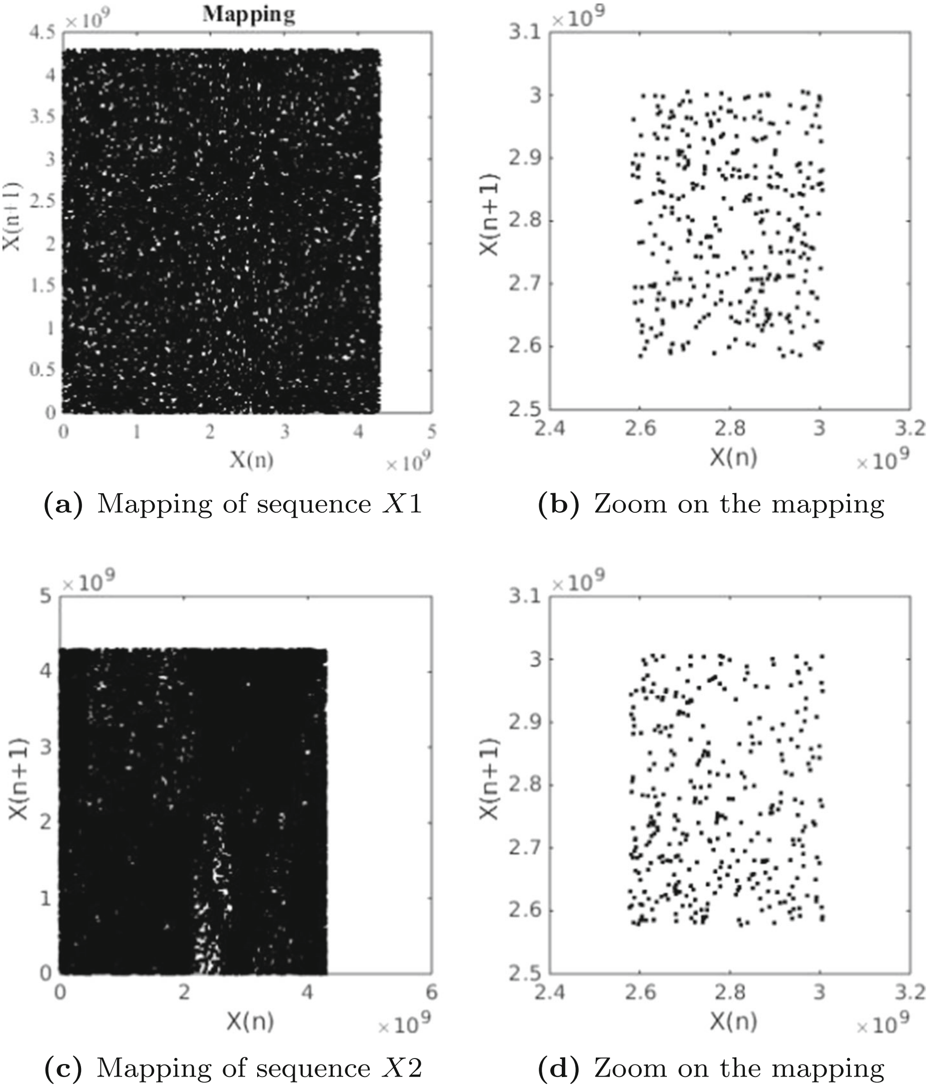
For each new running of the system, the initial vector IV is updated using the Linux generator.

In the following paragraphs, we study two statistical performances, namely mapping and approximated invariant values, and the speed performance of the two PCNGs.

2.1.3 Mapping and approximated invariant values

The mapping or the phase space trajectory is one of the characteristics of the generated sequence that reflects the dynamic behaviour of the system. We draw in Fig. 5a and c the mapping of sequences $X1$ and $X2$, each containing Ns = 31250 samples, generated by the first and second architectures and a zoom of these mappings in Fig. 5b and d.

The resulting mapping of $X1$ and $X2$ seems to be random. This is due to the used techniques of coupling and chaotic multiplexing. In this case, it is impossible from the generated



(a) Mapping of sequence $X1$

(b) Zoom on the mapping

(c) Mapping of sequence $X2$

(d) Zoom on the mapping

Fig. 5 Mapping of sequences $X1$ **(a)** and $X2$ **(c)** of length 31250 samples, generated by the first and the second PCNGs and a zoom of these mapping in **(b)** and **(d)** respectively

sequences to know which type of map is used. However, in the mapping of $X2$, we observe small empty areas. Then, we can say that the generated sequences of the first PCNG are more uniform than those generated by the second PCNG. This observation will be confirmed by the Chi-square test of the ciphered text. Also, we notice that the mapping of the coupled sequences seems to be random.

To prove the value uniformity of the generated sequences, Lozi [19] uses the ”approximated invariant measures”. This function was computed with floating numbers and based on the partition of the mapping space to M^2 small squares (boxes). In finite precision N, we defined the approximated invariant measures $Pd_N(si, tj)$ in the same manner as in [19]. First, the space mapping is divided into M^2 boxes $r_{i,j}$ as follows:

$$s_i = X_{min} + i \times l, i = 0, ..., M. \tag{12}$$

$$t_j = X_{min} + j \times l, j = 0, ..., M. \tag{13}$$

where

$$l = \frac{X_{max} - X_{min}}{M}. \tag{14}$$

with $X_{min} = min(Xi(Ns))$, $X_{max} = max(Xi(Ns))$ and Ns is the number of samples under test.

The box $r_{i,j}$ is given by :

$$r_{i,j} = [s_i, s_{i+1}[\times[t_j, t_{j+1}[, i, j = 0, ..., M - 1. \tag{15}$$

In Fig. 6a and b, we show the $r_{6,6}$ box, after zooming the mapping of sequences $X1$ and $X2$.

The approximated probability distribution function $Pd_N(si, tj)$ is defined as follows:

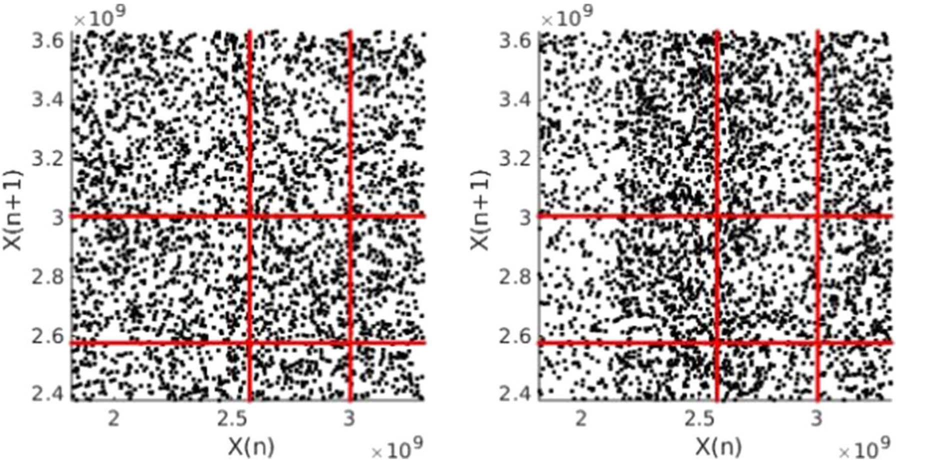
$$Pd_N(si, tj) = \frac{\#r_{i,j}}{Ns/M^2}. \tag{16}$$

with $\#r_{i,j}$ is the number of samples inside the box $r_{i,j}$.

Obtained values of $\#r_{i,j}$ and $Pd_N(si, tj)$ for sequences $X1$ and $X2$ are given in Tables 1 and 2 respectively, for all boxes with $M = 10$ and $Ns = 31250$. In Tables 3 and 4 we give the values of $\#r_{i,j}$ and $Pd_N(si, tj)$ for $Ns = 31250 \times 100$.

Theoretically, the number of samples inside each box $r_{i,j}$ is $Ns/M^2 \simeq 312$. Furthermore, the closer the $Pd_N(si, tj)$ value is to 1, the better the uniformity.

As we can see, compared to results in Table 2, results of Table 1 are closer to uniform distribution. Indeed, for sequence $X1$, the smallest value of $Pd_N(si, tj)$ is 0.832 and we only have 4 values smaller than 0.88. Likewise, the biggest $Pd_N(si, tj)$ value is 1.142 and



(a) Zoom on the phase space of $X1$ **(b)** Zoom on the phase space of $X2$

Fig. 6 Zoom on the phase space of sequences $X1$ and $X2$ generated by the first and second PCNGs

Table 1 Values of $\#r_{i,j}$ and $Pd_N(si, tj)$ for sequence $X1$ with $Ns = 31250$ samples

$\#r_{i,j}$ $Pd_N(si, tj)$									
312	338	291	322	330	289	348	296	323	301
0.998	1.082	0.931	1.03	1.056	0.925	1.114	0.947	1.034	0.963
343	343	311	284	306	314	305	337	316	338
1.098	1.098	0.995	0.909	0.979	1.005	0.976	1.078	1.011	1.082
295	313	293	297	295	312	312	316	321	286
0.944	1.002	0.938	0.95	0.944	0.998	0.998	1.011	1.027	0.915
313	314	318	315	317	307	301	314	287	324
1.002	1.005	1.018	1.008	1.014	0.982	0.963	1.005	0.918	1.037
319	292	314	306	337	285	315	357	319	333
1.021	0.934	1.005	0.979	1.078	0.912	1.008	1.142	1.021	1.066
299	304	301	343	326	313	314	278	320	298
0.957	0.973	0.963	1.098	1.043	1.002	1.005	0.89	1.024	0.954
339	303	313	289	304	310	349	294	318	321
1.085	0.97	1.002	0.925	0.973	0.992	1.117	0.941	1.018	1.027
290	319	313	307	312	328	320	290	324	310
0.928	1.021	1.002	0.982	0.998	1.05	1.024	0.928	1.037	0.992
319	337	273	298	319	346	298	320	260	315
1.021	1.078	0.874	0.954	1.021	1.107	0.954	1.024	0.832	1.008
321	333	313	348	331	292	278	312	297	314
1.027	1.066	1.002	1.114	1.059	0.934	0.89	0.998	0.95	1.005

only we have 5 values bigger than 1.10. For sequence $X2$, we observe that: the smallest value of $Pd_N(si, tj)$ is 0.246 and there are 34 values smaller than 0.88. Additionally, the highest $Pd_N(si, tj)$ value is 1.658 and there are 40 values higher than 1.10.

Besides, compared to results in Table 1, the obtained values of $Pd_N(si, tj)$ in Table 3, are closer to 1. Indeed, the uniformity is better when the number of samples Ns is larger. However, these results are not valid for the PCNG2 when comparing Tables 2 and 4. This is due to the fact that the samples are distributed on a periodic orbit with a small period length.

We give also the cumulative relative error calculated by:

$$CRE = \sum_{i,j=1}^{M} \left| \frac{Ns/M^2 - \#r_{i,j}}{Ns/M^2} \right|. \tag{17}$$

In Table 5, we report the obtained values of CRE for sequences $X1$ and $X2$. For this experiment, we took three different values for Ns : $Ns = 31250$, $Ns = 31250 \times 10$, and $Ns = 31250 \times 100$. And for each Ns, we consider two values of M: $M = 5$ and $M = 10$.

We observe that, whatever the values of Ns and M, the Cumulative Relative Error CRE of sequences generated by PCNG1 is smaller than the CRE of sequences generated by PCNG2. Also, we notice that, for each M, the CRE of PCNG1 decreases with a factor approximately equal to $\sqrt{Ns}$, when Ns increases. However, sequences generated by PCNG2 do not follow the previous rule.

Table 2 Values of $\#r_{i,j}$ and $Pd_N(si, tj)$ for sequence $X2$ with $Ns = 31250$ samples

$\#r_{i,j}$ $Pd_N(si, tj)$									
291	307	284	293	287	313	332	349	328	353
0.931	0.982	0.909	0.938	0.918	1.002	1.062	1.117	1.05	1.13
474	399	415	352	422	205	211	214	184	196
1.517	1.277	1.328	1.126	1.35	0.656	0.675	0.685	0.589	0.627
381	344	359	331	361	292	233	224	250	328
1.219	1.101	1.149	1.059	1.155	0.934	0.746	0.717	0.8	1.05
328	332	339	452	377	277	292	288	236	210
1.050	1.062	1.085	1.446	1.206	0.886	0.934	0.922	0.755	0.672
518	420	421	425	394	169	199	190	214	247
1.658	1.344	1.347	1.36 0	1.261	0.541	0.637	0.608	0.685	0.790
77	141	184	211	208	442	428	477	456	438
0.246	0.451	0.589	0.675	0.666	1.414	1.37	1.526	1.459	1.402
315	226	210	214	257	350	344	351	368	405
1.008	0.723	0.672	0.685	0.822	1.12	1.101	1.123	1.178	1.296
197	298	283	285	276	342	371	387	378	328
0.63	0.954	0.906	0.912	0.883	1.094	1.187	1.238	1.21	1.05
216	253	230	200	232	389	365	402	437	411
0.691	0.81	0.736	0.64	0.742	1.245	1.168	1.286	1.398	1.315
340	352	378	368	383	283	265	263	284	311
1.088	1.126	1.21	1.178	1.226	0.906	0.848	0.842	0.909	0.995

2.1.4 Speed performance of the proposed PCNGs

Speed performance is an important factor for practical applications of the encryption algorithms. We study the computing performance of the proposed PCNGs. The experiment is performed on the flowing materials composed of Intel(R) Core(TM) i5-4300M CPU @2.60GHZz 2.60 GHz with 16.0 GB Running on Ubuntu 14.04 Trusty Linux distribution, using GNU GCC Compiler. The two PCNGs are implemented in C language using sequential and parallel programming.

For the parallel version, we implement the proposed PCNGs with multi-threaded programming using the POSIX threads (`"pthread library"`). Only the internal state and the output functions are implemented in parallel programming. The key-setup function is implemented sequentially. We use a number of cores equal to four. For this, we create four threads $Th_i, i = 0..3$ using the function `"pthread_create"` of the C Language. New sub-initial conditions and parameters (Xp_i, Xs_i, Xs_i, IVp_i, IVs_i and IVl_i) are needed for each thread Th_i. These parameters are calculated as follows:

$$\begin{cases} Xp_i = L_{cir}[Xp_{i-1}, 3] \\ Xs_i = L_{cir}[Xs_{i-1}, 3] \\ Xl_i = L_{cir}[Xl_{i-1}, 3] \\ IVp_i = L_{cir}[IVp_{i-1}, 3] \\ IVs_i = L_{cir}[IVs_{i-1}, 3] \\ IVl_i = L_{cir}[IVl_{i-1}, 3] \end{cases} \tag{18}$$

Table 3 Values of $\#r_{i,j}$ and $Pd_N(si, tj)$ for sequence $X1$ with $Ns = 31250 \times 100$ samples

$\#r_{i,j}$ $Pd_N(si, tj)$									
31300	31602	31335	31421	31404	31468	31206	31201	31286	31462
1.002	1.011	1.003	1.005	1.005	1.007	0.999	0.998	1.001	1.007
31568	31215	30925	31044	31711	31264	31150	31098	31072	31293
1.01	0.999	0.99	0.993	1.015	1	0.997	0.995	0.994	1.001
31425	31076	31270	31229	31304	31279	31287	31068	31073	31020
1.006	0.994	1.001	0.999	1.002	1.001	1.001	0.994	0.994	0.993
31375	30950	31126	30988	31286	31380	31203	30913	31221	31396
1.004	0.99	0.996	0.992	1.001	1.004	0.998	0.989	0.999	1.005
31475	30911	31154	31304	31411	31362	31286	31506	31358	31427
1.007	0.989	0.997	1.002	1.005	1.004	1.001	1.008	1.003	1.006
31395	31360	31380	31601	31251	31458	31173	31380	31096	31440
1.005	1.004	1.004	1.011	1	1.007	0.998	1.004	0.995	1.006
31269	31478	31470	31108	31358	31499	31384	31250	31060	30705
1.001	1.007	1.007	0.995	1.003	1.008	1.004	1	0.994	0.983
31233	31368	31183	31053	31198	31122	31271	31180	30889	31361
0.999	1.004	0.998	0.994	0.998	0.996	1.001	0.998	0.988	1.004
31218	31022	31083	31013	31154	31265	31233	30911	31027	31351
0.999	0.993	0.995	0.992	0.997	1	0.999	0.989	0.993	1.003
31427	31357	31105	31077	31117	31438	31388	31351	31195	31206
1.006	1.003	0.995	0.994	0.996	1.006	1.004	1.003	0.998	0.999

where i = 1..3 and and $L_{cir}[S, q]$ performs the q-bits left circular shift on the binary sequence S.

Figure 7 shows the structure and locations of samples in $Sequence\_X$ of length equal to 10 samples, generated using parallel programming.

In Tables 6 and 7 we give, the average generation time in microsecond (μs), the average bit rate in Megabits/second (Mbits/s) and the average required number of cycles to generate one byte for different lengths of sequences, using sequential and parallel programming respectively. The average is calculated over 100 different sequences using a different secret key for each one.

The bit rate and the number of cycles needed to generate one byte $NCpB$ is defined as follows:

$$Bit\ rate(Mbits/s) = \frac{Generated\ data\ size(Mbits)}{Average\ generation\ time(s)} \tag{19}$$

$$NCpB = \frac{CPUspeed(Hz)}{Bit\ rate(Byte/s)} \tag{20}$$

From results of Tables 6 and 7, we remark first that, due to its less complex internal state, the speed performance of the second PCNG is better than the first one. Second, we observe that, for small size data (up to 32768 bytes) the PCNG implemented with sequential programming is faster than that programmed in parallel (see also Figs. 8 and 9). This is due to the time synchronization between the four threads.

Table 4 Values of $\#r_{i,j}$ and $Pd_N(si,tj)$ for sequence $X2$ with $Ns = 31250 \times 100$ samples

$\#r_{i,j}$ $Pd_N(si,tj)$									
2939	2823	2797	2802	2608	3402	3367	3319	3327	3407
0.94	0.903	0.895	0.897	0.835	1.089	1.077	1.062	1.065	1.09
4531	4195	3813	3800	3876	2230	2323	2216	2031	2059
1.45	1.342	1.22	1.216	1.24	0.714	0.743	0.709	0.65	0.659
3887	3292	3650	3783	3591	2778	2663	2389	2566	2922
1.244	1.053	1.168	1.211	1.149	0.889	0.852	0.764	0.821	0.935
3233	3345	3829	4320	4044	2906	2855	2947	2329	2008
1.035	1.07	1.225	1.382	1.294	0.93	0.914	0.943	0.745	0.643
4950	4343	4116	4080	3985	1571	1806	2010	2286	2323
1.584	1.39	1.317	1.306	1.275	0.503	0.578	0.643	0.732	0.743
900	1425	1851	2093	2094	4643	4681	4707	4572	4391
0.288	0.456	0.592	0.67	0.67	1.486	1.498	1.506	1.463	1.405
2771	2687	2516	2297	2433	3552	3347	3543	3772	4042
0.887	0.86	0.805	0.735	0.779	1.137	1.071	1.134	1.207	1.293
2261	2867	2942	2709	2770	3532	3530	3751	3549	3418
0.724	0.917	0.941	0.867	0.886	1.13	1.13	1.2	1.136	1.094
2041	2535	2427	2167	2329	4037	3713	3823	3920	3974
0.653	0.811	0.777	0.693	0.745	1.292	1.188	1.223	1.254	1.272
3278	3563	3580	3764	3740	2706	2675	2624	2614	2669
1.049	1.14	1.146	1.204	1.197	0.866	0.856	0.84	0.836	0.854

Notice that, apart from the stream cipher, the proposed PCNGs can be used in several applications that require the generation of a large amount of secure random numbers.

In Table 8, we give the performance in terms of $NCpB$ of some known pseudo random number generators: Wang et al., [32], Akhshani et al., [2] and our proposed PCNGs. The comparison is performed for a data size equal to 786432 bytes. It can be observed that the $NCpB$ performance of the proposed PCNGs is better than the others cited.

In the following section, we will study the security analysis and the speed performance of the two proposed stream ciphers.

Table 5 Values of the cumulative relative error

		Ns		
M	PCNG	31250	31250 × 10	31250 × 100
5	1	0.5432	0.1982	0.0651
	2	2.5512	2.3857	2.3657
10	1	4.4869	1.5268	0.4722
	2	23.0597	22.8217	22.6401

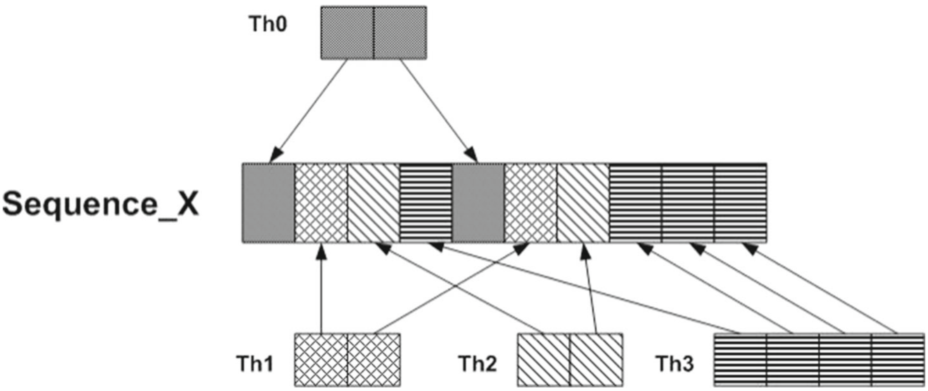


Fig. 7 Location of samples in Sequence_X using parallel programming

3 Security analysis and speed performance of the proposed stream ciphers

A good stream cipher algorithm should be robust against all kinds of cryptanalytic, statistical and brute-force attacks. Also, it should provide a high encryption speed. In this section, we discuss the security analysis of the proposed stream cipher algorithms, based on the first and second proposed PCNGs described in Section 2.1 and their speed performance. Key space, Key sensitivity and Statistical analysis is carried out in order to prove that the proposed stream ciphers are secure against the most common attacks.

As most encryption algorithms (AES-CTR, Rabbit, HC-128...) encrypt 128 bits by 128 bits, our stream cipher algorithms are adjusted also to encrypt 128 by 128 bits of the plain

Table 6 Speed Performance of PCNG1 and PCNG2 using sequential implementation

Data size (Byte)	Generation time (μs)		Bit rate (Mbits/s)		NCpB	
	PCNG1	PCNG2	PCNG1	PCNG2	PCNG1	PCNG2
64	4.02	1.92	127.36	266.66	163.31	74.39
128	5.69	2.58	179.96	369.89	115.58	49.98
256	8.9	3.59	230.11	570.47	90.39	34.77
512	14.89	6.16	275.08	664.93	75.61	29.83
1024	27.36	12.03	299.41	680.96	69.47	29.13
2048	49.63	21.55	330.12	760.27	63.01	26.09
4096	89.33	42.52	366.81	770.51	56.7	25.74
8192	144.19	84.72	454.51	773.52	45.76	25.64
16384	262.31	169.33	499.68	774.03	41.63	25.63
32768	521.49	337.30	502.68	777.17	41.38	25.52
65536	782.22	671.59	670.25	780.65	31.03	25.41
125000	1269.86	1194.58	787.48	837.11	25.19	24.85
196608	1970.40	1867.03	798.24	842.44	24.85	24.69
393216	3930.71	3465.25	800.29	907.79	24.79	22.91
786432	7826.19	6413.96	803.89	980.90	24.68	21.2
3145728	31229.65	25664.74	805.83	980.56	24.63	21.08

Table 7 Speed performance of PCNG1 and PCNG2 using parallel implementation

Data size (Byte)	Generation time (μs)		Bit rate (Mbits/s)		NCpB	
	PCNG1	PCNG2	PCNG1	PCNG2	PCNG1	PCNG2
64	79.56	52.79	6.43	9.69	3082.39	2144.59
128	98.13	65.16	10.43	15.71	1900.93	1323.56
256	95.29	98.96	21.49	35.59	922.96	584.39
512	96.90	79.45	42.27	51.55	492.07	384.77
1024	93.51	76.29	87.60	107.37	237.43	184.73
2048	89.95	84.26	182.14	194.44	108.9	106.97
4096	99.75	77.00	328.50	425.55	60.38	48.88
8192	141,03	127.67	464.67	513.32	44.76	38.64
16384	271,64	222.98	482.52	587.81	43.11	33.75
32768	359.05	293.37	724.48	893.56	27.38	23.28
65536	616.22	491.9	850.81	1065.84	23.31	19.52
125000	1140.89	718.93	876.50	1458.52	22.63	14.26
196608	1548.23	1293.01	1015.91	1621.91	19.53	12.82
393216	2838.58	2235.05	1108.2	1789.66	17.9	11.62
786432	4279.44	3417.52	1470.15	1840.94	13.49	11.3
3145728	16936.79	14827.4	1485.86	1697.25	13.35	12.26

text. For this, the keystream generator produces 128 bits of keystream to be combined with 128 bits of plain text by an XOR operation. Recall that for each new encryption, a new IV is produced.

From the speed performance of the PCNGs given in Tables 6 and 7, the PCNGs are faster when generating 128 bits of keystream in sequential programming. For this, we use sequential programming in the implementation of the two proposed stream ciphers.

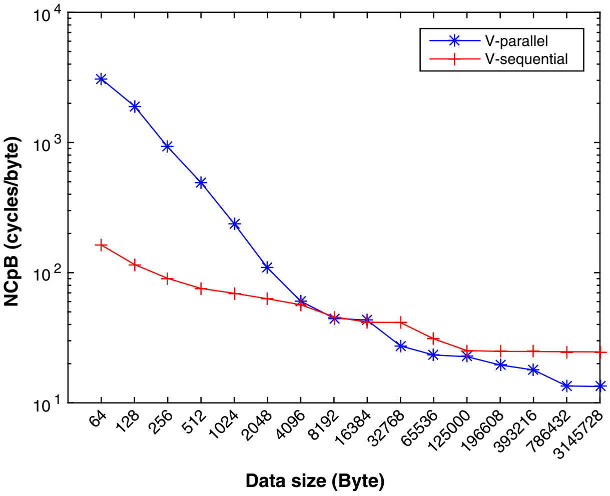


Fig. 8 NCpB of the first proposed PCNG

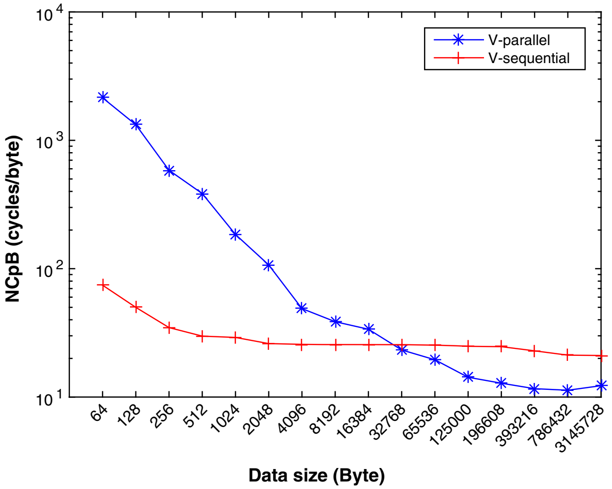


Fig. 9 NCpB of the second proposed PCNG

To evaluate the performance of the proposed stream ciphers, a number of experiments were performed based on several color images, which were used as plain images having the sizes (128 × 128), (256 × 256), (512 × 512) and (1024 × 1024).

3.1 Cryptanalytic analysis

In the following part, some habitual cryptanalytic analysis is performed.

3.1.1 Key space analysis

For any secure crypto-system, the key space should be large enough to resist a brute-force attack. The sizes of the secret keys for the first and second architectures are respectively given by:

$$|K1| = (|Xp| + |Xs| + |Xl|) + (|Pp| + |Ps|) + 6 \times |\varepsilon_{ij}| = 189\, bits \tag{21}$$

$$|K2| = (|Xp| + |Xs| + |Xl|) + (|Pp| + |Ps|) = 159\, bits. \tag{22}$$

where $|Xp| = |Xs| = |Xl| = |Ps| = 32$ bits ; $|Pp| = 31$ bits and $|\varepsilon_{ij}|$ is equal to 5 bits.

Table 8 Computing performance of some known pseudo random number generators

Pseudo random generator	NCpB
Wang et al., [32]	160
Akhshani et al., [2]	45
Abu Taha et al., [13]	17.3
PCNG1	24.68
PCNG2	21.2

Table 9 NPCR and UACI performance

		Baboon	Peppers	Lena
Alg.1	D_H	0.500022	0.500009	0.500015
	$NPCR$	99.60918	99.60866	99.61024
	$UACI$	33.46386	33.46330	33.465842
Alg.2	D_H	0.500017	0.500018	0.500018
	$NPCR$	99.60954	99.60987	99.60899
	$UACI$	33.469	33.46388	33.47010

The proposed algorithms have 2^{189} and 2^{159} different combinations of the secret key. Therefore the secret key sizes of the two architectures are large enough to make brute-force attack infeasible.

3.1.2 Key sensitivity analysis

An efficient stream cipher should be very sensitive to the secret key. The change of a single bit in the secret key should produce a completely different encrypted image. Indeed, to verify this feature, we calculate the average Hamming Distance $D_H(X, Y)$ (using 100 secret keys), between two ciphered images C_1 and C_2, of the same plain image P, with only one change in the least significant bit of the parameter Pp.

$D_H(C_1, C_2)$ is given by the following equation :

$$D_H(C_1, C_2) = \frac{1}{Nb} \times \sum_{K=1}^{Nb} (C_1[K] \oplus C_2[K]) \tag{23}$$

With Nb is the number of bits in an encrypted image.

The obtained results of the Hamming distance for three different ciphered images by the two algorithms are close to the optimal value of 50% (see Table 9). Such results are obtained regardless of the position of the changed bit in the secret key. This demonstrates that the proposed algorithms are highly sensitive to the secret key.

Other common measures used to test sensitivity to the secret key on the encrypted image when changing one bit are the Number of Pixel Change Rate (NPCR) and Unified Average Changing Intensity (UACI). The former is used to measure the number of different pixels between the two images, whereas the latter is used to measure the average intensity difference.

Let $C_1[i, j, p]$ and $C_2[i, j, p]$ be the (i,j,p)th pixel of two ciphered images C_1 and C_2, respectively. The NPCR and UACI are defined by (24) and (26), respectively.

$$NPCR = \frac{1}{L \times C \times P} \times \sum_{p=1}^{P} \sum_{i=1}^{L} \sum_{j=1}^{C} D[i, j, p] \times 100\% \tag{24}$$

$$D[i, j, p] = \begin{cases} 0, & \text{if } C_1[i, j, p] = C_2[i, j, p] \\ 1, & \text{if } C_1[i, j, p] \neq C_2[i, j, p] \end{cases} \tag{25}$$

$$UACI = \frac{1}{L \times C \times P \times 255} \times \sum_{p=1}^{P} \sum_{i=1}^{L} \sum_{j=1}^{C} |C_1 - C_2| \times 100\% \tag{26}$$

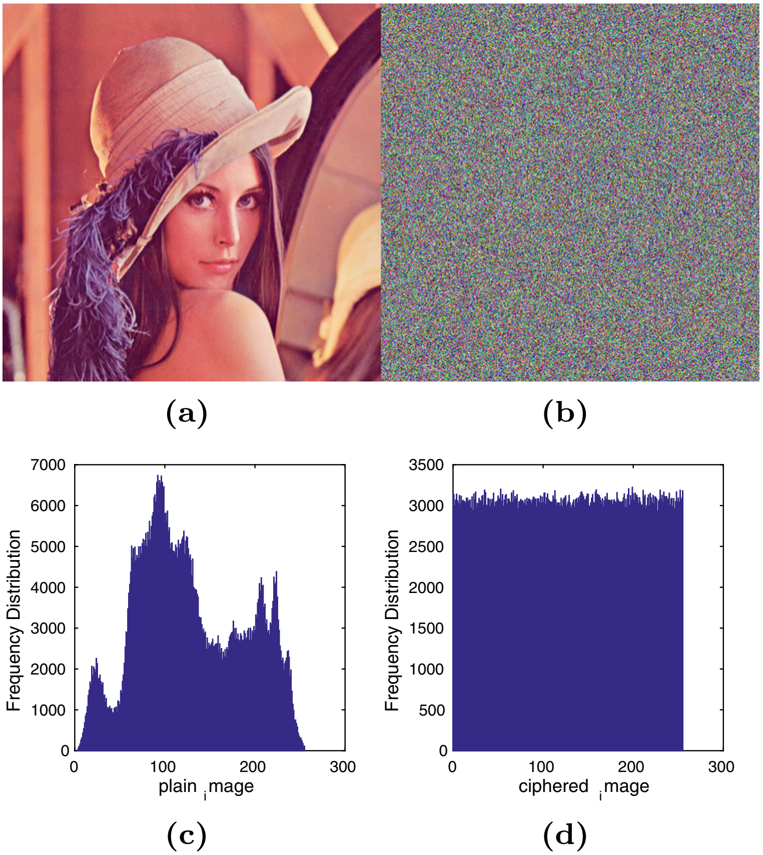


Fig. 10 **a** Lena image, **b** Lena cipher image, **c** the histogram of Lena image and **d** the histogram of the ciphered Lena image

Table 9 also shows the obtained results of NPCR and UACI for the previous three ciphered images. The resulting values are near to the expected values of NPCR and UACI which are 99.60% and 33.46%, respectively [20, 34].

So, as we can see, the proposed algorithms are very sensitive with respect to small changes in the secret Key.

3.2 Statistical analysis

To prove the robustness of the proposed stream ciphers against statistical attacks, we perform the following experiments: histogram, chi-square test, correlation and NIST.

3.2.1 Histogram and Chi-square test analysis

Another key property of a secure stream cipher algorithm is that the encrypted image should have a uniform distribution. We applied the first proposed stream cipher on three different plain images (Lena, Baboon and Peppers) of size (512 × 512 × 3). The obtained results are

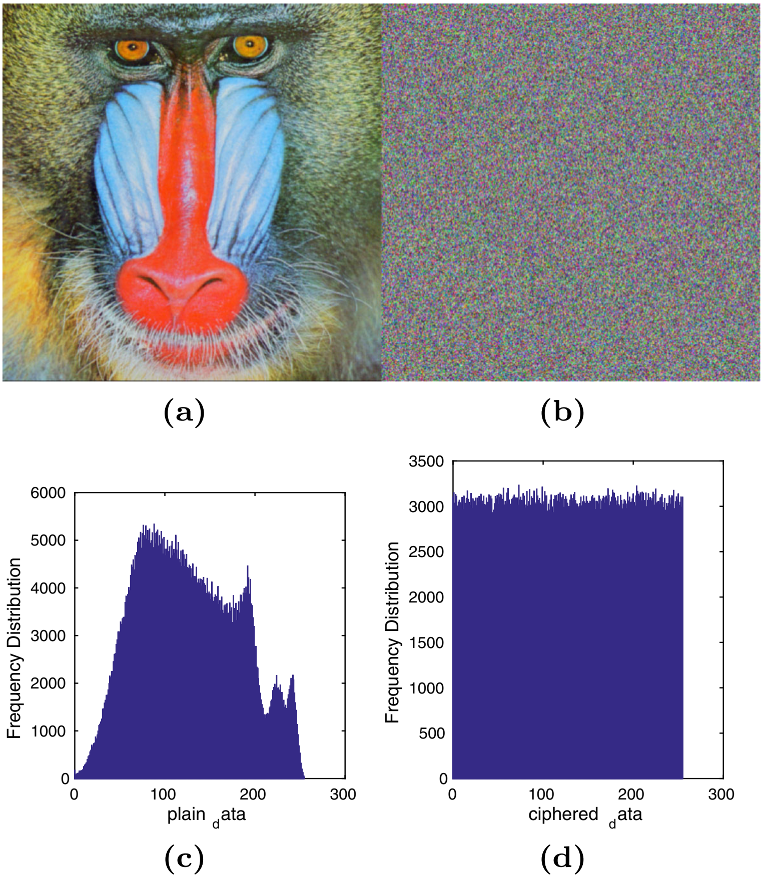


Fig. 11 **a** Baboon image, **b** Baboon cipher image, **c** the histogram of Baboon image and **d** the histogram of the ciphered Baboon image

given in Figs. 10, 11 and 12. On each one, we show (a) the plain image, (b) the corresponding cipher image, (c) the histogram of the plain image and (d) the histogram of the ciphered image.

We can visually observe that the histograms of the encrypted images are uniform and significantly different from those of the plain-images. The same visual results are obtained for the second proposed algorithms.

In order to assert the uniformity of the encrypted images, we apply the Chi-Square test. The experimental Chi-Square test χ^2 is calculated by the following formula:

$$\chi^2_{\text{exp}} = \sum_{i=0}^{K-1} \frac{(O_i - E_i)^2}{E_i}. \tag{27}$$

Where K is the number of levels (here 256), O_i are the observed occurrence frequencies of each color level (0−255) in the histogram of the ciphered image, and E_i is the expected occurrence frequency of the uniform distribution, given here by $E_i = (L \times C \times P)/256$ [9].

We compare the experimental value with the theoretical value obtained for a threshold $\alpha = 0.05$ and a degree of freedom $K-1 = 255$. To prove the uniformity of a sequence,

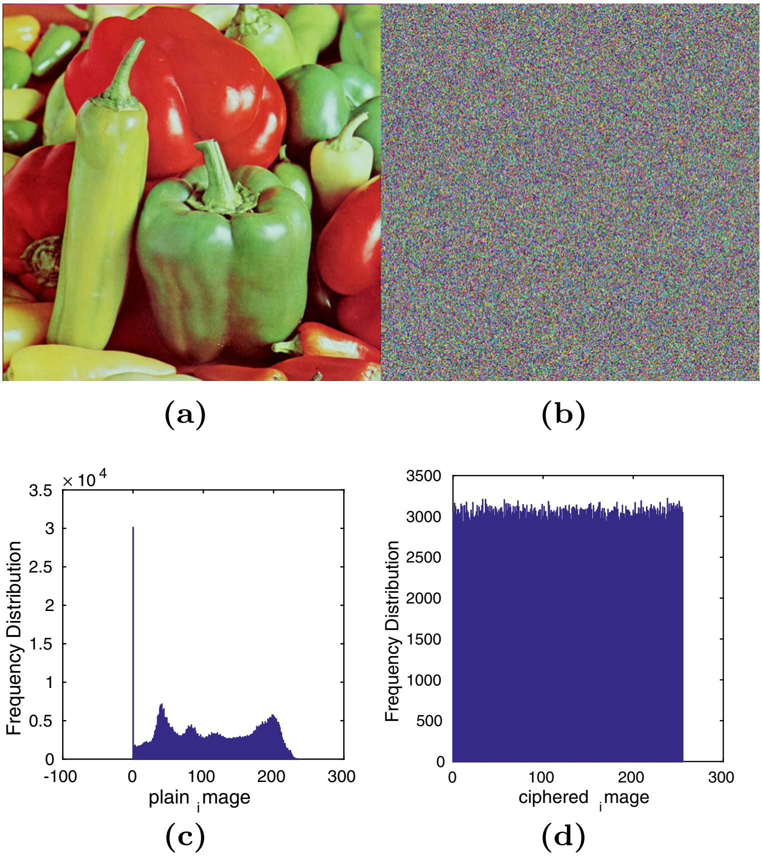


Fig. 12 **a** Peppers image, **b** Peppers cipher image, **c** the histogram of Peppers image and **d** the histogram of the ciphered Peppers image

the experimental value of Chi-Square must be lower than the theoretical one $\chi^2_{\exp} < \chi^2_{th}$ (255, 0.05). The smaller the experimental value of Chi-Square is than the theoretical one, the better the uniformity of the histogram.

In Table 10, we reported the experimental and theoretical values of the Chi-Square test for the three ciphered images (Baboon, Peppers, and Lena) obtained by both proposed algorithms. We note that for the three images, $\chi^2_{\exp} < \chi^2_{th}(255, 0.05)$. Also, images encrypted by the first algorithm have a better uniform distribution those encrypted by the second algorithm.

Table 10 Theoretical and experimental values for the Chi-Square test

	Baboon	Peppers	Lena
χ^2_{th}	293.247	293.247	293.247
$\chi^2_{\exp}$ of Alg.1	211.966	239.10	252.703
$\chi^2_{\exp}$ of Alg.2	267.293	240.68	262.31

Table 11 Correlation coefficients between pairs of plain and encrypted images

Image	Direction	Plain image	Ciphered image by Alg. 1	Ciphered image by Alg.2
Lena	Horizontal	0.993176	0.008713	0.00131
	Vertical	0.997055	0.008154	0.00121
	Diagonal	0.988176	0.008324	0.00117
Baboon	Horizontal	0.99233	0.00157	0.00317
	Vertical	0.99649	−0.00151	−0.00326
	Diagonal	0.98712	−0.00158	−0.00309
Peppers	Horizontal	0.96775	0.00320	0.01183
	Vertical	0.95753	−0.00309	0.00016
	Diagonal	0.93002	−0.00306	0.01480

3.2.2 Correlation analysis

The adjacent pixels in a plain image may have strong correlation. Also, the pixels in an encrypted image with a high security level is expected to be randomly distributed. Therefore, a good encryption scheme should have the ability to efficiently reduce the correlation among adjacent pixels. We measured the correlation coefficient between adjacent pixels, selected randomly from three directions: horizontally, vertically and diagonally. The correlation coefficient ρ_{xy} of adjacent pixels is calculated by the following (28):

$$\rho_{xy} = \frac{\sum_{i=1}^{M}(x_i - \frac{1}{M}\sum_{j=1}^{M} x_j)(y_i - \frac{1}{M}\sum_{j=1}^{M} y_j)}{[\sum_{i=1}^{M}(x_i - \frac{1}{M}\sum_{j=1}^{M} x_j)^2]^{1/2} \times [\sum_{i=1}^{M}(y_i - \frac{1}{M}\sum_{j=1}^{M} y_j)^2]^{1/2}}. \tag{28}$$

where x_i and y_i form i^{th} pair of horizontally/vertically/diagonally adjacent pixels, M is the total number of pairs of horizontally/ vertically/diagonally adjacent pixels.

In Table 11, we give the obtained correlation coefficients in horizontal, vertical and diagonal directions of 1000 pairs of adjacent pixels of the plain images mentioned above and their corresponding ciphered images.

These results show that the correlation coefficients of the plain images are close to 1 while those of encrypted images are near to 0. Then, the proposed encryption schemes generate an image with uncorrelated adjacent pixels. This indicates that the proposed algorithms are secure against statistical attacks.

In addition, such results are confirmed in Fig. 13, which shows the correlation of two horizontally, vertically and diagonally adjacent pixels in the plain and ciphered Baboon image (512 × 512 × 3) using the first algorithm. Similar results are obtained when using the second algorithm.

3.2.3 NIST

To evaluate the performance of the proposed algorithms, we also use one of the most popular standards for investigating the randomness of binary data, namely the NIST statistical test. This test is a statistical package that consists of 15 tests that are proposed to assess the randomness of arbitrarily long binary sequences [10, 27].

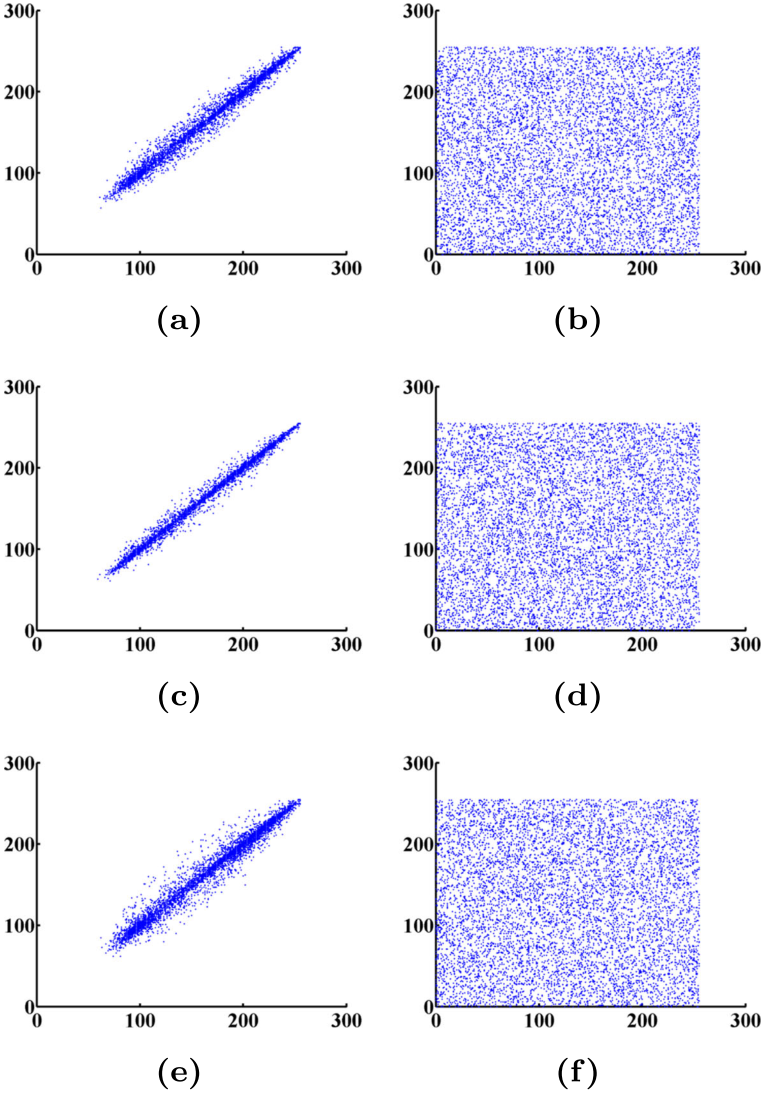


Fig. 13 The distribution of two adjacent pixels in the plain and encrypted images of ’Lena’ **a** and **b** distributions of two horizontally adjacent pixels in the plain and encrypted images of ’Lena’, respectively. **c** and **d** are distributions of two vertically adjacent pixels in the plain and encrypted images of ’Lena’, respectively. **e** and **f** are distributions of two diagonally adjacent pixels in the plain and encrypted images of ’Lena’, respectively

We encrypted 100 different binary sequences of plain text, P_1 and P_2; using the first and second algorithms respectively, each one with a different secret key and containing 10^6 bits. We present the results of NIST for the encrypted sequences C_1 and C_2 in Table 12, with a level of significance of the test $\alpha = 0.01$. Results show that sequences C_1 and C_2 have successfully passed all NIST tests. In addition, we observe that globally, the data ciphered by the first algorithm pass NIST tests more efficiently than data ciphered by the second one. This is in accordance with results previously obtained by the others tests.

Table 12 P-values and Proportion results of NIST for the first and second proposed algorithms

	Alg.1		Alg.2	
Test	*P*-value	Proportion	*P*-value	Proportion
Frequency test	0.946	100	0.740	100
Block-frequency test	0.883	99	0.091	100
Cumulative-sums test	0.376	100	0.646	100
Runs test:	0.616	98	0.658	100
Longest-run test	0.898	100	0.596	99
Rank test	0.290	99	0.534	98
FFT test	0.534	100	0.554	100
Non-periodic-templates	0.483	99.061	0.494	99.088
Overlapping-templates	0.063	100	0.798	100
Universal	0.172	99	0.040	99
Approximty entropie	0.419	99	0.097	98
Random-excursions:	0.335	99.123	0.545	97.656
Random-excursions-variant	0.436	99.318	0.576	99.566
Serial test	0.478	100	0.627	99.5
Linear-complexity	0.249	98	0.262	98

3.3 Speed performance of the proposed stream ciphers

In Table 13 we give the speed performance in terms of average encryption time in (μs), average encryption throughput in (Mbits/s), and the number of cycles needed to encrypt one byte (NCpB) of the proposed algorithms for different sizes of data. We remark that globally, the speed performance of the stream ciphers is approximately 17% less than that of the corresponding PCNGs.

In Table 14, we give a comparison of NCpB for the proposed algorithms (for Lena $512 \times 512 \times 3$) with other chaos-based algorithms and the most known stream ciphers [4, 6].

We observe that the proposed algorithms have a better speed performance than the cited chaos-based algorithms except that of [31]. However, in [31], the authors do not explain the measurement method used to obtain such excellent results, given that the complexity

Table 13 Speed Performance of the two proposed stream ciphers

Data Size (Byte)	Encryption Time (μs)		Encryption throughput (Mbits/s)		NCpB	
	Alg.1	Alg.2	Alg.1	Alg.2	Alg.1	Alg.2
512	21.26	12.50	183.73	312.50	107.96	63.48
1024	37.05	23.35	210.86	334.58	94.07	59.29
2048	44.28	34.70	352.86	450.28	56.21	44.05
4096	73.58	49.05	424.70	637.10	46.71	31.14
256×256×3	2403.04	2168.31	624.20	691.783	31.78	28.71
512×512×3	8511.00	7267.27	704.97	825.6195	28.14	24.03
1024×1024×3	32710.50	28808.11	733.70	833.0987	27.04	23.81

Table 14 Comparison of speed performance between different algorithms

Algorithms	NCpB
Ref. [1]	321
Ref. [17]	226
Ref. [31]	1.77
Proposed Alg.1	28.14
Proposed Alg.2	24.03
AES-CTR	21.2
Rabbit	9.5
HC-128	14.4
Salsa20/12	9.9
SOSEMANUK	10.5

of their system is similar to ours. Compared to the AES-CTR, Rabbit, HC-128, Salsa20/12 and SOSEMANUK, the obtained performance is not as good. However, the non linearity of the proposed systems is higher than the other systems, consequently, its robustness against known attacks is higher.

4 Conclusion

In this paper, we developed two novel chaos-based stream ciphers. The high efficiency obtained from these systems is due to the designed PCNG structure. Indeed, their architectures integrate three chaotic maps weakly coupled using a predefined matrix or coupled by a binary diffusion matrix and using a chaotic multiplexing technique. Simulation tests and security analyses were carried out to prove the efficiency in terms of robustness and speed performance of the proposed stream ciphers. The obtained results show that the proposed stream ciphers can be used in practical applications including secure network communication.

References

1. Ahmed HEdH, Kalash HM, Allah OSF (2007) An efficient chaos-based feedback stream cipher (ecbfsc) for image encryption and decryption. Informatica 31(1)
2. Akhshani A, Akhavan A, Mobaraki A, Lim SC, Hassan Z (2014) Pseudo random number generator based on quantum chaotic map. Commun Nonlinear Sci Numer Simul 19(1):101–111
3. Amato P, Mascolo D, Pedaci I, Ruggiero D (2006) Method of generating successions of pseudo-random bits or numbers. US Patent App 11/381:474
4. Arlicot A (2014) Sequences generator based on chaotic maps. Tech rep IETR, Rennes, France
5. Barkan E, Biham E, Keller N (2003) Instant ciphertext-only cryptanalysis of gsm encrypted communication. In: Advances in cryptology-CRYPTO 2003. Springer, pp 600–616
6. Bougouin M (2015) Time performance analysis of estream stream ciphers. Tech rep, IETR-Polytech Nantes, France
7. Chai X, Yang K, Gan Z (2016) A new chaos-based image encryption algorithm with dynamic key selection mechanisms. Multimedia Tools and Applications 1–21

8. Ekdahl P, Johansson T (2000) Snow-a new stream cipher. In: Proceedings of first open NESSIE workshop, KU-Leuven, pp 167–168
9. El Assad S, Farajallah M (2016) A new chaos-based image encryption system. Signal Process Image Commun 41:144–157
10. Elaine B, John K (2012) Recommendation for random number generation using deterministic random bit generators. Tech rep, NIST SP 800-90 rev a
11. Farajallah M, El Assad S, Deforges O (2016) Fast and secure chaos-based cryptosystem for images. International Journal of Bifurcation and Chaos 26(02):1650,021-1–1650,021-21
12. Guesmi R, Farah MAB, Kachouri A, Samet M (2016) Hash key-based image encryption using crossover operator and chaos. Multimedia Tools and Applications 75(8):4753–4769
13. Jallouli O, Abutaha M, El Assad S, Chetto M, Queudet A, Deforges O (2016) Comparative study of two pseudo chaotic number generators for securing the iot. In: 2016 International conference on advances in computing, communications and informatics (ICACCI). IEEE, pp 1340–1344
14. Klein A (2008) Attacks on the rc4 stream cipher. Des Codes Crypt 48(3):269–286
15. Klein A (2013) Introduction to stream ciphers. In: Stream ciphers. Springer, pp 1–13
16. Kocarev L, Lian S (2011) Chaos-based cryptography. Springer
17. Liu H, Wang X (2010) Color image encryption based on one-time keys and robust chaotic maps. Computers & Mathematics with Applications 59(10):3320–3327
18. Lozi R (2007) New enhanced chaotic number generators. Indian Journal of Industrial and Applied Mathematics 1(1):1–23
19. Lozi R (2012) Emergence of randomness from chaos. International Journal of Bifurcation and Chaos 22(02):1250,021-1–1250,021-15
20. Maleki F, Mohades A, Hashemi SM, Shiri ME (2008) An image encryption system by cellular automata with memory. In: Third international conference on availability, reliability and security, 2008. ARES 08. IEEE, pp 1266–1271
21. Manifavas C, Hatzivasilis G, Fysarakis K, Papaefstathiou Y (2015) A survey of lightweight stream ciphers for embedded systems. Security and Communication Networks
22. Matthews R (1989) On the derivation of a “chaotic” encryption algorithm. Cryptologia 13(1):29–42
23. Pareek NK, Patidar V, Sud KK (2006) Image encryption using chaotic logistic map. Image Vis Comput 24(9):926–934
24. Petersen MV, Sørensen HMB (2007) Method of generating pseudo-random numbers in an electronic device, and a method of encrypting and decrypting electronic data, US Patent 7,170,997
25. Robshaw M (2008) The estream project. In: New stream cipher designs. Springer, pp 1–6
26. Rogaway P, Coppersmith D (1998) A software-optimized encryption algorithm. J Cryptol 11(4):273–287
27. Rukhin AL, Soto J, Nechvatal JR, Smid M, Barker EB, Leigh S, Levenson M, Vangel M, Banks D, Heckert A, Dray J, Vo S (2008) A statistical test suite for random and pseudorandom number generators for cryptographic applications. Tech. rep., NIST SP 800-22 Rev 1
28. Stallings W (2014) Cryptography and network security: principles and practice, international edition: principles and practice. Pearson Higher Ed
29. Tang Y, Wang Z, Ja Fang (2010) Image encryption using chaotic coupled map lattices with time-varying delays. Commun Nonlinear Sci Numer Simul 15(9):2456–2468
30. Taralova I, Lozi R, El Assad S (2012) Chaotic generator synthesis: dynamical and statistical analysis. In: 2012 international conference for internet technology and secured transactions. IEEE, pp 56–59
31. Vidal G, Baptista MS, Mancini H (2014) A fast and light stream cipher for smartphones. The European Physical Journal Special Topics 223(8):1601–1610
32. Wang Y, Liu Z, Ma J, He H (2016) A pseudorandom number generator based on piecewise logistic map. Nonlinear Dyn 83(4):2373–2391
33. Wong KW, Kwok BSH, Law WS (2008) A fast image encryption scheme based on chaotic standard map. Phys Lett A 372(15):2645–2652
34. Wu Y, Noonan JP, Agaian S (2011) Npcr and uaci randomness tests for image encryption. Cyber Journals: multidisciplinary journals in science and technology, Journal of Selected Areas in Telecommunications (JSAT), pp 31–38
35. Zhang W, Wong Kw, Yu H, Zhu Zl (2013) An image encryption scheme using reverse 2-dimensional chaotic map and dependent diffusion. Commun Nonlinear Sci Numer Simul 18(8):2066–2080
36. Zhang X, Zhao Z, Wang J (2014) Chaotic image encryption based on circular substitution box and key stream buffer. Signal Process Image Commun 29(8):902–913

Ons Jallouli is currently a third year PhD student in Computer Systems at IETR Laboratory, University of Nantes (France). She is working under the supervision of Professors Safwan EL ASSAD and Maryline CHETTO. Her thesis title is Chaos-based security under Real-time and Energy constraints for the Internet of Things. She holds a Master Degree in Real-time, Steering and Supervision from Ecole Centrale de Nantes and a computer engineer diploma from National Engineering School of Sfax (Tunisia). Her major research interest is the development of real time cryptosystem for multiple communication. Her research is supported by the European Project Erasmus Mundus-Egovtn.

Safwan El Assad received his PhD degree in electrical engineering from the University of Lille 1, France in 1987. His doctoral thesis was on electromagnetic compatibility. He joints the University of Nantes, France in September 1987, where he is now an Associate Professor. From 1988 to 1996, his main interest was in radar imaging, remote sensing, signal and images processing. From 1996 until 2002, he is interested on digital communications, adaptive equalization for digital channels by neural network, and e-learning. Now his current research are in chaos-based exchanged and stored information security, including: Chaos-based crypto and crypto-compression systems for secure transmitted and stocked data; chaos-based watermarking and steganography. He has graduated 11 Ph.Ds and 20 Master students. He worked on 4 European projects and he published (as an author, co-author) 3 patents, 30 international journals, 6 invited papers 4 bookchapters and 113 articles in international conferences. Applications: Transactions Security: IP over satellite DVB, UMTS, UWB, WIFI security, wireless network security, copyright, data integrity, Mobile security, etc.

Maryline Chetto is a full professor at the University of Nantes. She is conducting her research at IRCCyN (Institut de Recherche en Communications et Cyberntique de Nantes). She received the degree of Docteur de 3ime cycle in control engineering and the degree of Habilit Diriger des Recherches in Computer Science from the University of Nantes, France, in 1984 and 1993, respectively. From 1984 to 1985, she held the position of Assistant professor of Computer Science at the University of Rennes 1, while her research was with the Institut de Recherche en Informatique et Systmes Alatoires, Rennes, France. Her current research interests include scheduling and power management for real-time energy harvesting applications.

René Lozi is professor of highest rank at University of Nice, Dieudonn Center of Mathematics, France. He received his French State Thesis (on chaotic dynamical systems) under the supervision of Prof. Ren Thom (Field medallist) in 1983. In 1991, he became Full Professor at University of Nice and IUFM (Institute for teacher trainees). He has served as Director of this institute (2001-2006) and as Vice-Chairman of the French Board of Directors of IUFM (2004-2006). He is member of several Editorial Boards of International Journals. In 1977 he discovered a particular mapping of the plane having a strange attractor (now classically known as "Lozi map",). Nowadays his research areas include complexity and emergence theory, dynamical systems, bifurcation, control of chaos, cryptography based chaos, and recently memristor (physical device for neuro-computing). He is working in those fields with renowned researchers from many countries. He received the Dr. Zakir Husain Award 2012 of the Indian Society of Industrial and Applied Mathematics during the 12th biannual conference of ISIAM at the University of Punjab, Patialia, January 2015.