State-of-the-art in Chaotic Iterations-based Pseudorandom Numbers Generators Application in Information Hiding

Jacques M. Bahi, Xiaole Fang, and Christophe Guyeux
FEMTO-ST Institute, UMR 6174 CNRS
University of Franche-Comté, Besançon, France
Email:{jacques.bahi, xiaole.fang, christophe.guyeux}@univ-fcomte.fr

***Abstract*—The confidentiality of information transmitted through the Internet requires an intensive use of pseudorandom number generators having strong security properties. For instance, these generators are used to produce encryption keys, to encrypt data with a one-time pad process, or to dissimulate information into cover media. In our previous work, we have proposed the use of discrete chaotic iterations to build pseudorandom number generators that receive two inputted possibly deficient generators, and mix them to produce pseudorandom numbers with high statistical qualities. In this article, we summarize these contributions and we propose simple applications of these generators for encryption and information hiding. For each application, first experimental evaluations are given, showing that an attacker using these statistics as detection tools cannot infer the presence of an hidden message into given cover documents.**

***Keywords*-Internet Security; Pseudorandom Number Generators; Information Hiding; Discrete Chaotic Iterations.**

I. Introduction

Since pseudorandom sequences are easy to be generated and processed, and due to their need in almost all cryptographic protocols and information hiding schemes, PRNGs (Pseudo Random Number Generators) are widely used for a secure Internet use. Among other things, they are part of the keys generation of any asymmetric cryptosystem, they produce keystreams in symmetric cryptosystems, they determine which bits will receive the secret message in information hiding, and so on. However, a lot of existing pseudorandom number generators (PRNGs) used in numerical simulations are eliminated for such applications, due to the requirements of speed, statistical quality, and security in that context.

Recent years, some researchers have investigated with success the use of chaotic dynamical systems to generate pseudorandom sequences [1], [2]. Indeed, chaotic systems have many advantages as unpredictability or disorder-like, which are needed when producing complex sequences. They are extremely sensitive to the initial states too: a minute difference can cause a significant change in output. All these features fit well the requirements of PRNGs, thus explaining the proposal of such dynamics to secure exchanges. However, chaotic systems using real numbers on infinite bit representation, realized in finite computing precision, lead to short cycle length, non-ideal distribution, and other deflation of this kind. This is why chaotic systems on a infinite space of integers have been dig for these years, leading to the proposition to use chaotic iterations (CIs) techniques to reach the desired goals [3].

Having these goals in mind, we have investigated the proposition to mix secure and fast PRNGs, to take benefits from their respective qualities [4], [5]. In [5], CIs have been proven to be a suitable tool for fast computing iterative algorithms on integers satisfying the topological chaotic property, as it has been defined by Devaney [6]. The way that mix two given generators by using these chaotic iterations has been firstly presented in Internet 2009 [3]. It was called "Old CIPRNG". Then, further investigations have been proposed in [7], [5], [8]. These generators were chaotic and able to pass the most stringent batteries of tests, even if the inputted PRNGs were defective. This claim has been verified experimentally, by evaluating the scores of the logistic map, XORshift, and ISAAC generators through these batteries, when considering them alone or after chaotic iterations. Then, in [9], a new version of this family has been expressed. This so-called "New CIPRNG" family uses a decimation of strategies leading to the improvement of both speed and statistical qualities. Finally, most recently, efficient implementations on GPU (Graphics Processing Unit) using a last family named XOR CIPRNG, have been designed in [10], showing that a very large quantity of pseudorandom numbers can be generated per second (about 20 Gsamples/s).

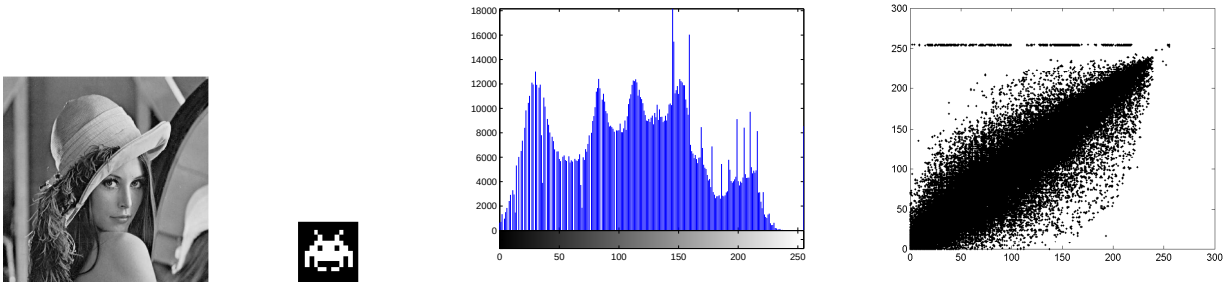
The objective of this article is to make a state-of-the-art of chaotic iterations-based PRNGs, and to propose a possible use of them in the field of secrecy preservation through the Internet, by using information hiding techniques. Random binary sequences will be generated by the three methods mentioned above and a XORshift generator. The application of these pseudorandom bits for information hiding will be carried out systematically, and results will be discussed in order to verify that an attacker, who has only access to some elementary statistical tests, cannot determine whether hidden information are embedded into cover documents or not.

The remainder of this paper is organized in the following way. In Section II, some basic definitions concerning chaotic iterations and XORshift are recalled. Their use to produce three new families of generators is recalled in the next section. Section IV contains the proposed applications of the PRNGs for information hiding. This summary of our previous works in the field of PRNGs and their applications ends by a conclusion section, where our contribution is summarized and intended future work is presented.

II. Basic Remainders

In this section, notations used in this document are introduced, chaotic iterations embedded in the proposed pseudorandom number generators (PRNGs) are defined, and the well-known XORshift generator is recalled.

 ISBN: 978-1-61208-204-2



(a) (b) (c) (d)

Figure 1: (a) The original image. (b) The hidden image. (c) Correlation distribution of the original image. (d) Histogram of the original image.

A. Notations

S^n $\rightarrow$ the n^{th} term of a sequence $S = (S^1, S^2, \ldots)$
v_i $\rightarrow$ the i^{th} component of a vector $v = (v_1, \ldots, v_n)$
f^k $\rightarrow$ k^{th} composition of a function f
$[\![a; b]\!]$ $\rightarrow$ the interval $\{a, a+1, \ldots b\}$ of integers
$X^{\mathbb{N}}$ $\rightarrow$ the set of sequences belonging into X
strategy $\rightarrow$ a sequence of $[\![1; \mathsf{N}]\!]^{\mathbb{N}}$
$\mathbb{S}$ $\rightarrow$ the set of all strategies
$\mathbf{C}_n^k$ $\rightarrow$ the binomial coefficient $\binom{n}{k} = \frac{n!}{k!(n-k)!}$
$\oplus$ $\rightarrow$ bitwise exclusive or
$\ll$ and $\gg$ $\rightarrow$ the usual shift operators

B. Chaotic iterations

Definition 1 The set $\mathbb{B}$ denoting $\{0, 1\}$, let $f : \mathbb{B}^{\mathsf{N}} \longrightarrow \mathbb{B}^{\mathsf{N}}$ be an "iteration" function and $S \in \mathbb{S}$ be a chaotic strategy. Then, the so-called *chaotic iterations* are defined by $x^0 \in \mathbb{B}^{\mathsf{N}}$, and

$$\forall n \in \mathbb{N}^*, \forall i \in [\![1; \mathsf{N}]\!], x_i^n = \begin{cases} x_i^{n-1} & \text{if } S^n \neq i \\ f(x^{n-1})_{S^n} & \text{if } S^n = i. \end{cases}$$

In other words, at the n^{th} iteration, only the S^n-th cell is "iterated".

C. XORshift

XORshift is a category of very fast PRNGs designed by George Marsaglia [11]. It repeatedly uses the transform of exclusive or (XOR) on a number with a bit shifted version of it. The state of a XORshift generator is a vector of bits. At each step, the next state is obtained by applying a given number of XORshift operations to w-bit blocks in the current state, where $w = 32$ or 64. A XORshift operation is defined as follows. Replace the w-bit block by a bitwise XOR of the original block, with a shifted copy of itself by a positions either to the right or to the left, where $0 < a < w$. This Algorithm 1 has a period of $2^{32} - 1 = 4.29 \times 10^9$.

Input: the internal state z (a 32-bit word)
Output: y (a 32-bit word)

1: $z \leftarrow z \oplus (z \ll 13)$;
2: $z \leftarrow z \oplus (z \gg 17)$;
3: $z \leftarrow z \oplus (z \ll 5)$;
4: $y \leftarrow z$;
5: return y;

Algorithm 1: An arbitrary round of XORshift algorithm

III. Chaotic Iterations applied to PRNGs

In this section, we describe the CIPRNG implementation techniques that can improve the statistical properties of a large variety of defective generators. They all are based on chaotic iterations (CIs), which have been defined in the previous section.

A. The Old CIPRNG

Let $\mathsf{N} = 4$. Some chaotic iterations are fulfilled to generate a sequence $(x^n)_{n\in\mathbb{N}} \in \left(\mathbb{B}^4\right)^{\mathbb{N}}$ of Boolean vectors: the successive states of the iterated system. Some of these vectors are randomly extracted and their components constitute our pseudorandom bit flow [3]. Chaotic iterations are realized as follows. Initial state $x^0 \in \mathbb{B}^4$ is a Boolean vector taken as a seed and chaotic strategy $(S^n)_{n\in\mathbb{N}} \in [\![1, 4]\!]^{\mathbb{N}}$ is constructed with $PRNG_2$. Lastly, iterate function f is the vectorial Boolean negation. At each iteration, only the S^n-th component of state x^n is updated. Finally, some x^n are selected by a sequence m^n, provided by a second generator $PRNG_1$, as the pseudorandom bit sequence of our generator.

The basic design procedure of the Old CI generator is summed up in Algorithm 2. The internal state is x, the output array is r. a and b are those computed by $PRNG_1$ and $PRNG_2$.

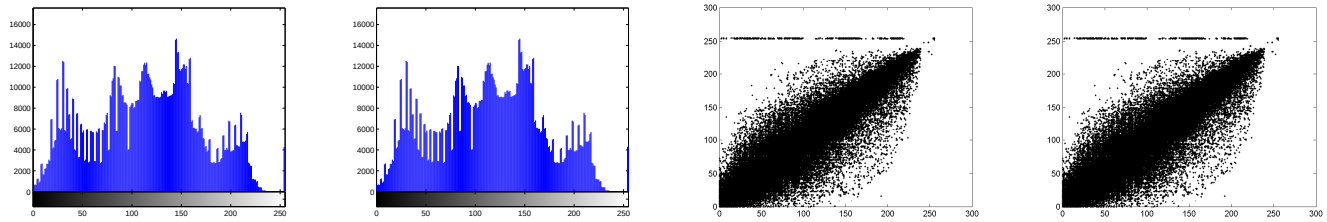
Input: the internal state x (an array of 4-bit words)
Output: an array r of 4-bit words

1: $a \leftarrow PRNG_1()$;
2: $m \leftarrow a \ mod \ 2 + 13$;
3: **while** $i = 0, \ldots, m$ **do**
4: $\quad b \leftarrow PRNG_2()$;
5: $\quad S \leftarrow b \ mod \ 4$;
6: $\quad x_S \leftarrow \overline{x_S}$;
7: **end while**
8: $r \leftarrow x$;
9: return r;

Algorithm 2: An arbitrary round of the Old CI generator

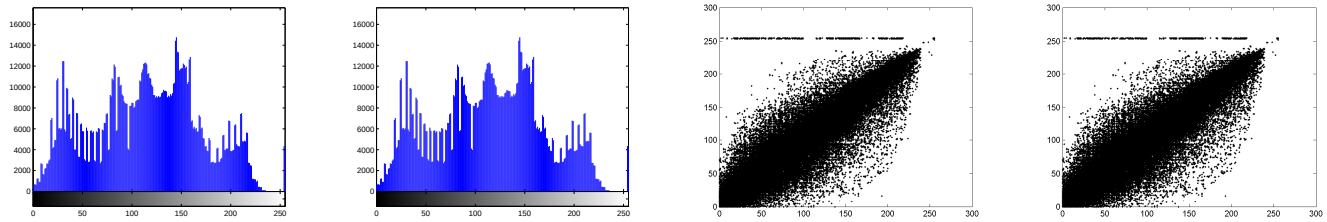
In the paper [3] presented at Internet 2009, the chaotic behavior of CIs is exploited in order to obtain an unpredictable PRNG constituted by two logistic maps. This novel generator has successfully passed the NIST [12]. Then, in [7], we have achieved to improve the speed of the former PRNG, by using two XORshifts in place of the logistic map. In addition,

 ISBN: 978-1-61208-204-2



(a) (b) (c) (d)

Figure 2: (a) Histogram of pixel values when LSBs are replaced by Old CI. (b) Histogram of pixel values when LBSs are an hidden message xored with Old CI. (c) Correlation distribution of two adjacent pixels in Fig.(a). (d) Correlation distribution of two adjacent pixels in Fig.(b).



(a) (b) (c) (d)

Figure 3: (a) Histogram of pixel values when LSBs are replaced by New CI. (b) Histogram of pixel values when LBSs are an hidden message xored with New CI. (c) Correlation distribution of two adjacent pixels in Fig.(a). (d) Correlation distribution of two adjacent pixels in Fig.(b).

this new version of our PRNG is able to pass the famous DieHARD statistical battery of tests [13]. Its security has been improved compared to XORshift alone, and to our former PRNG. However, this latter cannot pass the TestU01 [14] battery, widely considered as the most comprehensive and stringent battery of tests. This goal is achieved by using XORshift and ISAAC as $PRNG_1$ and $PRNG_2$ in [8].

B. New CIPRNG

The New CI generator is designed by the following process [9]. First of all, some chaotic iterations have to be done to generate a sequence $(x^n)_{n\in\mathbb{N}} \in \left(\mathbb{B}^{32}\right)^{\mathbb{N}}$ of Boolean vectors, which are the successive states of the iterated system. Some of these vectors will be randomly extracted and our pseudorandom bit flow will be constituted by their components. Such chaotic iterations are realized as follows. Initial state $x^0 \in \mathbb{B}^{32}$ is a Boolean vector taken as a seed and chaotic strategy $(S^n)_{n\in\mathbb{N}} \in [\![1, 32]\!]^{\mathbb{N}}$ is an *irregular decimation* of $PRNG_2$ sequence, as described in Algorithm 3.

Another time, at each iteration, only the S^n-th component of state x^n is updated, as follows: $x_i^n = x_i^{n-1}$ if $i \neq S^n$, else $x_i^n = \overline{x_i^{n-1}}$. Finally, some x^n are selected by a sequence m^n as the pseudorandom bit sequence of our generator. $(m^n)_{n\in\mathbb{N}} \in \mathcal{M}^{\mathbb{N}}$ is computed from $PRNG_1$, where $\mathcal{M} \subset \mathbb{N}^*$ is a finite nonempty set of integers.

The basic design procedure of the New CI generator is summarized in Algorithm 3. The internal state is x, the output state is r. a and b are those computed by the two input PRNGs. Lastly, the value $g_1(a)$ is an integer defined as in Eq. 1.

$$m^n = g_1(y^n) = \begin{cases} 0 \text{ if } 0 \leqslant y^n < C_{32}^0, \\ 1 \text{ if } C_{32}^0 \leqslant y^n < \sum_{i=0}^{1} C_{32}^i, \\ 2 \text{ if } \sum_{i=0}^{1} C_{32}^i \leqslant y^n < \sum_{i=0}^{2} C_{32}^i, \\ \vdots \qquad \vdots \\ N \text{ if } \sum_{i=0}^{N-1} C_{32}^i \leqslant y^n < 1. \end{cases} \tag{1}$$

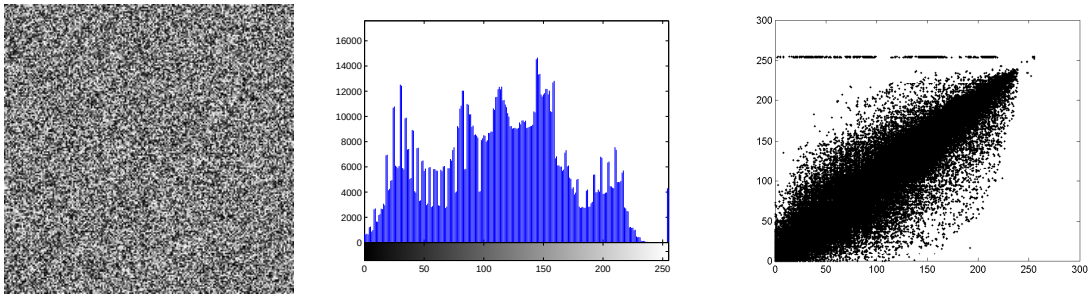
Input: the internal state x (32 bits)
Output: a state r of 32 bits

1: **for** $i = 0, \ldots, N$ **do**
2: $d_i \leftarrow 0$;
3: **end for**
4: $a \leftarrow PRNG_1()$;
5: $m \leftarrow f(a)$;
6: $k \leftarrow m$;
7: **while** $i = 0, \ldots, k$ **do**
8: $b \leftarrow PRNG_2()\ mod\ \mathsf{N}$;
9: $S \leftarrow b$;
10: **if** $d_S = 0$ **then**
11: $x_S \leftarrow \overline{x_S}$;
12: $d_S \leftarrow 1$;
13: **else if** $d_S = 1$ **then**
14: $k \leftarrow k + 1$;
15: **end if**
16: **end while**$r \leftarrow x$;
return r;

Algorithm 3: An arbitrary round of the New CI generator

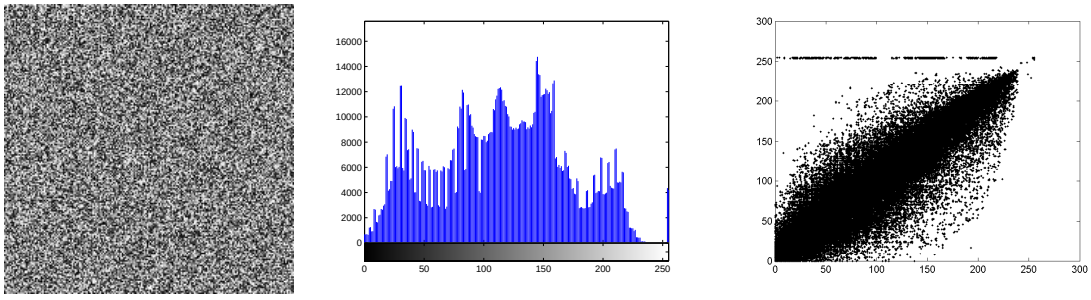
This New CI method presented at Internet 2010 has been

 ISBN: 978-1-61208-204-2



(a) (b) (c)

Figure 4: (a) The encrypted Lena (one-time pad using Old CI). (b) Histogram of Fig.(a). (c) Correlation distribution of two adjacent pixels in Fig.(a)



(a) (b) (c)

Figure 5: (a) The encrypted Lena (one-time pad using New CI). (b) Histogram of Fig.(a). (c) Correlation distribution of two adjacent pixels in Fig.(a)

published in [9]. It was initially using two XORshifts, showing better speed and statistical performance while preserving chaotic properties of the Old CIPRNG. For more information, the reader is referred to [9].

C. XOR CIPRNG

Instead of updating only one cell at each iteration as Old CI and New CI, we can try to choose a subset of components and to update them together. Such an attempt leads to a kind of merger of the two random sequences. When the updating function is the vectorial negation, this algorithm can be simply rewritten as follows [10]:

$$\begin{cases} x^0 \in [\![0, 2^{\mathsf{N}} - 1]\!], S \in [\![0, 2^{\mathsf{N}} - 1]\!]^{\mathbb{N}} \\ \forall n \in \mathbb{N}^*, x^n = x^{n-1} \oplus S^n, \end{cases} \quad (2)$$

The single basic component presented in Eq. 2 is of ordinary use as a good elementary brick in various PRNGs. It corresponds to the discrete dynamical system in chaotic iterations.

IV. Application Evaluation

In this section, the application of PRNGs using CI methods for information hiding is given.

A. The Proposed Information Hiding Method

Suppose that the size of the image is $M \times N$. The steps of the proposed information hiding algorithm using the CIPRNG family are summed up below.

1) Generate a pseudorandom sequence S of length $M \times N$ using the above CI methods respectively.
2) Transform the image into a $M \times N$ integer sequence.
3) The LSBs (Least Significant Bits) of the image integer sequence are replaced by the generated random bits S. These random LSBs will be treated as a keystream.
4) The information (text or picture) to hide is transformed into a binary sequence.
5) The binary message is hiding into the random LSBs of the image sequence, by using the bitwise exclusive or operation between the two sequences, starting from a selected position acting as part of the secret key.

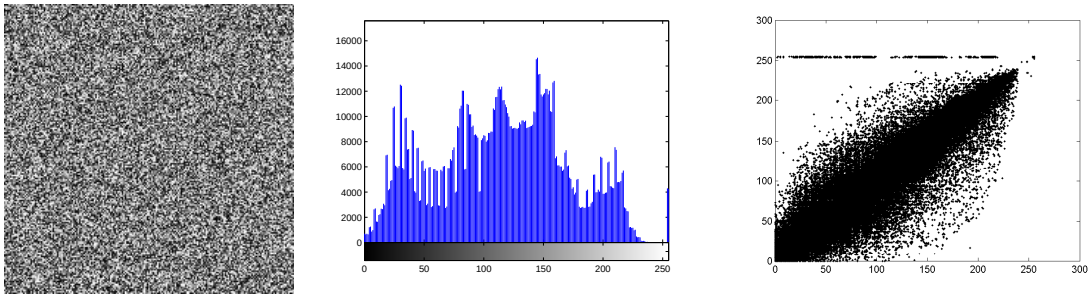
Pseudorandom sequences generated by the three CI methods mentioned in the previous section, with two XORshift generators and a given image, are used in this application to process to an evaluation of the scheme.

B. First Experimental Evaluation of the Proposed Scheme

1) The context: The original image of size 713×713, probably the most widely used test image for all kind of processing algorithms (such as compression and encryption), is depicted in Fig. 1-a. Fig. 1-c presents its histogram, and Fig. 1-d shows the correlation distribution of two horizontal adjacent pixels in this original image. Finally, information that must be hidden into it is the picture of Fig. 1-b, which has 89×89 pixels.

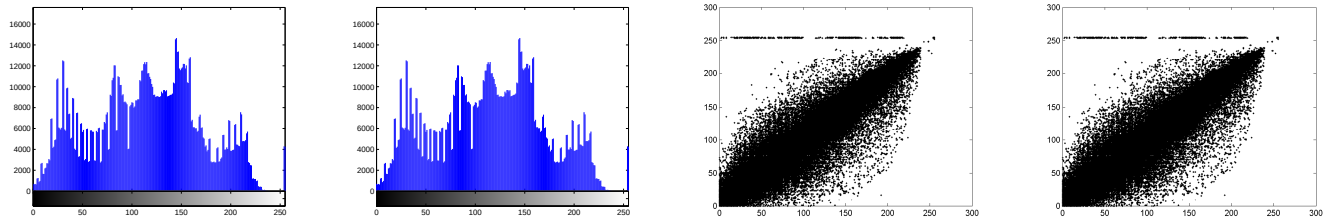
2) Histogram and Horizontal Correlation: Two XORshift generators are used to generate a random sequence based on the old CI method. Results are shown in Fig. 2. Histograms and correlation distributions (Fig. 2-a,b,c,d) are very closed to each other, leading to the assumption that such a method can well protect the hidden information when facing statistical attacks. The same experimental validation has been applied to the New CI method using two XORshift generators. Such

 ISBN: 978-1-61208-204-2



(a) (b) (c)

Figure 6: (a) The encrypted Lena (one-time pad using XOR CI). (b) Histogram of Fig.(a). (c) Correlation distribution of two adjacent pixels in Fig.(a)



(a) (b) (c) (d)

Figure 7: (a) Histogram of pixel values when LSBs are replaced by XOR CI. (b) Histogram of pixel values when LBSs are a hidden message xored with XOR CI. (c) Correlation distribution of two adjacent pixels in Fig.(a). (d) Correlation distribution of two adjacent pixels in Fig.(b).

experiments lead to results that are shown in Fig. 3. These first results are encouraging and confirm that simple histogram and correlation evaluations cannot detect the presence of hidden messages. The same conclusion can be claimed when using the XOR CI generator, as it is depicted in Fig. 7.

3) All directions correlation coefficients analysis: Using an identical experimental evaluation than in [15], the correlation coefficients of the horizontal, vertical, and diagonal directions of all the concerned images (original, with random as LSBs, and with secret information in these LSBs) are shown in Table I. It can be experimentally deduced that the correlation properties of these images are very similar to each other. So an attacker, whose intention is to analyze these coefficients in order to detect possible information hiding, cannot attain his/her goal by such a simple experiment.

4) Initial condition sensitivity: One of the most important properties of the chaotic sequences is that they are very sensitive to their initial conditions. This property can help to face an attacker who has access to the whole algorithm and to an approximation of the secret key. His/her intention, in this attack scenario, is to find the exact secret key (the seed of the keystream and the position of the message), by making small changes on this key. If the keystream and the position do not change a lot when the key is slightly updated, then the attacker can converge by small changes to the used secret key. In the experiments of Figure 8, we slightly alter the keys and try to extract the hidden information from the image. We can conclude that such optimistic attempts always fail in recovering the message.

C. A small evaluation of Encryption

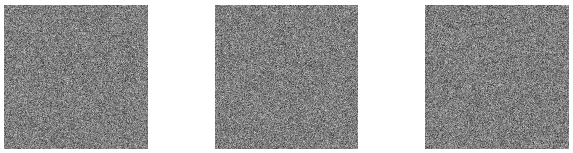
The dissimulation has been obtained in this paper by using the CIPRNGs recalled previously as stream cyphers: encryption is the result of the use of the bitwise exclusive or (XOR) between the given message and pseudorandom sequences generated from various CIPRNGs. We can wonder whether an attacker, who has access to the histogram of LSBs, can infer what can of CIPRNG has been used as keystream. For obvious reasons, these histograms should at least be uniform for each PRNG.

For illustration purpose, Lena has been encrypted by such method using each of the three kind of CIPRNGs, and histogram and correlation distribution of the encrypted image have been computed. The resulting images are depicted in Fig. 4 when using the Old CI method, in Fig. 5 for the New CI one, and in Fig. 6 for the last PRNG recalled here. We can show that this first reasonable requirement seems to be respected, even if this illustration is not a proof.

V. Conclusion

We have summarized in this paper our previous contributions in the field of pseudorandom generators, and we have proposed simple illustrative examples of use for information hiding. The three family of CIPRNGs recalled here are namely the Old CI, the New CI, and the XOR CI PRNGs. For each generator, firsts experimental evaluations of a simple information hiding scheme have been realized, to illustrate that that an attacker using simple statistics cannot determine easily, only by regarding the form of histograms or correlation distributions, the presence of an hidden message into a given document. No evidence of dissimulation appears at first glance, when comparing histograms, correlation distribution, or all

 ISBN: 978-1-61208-204-2



(a) (b) (c)

Figure 8: (a) The difference of two random LSBs image using Old CI PRNG with slight change in initial condition. (b) The difference of two random LSBs image using New CI PRNG with slight change in initial condition (c) The difference of two random LSBs image using XOR CI PRNG with slight change in initial condition

Table I: Correlation coefficients of two adjacent pixels in all directions in the original image, random LSBs images and infomation intergraded random LSBs images

Image \ Direction	Horizontal	Vertical	Diagonal
Original image	0.9793	0.9686	0.9488
Old CI			
no info	0.9792	0.9686	0.9488
intergrading info	0.9792	0.9686	0.9488
New CI			
no info	0.9793	0.9686	0.9488
intergrading info	0.9793	0.9686	0.9488
XOR CI			
no info	0.9793	0.9686	0.9487
intergrading info	0.9793	0.9686	0.9487

directions' correlation coefficients. Furthermore, experiments have illustrated high sensitivity to the secret parameters. These simple evaluations do not imply the security of the proposed scheme, they only illustrate that the use of the recalled PRNGs for information hiding can be further investigated by more stringent tools as steganalyzers and mathematical proofs.

REFERENCES

[1] Y. Hu, X. Liao, K. wo Wong, and Q. Zhou, "A true random number generator based on mouse movement and chaotic cryptography," *Chaos, Solitons & Fractals*, vol. 40, no. 5, pp. 2286–2293, 2009.

[2] L. D. Micco, C. Gonzaez, H. Larrondo, M. Martin, A. Plastino, and O. Rosso, "Randomizing nonlinear maps via symbolic dynamics," *Physica A: Statistical Mechanics and its Applications*, vol. 387, no. 14, pp. 3373–3383, 2008.

[3] Q. Wang, C. Guyeux, and J. M. Bahi, "A novel pseudo-random generator based on discrete chaotic iterations for cryptographic applications," *INTERNET '09*, pp. 71–76, 2009.

[4] J. M. Bahi and C. Guyeux, "A new chaos-based watermarking algorithm," in *SECRYPT 2010, International conference on security and cryptography*, (Athens, Greece), pp. 1–4, July 2010.

[5] J. M. Bahi and C. Guyeux, "Topological chaos and chaotic iterations, application to hash functions," in *WCCI'10, IEEE World Congress on Computational Intelligence*, (Barcelona, Spain), pp. 1–7, July 2010. Best paper award.

[6] R. L. Devaney, *An Introduction to Chaotic Dynamical Systems*. Redwood City: Addison-Wesley, 2nd ed., 1989.

[7] J. Bahi, C. Guyeux, and Q. Wang, "A pseudo random numbers generator based on chaotic iterations. application to watermarking," in *WISM 2010, Int. Conf. on Web Information Systems and Mining*, vol. 6318 of *LNCS*, (Sanya, China), pp. 202–211, Oct. 2010.

[8] J. M. Bahi, C. Guyeux, and Q. Wang, "Improving random number generators by chaotic iterations. application in data hiding," in *ICCASM 2010, Int. Conf. on Computer Application and System Modeling*, (Taiyuan, China), pp. V13–643 – V13–647, Oct. 2010.

[9] Q. Wang, J. Bahi, C. Guyeux, and X. Fang, "Randomness quality of CI chaotic generators. application to internet security," in *INTERNET'2010. The 2nd Int. Conf. on Evolving Internet*, (Valencia, Spain), pp. 125–130, IEEE Computer Society Press, Sept. 2010. Best Paper award.

[10] J. M. Bahi, R. Couturier, C. Guyeux, and P.-C. Héam, "Efficient and cryptographically secure generation of chaotic pseudorandom numbers on gpu," *CoRR*, vol. abs/1112.5239, 2011.

[11] G. Marsaglia, "Xorshift rngs," *Journal of Statistical Software*, vol. 8(14), pp. 1–6, 2003.

[12] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo, "A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications," *NIST Special Publication 800-22*, 2010.

[13] G. Marsaglia, "Diehard: a battery of tests of randomness." http://www.stat.fsu.edu/pub/diehard/ [retrieved: July,2011].

[14] P. L'ecuyer and R. Simard, "Testu01: A software library in ansi c for empirical testing of random number generators," *Laboratoire de simulation et doptimisation. Universit de Montral IRO*, 2009.

[15] G. Chen, Y. Mao, and C. K. Chui, "A symmetric image encryption scheme based on 3d chaotic cat maps," *Chaos, Solitons and Fractals*, vol. 21, no. 3, pp. 749 – 761, 2004.

 ISBN: 978-1-61208-204-2