Available online at www.sciencedirect.com

ScienceDirect

Chaos, Solitons and Fractals 40 (2009) 2557–2568

CHAOS SOLITONS & FRACTALS

www.elsevier.com/locate/chaos

Logistic chaotic maps for binary numbers generations

Ali Kanso, Nejib Smaoui [*]

Department of Mathematics and Computer Science, Kuwait University, P.O. Box 5969, Safat 13060, Kuwait

Accepted 30 October 2007

Abstract

Two pseudorandom binary sequence generators, based on logistic chaotic maps intended for stream cipher applications, are proposed. The first is based on a single one-dimensional logistic map which exhibits random, noise-like properties at given certain parameter values, and the second is based on a combination of two logistic maps. The encryption step proposed in both algorithms consists of a simple bitwise XOR operation of the plaintext binary sequence with the keystream binary sequence to produce the ciphertext binary sequence. A threshold function is applied to convert the floating-point iterates into binary form. Experimental results show that the produced sequences possess high linear complexity and very good statistical properties. The systems are put forward for security evaluation by the cryptographic committees.

1. Introduction

Cryptography addresses a wide range of problems such as confidentiality, data integrity, entity authentication, and data origin authentications. One of the field's central goals remains the classical one of maintaining privacy of communications across a public channel. The techniques of secure communication by which one can exchange private messages secretively over public channels are of great interest in many areas, including systems for electronic commerce, database, secure electronic mail, internet banking, etc. [1]. A cryptosystem is an algorithm that transforms an original message, referred to as plaintext, into a scrambled (non-readable) message, referred to as ciphertext and recovers the message back in its original form. The transformation process from the plaintext to the ciphertext is controlled by a key, and is known as the encryption process, while the transformation process from the ciphertext to the plaintext is also controlled by a key, and is known as the decryption process.

Based on the structure of the algorithm, cryptosystem can be classified into two categories, block ciphers and stream ciphers. Block ciphers tend to simultaneously encrypt groups of characters, whereas stream ciphers operate on individual characters of a plaintext message one at a time. In another classification, which is based on the method of distribution of secret key, one classifies cryptosystem into two classes, the symmetric (private) key and the asymmetric (public) key cryptosystems. In a symmetric key cryptosystem, the key used in the decryption process is the same as (or can be easily obtained from) the key used in the encryption process. Thus, knowledge of the enciphering key is

[*] Corresponding author.
E-mail addresses: akanso@hotmail.com (A. Kanso), nsmaoui64@yahoo.com (N. Smaoui).

0960-0779/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.chaos.2007.10.049

equivalent to knowledge of the deciphering key. In an asymmetric cryptosystem, the sender and receiver hold different (but related) keys. In such a system, the receiver's key and the decryption algorithm determine the reverse of the transformation. For public-key algorithm, the key used in the decryption process is computationally infeasible to compute from the one used for encryption. Thus, public-key algorithms permit the encryption key to be public (it can even be published in a newspaper), allowing anyone to encrypt with the key, whereas only the proper recipient (who knows the decryption key) can decrypt the message. The encryption key is also called the public key and the decryption key is called the private key or the secret key.

A common way to build a stream cipher is to use a pseudorandom sequence (or keystream) generator and mask the plaintext using the output of the keystream generator to produce the ciphertext. A binary additive stream cipher is a synchronous stream cipher in which the keystream, the plaintext and the ciphertext are binary sequences. The plaintext sequence $m_1, m_2, \ldots$ is masked using bit-wise addition modulo 2 (XOR) with the output sequence of the keystream generator, whose initial state constitutes the secret key k, $z_1, z_2, \ldots$ to produce the ciphertext sequence $c_1, c_2, \ldots$. Each secret key k as input to the keystream generator corresponds to an output sequence. Since the secret key k is shared between the transmitter and the receiver, the receiver can decrypt by XORing the output sequence of the keystream generator with the ciphertext sequence, obtaining the plaintext sequence. An efficient stream cipher generator is a generator whose output sequences are in some sense indistinguishable from truly random sequences. A suitable stream cipher generator should be resistant against a known-plaintext attack. In a known-plaintext attack the cryptanalyst has access to a ciphertext and the corresponding plaintext (i.e., part of the keystream sequence), and the challenge is to determine the key k. Apart from the security issues, when implemented in software, we would like our cipher to be very fast and deterministic on different platforms. A number of stream ciphers have been proposed in the literature [1]. Most of them have been bit-oriented stream ciphers based on linear feedback shift registers (LFSRs) [2].

The main difficulty with symmetric key cryptosystems lies in the security of the secret key k that is to be exchanged between the sender and the receiver. However, one can overcome this problem by using a secure channel, or an asymmetric cryptosystem such as RSA [1] to exchange the secret key k.

Most of the existing cryptosystems that have appeared in the literature [1], except few, utilize number theory, combinatorics, algebra, etc. as mathematical tools for constructing the cryptosystems algorithms.

The possibility of using chaotic signals to carry information was first proposed in 1993 by Hayes et al. [3] and, since then, chaotic communications have been given much attention and become an important topic in both nonlinear science and engineering. The high sensitivity of chaotic systems to their initial conditions and parameters together with the uncorrelation, random-like and unpredictability, yet deterministic and easily reproducible, of the chaotic signals can be very helpful in improving the security of transmission in communications. Chaotic communication systems are based either on discrete or continuous systems. In recent years, a growing number of cryptosystems based on continuous systems utilize the idea of synchronization of chaos [4–8]. However, recent studies show that the performance of these systems is very poor and insecure [9–11]. The insecurity results mainly from the insensitivity of synchronization to system parameters [12]. Recently, discrete chaotic communication systems have been given much more attention [13–30]. Most of these discrete chaotic cryptosystem algorithms use one or more chaotic maps as pseudorandom number generators to generate a binary keystream that is used for encryption of a plaintext message to produce a ciphertext. The secret key of such systems constitutes the initial values and/or the system parameters.

In this paper, two binary sequence generators are proposed. One is based on a single one-dimensional logistic map and the other is based on two logistic maps. Both systems are explored to generate pseudorandom binary keystreams for stream cipher applications. The encryption step proposed in both algorithms consists of just a simple bitwise XOR operation of the plaintext binary sequence with the keystream binary sequence to produce the ciphertext binary sequence.

This paper is structured as follows. In Section 2, the properties of the logistic map are discussed. In Section 3, a complete description of the binary sequence generators is presented. The statistical properties of these sequences are discussed in Section 4. Section 5 consists of some experimental results. Some concluding remarks are presented in Section 6.

2. The logistic map

One of the simplest and most studied nonlinear system is the logistic map. It was originally introduced as a demographic model by Pierre Franois Verhulst in 1838. In 1947, Ulam and von Neumann [31] studied the logistic map as pseudorandom number generator. The logistic map is given by:

$$x_{n+1} = \lambda x_n(1 - x_n) \tag{1}$$

where $x_n \in (0,1)$ and λ are the system variable and parameter, respectively, and n is the number of iterations. Thus, given an initial value x_0 and a parameter λ, the series $\{x_n\}_{n=0}^{\infty}$ is computed. In this paper, we refer to x_0 and λ as the initial state of the logistic map.

Fig. 1 shows the bifurcation diagram of this map. This is a plot of the logistic map as a function of λ. For $0 \leqslant \lambda \leqslant 1$, the trivial solution is the only fixed point. For $1 < \lambda \leqslant 3$, we have a non-trivial fixed point. For $3 < \lambda \leqslant 3.57$, the map exhibits the phenomenon of periodic doubling. For $3.57 < \lambda \leqslant 4$, the map becomes chaotic. Finally, for $\lambda = 4$, we observe that chaos values are generated in the complete range of 0–1. Fig. 2 is a blow up of Fig. 1 for the values of λ between 3.5 and 4.

In this paper, the logistic map that we are interested in for the generation of binary keystreams for cryptographic applications is of the form

$$x_{n+1} = \lambda x_n(1 - x_n), \quad \text{for } x_n \in (0,1), \quad \text{and} \quad \lambda \in (3.99996, 4].$$

The choice of λ in the equation above guarantees the existence of a chaotic orbit that can be shadowed by only one map as stated in [32]. Furthermore, the above map is supposed to have good qualities as a pseudorandom number generator when $\lambda \cong 4$ (see [23,33,34]).

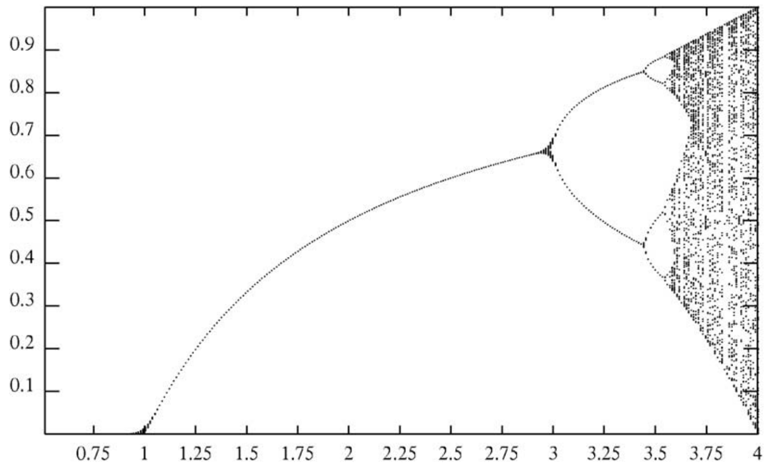

Fig. 1. The bifurcation diagram for the logistic map.

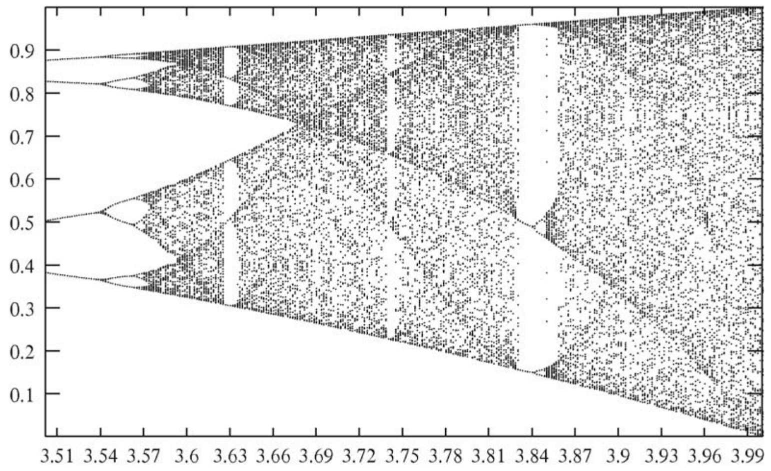

Fig. 2. A blow up of Fig. 1 for the values of $3.5 \leqslant \lambda \leqslant 4$.

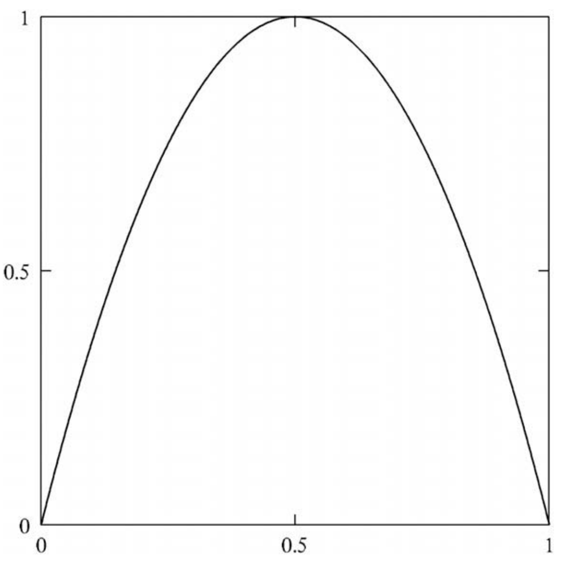

Fig. 3. The attractor of the logistic map.

The iterative values $\{x_n\}_{n=0}^{\infty}$, for the values of λ in the above equation, exhibit sensitive dependence on initial state, and long-term unpredictability. Although these values are limited between bounds, they are pseudorandom and do not converge after any value of iterations. However, these values are highly deterministic as it can be noted from their structure in Fig. 3. This makes the system insecure for cryptographic applications. The most fascinating aspect of the logistic map is its very sensitive dependence upon its initial state. For example, if the initial state is subjected to a disturbance as small as 10^{-30}, iterative values generated after some number of iterations are completely different from each other. As mentioned earlier, this high sensitivity to initial state makes logistic maps and other chaotic maps very important components for cryptographic applications.

3. Proposed pseudorandom bits generators

Generating a pseudorandom binary sequence from the orbit of the logistic map

$$x_{n+1} = \lambda x_n(1 - x_n), \quad \text{for } x_n \in (0, 1), \quad \text{and} \quad \lambda \in (3.99996, 4], \tag{2}$$

essentially requires mapping the state of the system to $\{0, 1\}$. A simple way for turning a real number x_i to a discrete bit symbol z_i is simply by using a threshold function [33]:

$$z_i = F(x_i) = \begin{cases} 0, & \text{if } x_i < c, \\ 1, & \text{otherwise.} \end{cases} \tag{3}$$

Here, c is an appropriately chosen threshold value for x_i. For balanced binary sequence $\{z_i\}_{i=0}^{\infty}$, c should be chosen such that the likelihood of $x_i < c$ is equal to that of $x_i \geqslant c$.

In [35] it is shown that, for $\lambda = 4$, the logistic map is ergodic, which implies that $\forall$ i and almost $\forall x_0$:

1. z_i and z_{i+1} behave as if statistically independent,
2. z_i is equally likely to be 0 or 1.

Since the two symbols 0 and 1 are equally likely to occur for almost all x_0, it is also true that, for any positive integer m, all m-bit strings occur with equal probability. This is the main reason behind our choice for the parameter $\lambda \in (3.99996, 4]$.

For the logistic map (2), 0.5 is approximately the middle of the minimum and maximum of the x_i values, thus, $c = 0.5$ will be a perfect choice [36].

3.1. Proposed pseudorandom bits generator LOGMAP1

The first binary sequence generator proposed in this paper is based on a logistic map (2), and referred to as *LOGMAP*1. The secret key, k, consists of the initial value x_0 and the system parameter λ. The algorithm mixes the binary sequence $\{z_i\}_{i=0}^{\infty}$, generated from (3) by the logistic map (2), using XOR operation with another binary sequence $\{w_i\}_{i=0}^{\infty}$ generated from the same logistic map as follows:

First, we define a function $G : (0,1) \to (0,1)$ such that $G(x_i) = \sum_{h=0}^{i} x_h (\mathrm{mod} 1)$.

Then, we turn the real number $G(x_i)$ to a discrete bit symbol w_i as before using the threshold function:

$$w_i = F(G(x_i)) = \begin{cases} 0, & \text{if } G(x_i) < 0.5, \\ 1, & \text{otherwise.} \end{cases} \tag{4}$$

Finally, the binary sequence $\{\alpha_i\}_{i=0}^{\infty}$ of the algorithm is obtained by mixing the two sequences $\{z_i\}_{i=0}^{\infty}$ and $\{w_i\}_{i=0}^{\infty}$ using XOR operation. That is,

$$\alpha_i = z_i \oplus w_i. \tag{5}$$

For the chosen threshold value $c = 0.5$, the generated sequences pass most of the basic tests for randomness of well-known suites such as Beker and Piper's suite [37] and the FIPS 140-1 suite [38].

Briefly, the algorithm can be expressed as follows:

```
Set $i = 0$.
1. $x_{i+1} = \lambda x_i(1 - x_i)$.
2. $x_i = x_{i+1}$.
3. $y_i = \sum_{h=0}^{i} x_h$ (mod 1).
4. $\alpha_i = F(x_i) \oplus F(y_i)$.
```

3.2. Proposed pseudorandom bits generator LOGMAPS2

A second new algorithm adopting two logistic maps is proposed for the generation of pseudorandom binary sequences. This algorithm consists of two logistic maps:

$$\begin{aligned} x_{n+1} &= \lambda x_n(1 - x_n), \\ x'_{n+1} &= \lambda' x'_n(1 - x'_n), \end{aligned} \tag{6}$$

for $x_n, x'_n \in (0,1)$, and $\lambda, \lambda' \in (3.99996, 4]$.

Define a function SUM that is a mapping from $(0,2) \to (0,1)$ to be:

$$\mathrm{SUM}_i = (x_i + x'_i) \bmod 1, \tag{7}$$

whose attractor is shown in Fig. 4.

The second binary sequence generator proposed in this paper is based on two logistic maps (6), and referred to as *LOGMAP*2. The secret key, k, consists of the initial values x_0, x'_0 and the system parameters λ, λ'. The algorithm mixes the output of two logistic maps (7) to produce a third real number in $(0,1)$.

For the generation of binary sequences we require a mapping of the state of the system to $\{0,1\}$. To do so we apply the above map (3), with $c = 0.5$, for turning the real numbers SUM_i to discrete bit symbols β_i. That is,

$$\beta_i = F(\mathrm{SUM}_i) = \begin{cases} 0, & \text{if } \mathrm{SUM}_i < 0.5, \\ 1, & \text{otherwise.} \end{cases} \tag{8}$$

Let $\{\beta_i\}_{i=0}^{\infty}$ denote the output sequence of *LOGMAP*2.

For the chosen threshold value $c = 0.5$, the generated sequences pass most of the basic tests for randomness of well-known suites such as Beker and Piper's suite [37] and the FIPS 140-1 suite [38].

Briefly, the algorithm can be expressed as follows:

```
Set $i = 0$.
1. $x_{i+1} = \lambda x_i(1 - x_i)$.
2. $x_i = x_{i+1}$.
3. $x'_i = \lambda' x'_i(1 - x'_i)$.
4. $x'_i = x'_{i+1}$.
```

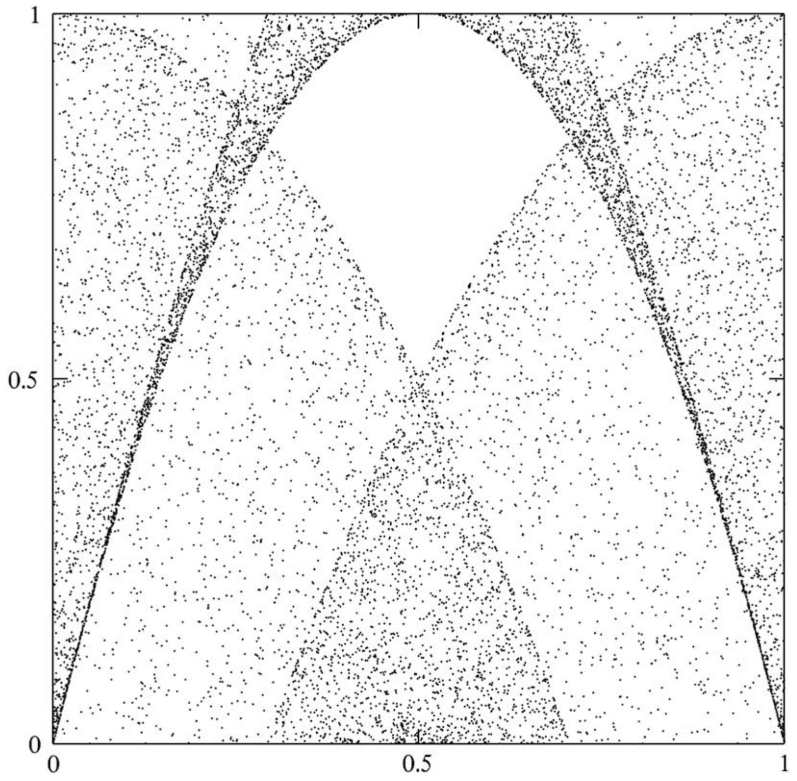

Fig. 4. The combination of two logistic maps represented by the function SUM defined in Eq. (7) destroys the structure of the chaotic attractor observed in Fig. 3.

5. $\mathrm{SUM}_i = (x_i + x_i') \bmod 1$.
6. $\beta_i = F(\mathrm{SUM}_i)$.

The mixing of two or more sequences is not new in cryptography. For example, Maclaren et al. [39] showed that the combination of two congruential generators would produce a more complex pseudorandom sequence than an individual one. Many other combiners based on linear feedback shift registers (LFSRs) such as the shrinking generator, the self-shrinking generator, the alternating step generator, etc. have also been used as alternate sources for the generations of pseudorandom binary sequences [1]. The mixing causes the systems to jump from one unstable trajectory to another one. Both systems remain deterministic as long as the logistic maps are defined.

In the next section, we experimentally demonstrate that sequences produced by the two proposed generators pass all randomness tests of Beker and Piper's [37], and FIPS 140-1 [38] which make them appear as random to a cryptanalyst. Thus, the mixing strengthen the binary sequences $\{\alpha_i\}_{i=0}^{\infty}$ and $\{\beta_i\}_{i=0}^{\infty}$ against various attacks.

Remark 1. It should be noted that the logistic map (2) is non-periodic in nature, but because of finite precision of digital computers the orbits actually turn out to be periodic [40]. In fact, the logistic map enters a periodic loop after 10^7 or more iterations [41].

3.3. Key initialization

In this subsection we discuss some important factors that should be considered before selecting a key for the desired generator *LOGMAP*1 or *LOGMAP*2. Recall that the key of *LOGMAP*1 consists only of x_0 and λ, whereas the key of *LOGMAP*2 consists of x_0, x_0', λ and λ'.

3.3.1. Initial values and system parameters

The initial values x_0 and x_0' should be chosen randomly from the interval $(0, 1)$. For good statistical distribution properties in the generated sequences, the system parameters λ and λ' have to be chosen from the interval $(3.99996, 4]$.

3.3.2. Number of decimal places and keyspace

The number of decimal places to be supported depends purely on the desired security of the system. Suppose that both sender and receiver have calculating machines supporting upon 30 decimal points. In [42] it is shown that a difference of the order of 10^{-30} in the initial value leads to different x_{n+1} values after only 99 iterations. Thus, for this sensitivity order we can have 10^{30} possible initial values between 0 and 1. Therefore, increasing the number of decimal places to be supported results in increasing the keyspace of the desired system and thus increasing its security.

3.3.3. Number of iterations before the encryption/decryption process

For security reasons, and in order to increase the keyspace of the desired algorithm, making use of the high sensitivity of the logistic map (2) upon its initial value and parameter, we iterate that algorithm 200 times without considering its output bits. Thereafter, the encryption process starts by XORing bitwise the plaintext bits with the keystream bits produced by the desired generator to produce the ciphertext bits. The decryption process is completely the same as the encryption process. That is, after the first 200 iterations we start the process by XORing the keystream bits with the ciphertext bits to reproduce the plaintext bits.

Remark 2. The high sensitivity upon initial values and parameters causes the generator to produce two different sequences (after some iterations) for the same initial values and parameters if generated on two generating machines which round off fractions after different decimal places.

In the next section, we consider some statistical properties of the generated sequences. The logistic map is shown in [43] to generate sequences with forbidden strings for some values of λ, such as $\lambda = 3.9, 3.91, 3.93, 3.95, 3.96, 3.97, 3.99$. For example, for $\lambda = 3.99$ the string “0000” never appears in the generated sequences. The proposed generator is shown to overcome this weakness and produce sequences with better statistical properties than the one of the logistic map with these values of λ.

4. Statistical properties

Due to the difficulty of proving the unpredictability in a theoretical way, sequences generated by the proposed generators are subjected to statistical tests. The statistical tests alone cannot verify the unpredictability of the produced sequences which are demanded in cryptographic applications. The predictability of the sequences here is a consequence of the logistic map. Statistical tests determine whether the sequences possess certain attributes that truly random sequences would be likely to exhibit. Hence, any random number generator which is proposed for use in cryptographic applications, must be subject to statistical tests. Beker and Piper [37] described one well-known statistical tests suite which can be applied to provide a quantitative measure of randomness. This suite includes the frequency test, serial test, poker test, runs test, and autocorrelation test. All these tests, in their various ways, measure the relative frequencies of certain patterns of 0’s and 1’s in a section of the sequence. Once we have this measure it is up to us to decide if the sequence is random enough to our purposes. To do this we establish statistical values corresponding to truly random sequences and then set a pass mark. As an illustration, suppose our pass mark is 95%. This means that a given sequence passes the test if its value lies in the range we would expect to find 95% of all truly random sequences.

4.1. Frequency test

In a randomly generated N-bit sequence we would expect approximately half the bits in the sequence to be ones and approximately half to be zeroes. The frequency test checks that the number of ones in the sequence is not significantly different from $N/2$.

4.2. Serial test

The serial test checks that the frequencies of the different transitions in a binary sequence (i.e., 11, 10, 01, and 00) are approximately equal. This will then give us an indication as to whether or not the bits in the sequence are independent of their predecessors.

4.3. Poker test

The poker test is a generalization of the frequency test and the serial test. Where the frequency test studies the number of occurrences of both of the 1-bit patterns, and the serial test studies the number of each of the four 2-bit patterns, the poker test studies the number of occurrences of each of the 2^l l-bit patterns, for some integer l.

4.4. Runs test

The binary sequence is divided into blocks (runs of ones) and gaps (runs of zeroes). The runs test checks that the number of runs of various lengths in our sequence are similar to what we would expect to find in a random sequence. This test is only applied if the sequence has already passed the serial test in which case it is known that the number of blocks and gaps are in acceptable limits.

Table 1
Statistical tests on the sequences $\{\alpha_i\}_{i=0}^{P-1}$ and $\{z_i\}_{i=0}^{P-1}$ with different initial states and with $P = 100{,}000$

Tests		$x_0 = 0.254561$		$x_0 = 0.254563$		$x_0 = 0.05862186$	
		$\lambda = 3.999999301$		$\lambda = 4$		$\lambda = 3.99999861$	
		$\{\alpha_i\}_{i=0}^{P-1}$	$\{z_i\}_{i=0}^{P-1}$	$\{\alpha_i\}_{i=0}^{P-1}$	$\{z_i\}_{i=0}^{P-1}$	$\{\alpha_i\}_{i=0}^{P-1}$	$\{z_i\}_{i=0}^{P-1}$
Frequency		P	P	P	P	P	**F**
Serial		P	P	P	P	P	**F**
Poker	l=3	P	P	P	P	P	P
	l=5	P	P	P	P	P	P
	l=7	P	P	P	**F**	P	P
Runs		P	P	P	P	P	P
Autocorrelation	d=1	P	P	P	P	P	P
	d=2	**F**	**F**	P	P	P	P
	d=3	P	**F**	P	P	P	P
	d=4	P	P	P	P	P	P
	d=5	P	P	P	P	P	P
	d=6	P	P	P	P	P	P
	d=7	P	P	P	P	P	P
	d=8	P	P	P	P	P	P
	d=9	**F**	P	P	P	P	P
	d=10	P	P	P	P	P	P
	d=11	P	P	P	P	P	P
	d=12	P	P	P	P	P	P
	d=13	P	P	P	P	P	P
	d=14	P	**F**	P	P	P	P
	d=15	P	P	P	**F**	P	P
	d=16	P	P	P	P	P	P
	d=17	P	P	P	P	P	P
	d=18	P	P	**F**	P	P	P
Linear Complexity		50001	50000	49999	50001	50001	50001

Table 2
More statistical tests on the sequences $\{\alpha_i\}_{i=0}^{P-1}$ and $\{z_i\}_{i=0}^{P-1}$ with different initial states and with $P = 100{,}000$

Tests		$x_0 = 0.1234854$		$x_0 = 0.786479647$		$x_0 = 0.32151$	
		$\lambda = 3.999995163$		$\lambda = 3.99997743$		$\lambda = 3.99999132$	
		$\{\alpha_i\}_{i=0}^{P-1}$	$\{z_i\}_{i=0}^{P-1}$	$\{\alpha_i\}_{i=0}^{P-1}$	$\{z_i\}_{i=0}^{P-1}$	$\{\alpha_i\}_{i=0}^{P-1}$	$\{z_i\}_{i=0}^{P-1}$
Frequency		P	P	P	P	P	P
Serial		P	P	P	**F**	P	P
Poker	l=3	P	P	P	P	P	**F**
	l=4	P	P	P	**F**	P	P
	l=5	P	P	P	P	P	P
	l=7	P	P	P	**F**	P	P
Runs		P	P	P	**F**	P	P
Autocorrelation	d=1	P	P	P	**F**	P	P
	d=2	P	P	P	P	P	P
	d=3	P	P	P	P	P	P
	d=4	P	P	P	P	P	P
	d=5	P	P	P	P	P	P
	d=6	P	**F**	P	P	P	P
	d=7	P	P	P	P	P	P
	d=8	P	P	P	P	P	P
	d=9	P	P	P	P	P	P
	d=10	P	P	P	P	P	P
	d=11	**F**	P	P	P	P	P
	d=12	P	P	P	P	P	P
	d=13	P	P	P	P	P	P
	d=14	P	P	P	P	P	**F**
	d=15	P	P	P	P	P	P
	d=16	P	P	P	P	P	P
	d=17	P	P	P	P	P	P
	d=18	P	P	P	P	P	P
Linear Complexity		50001	50000	50000	50001	50001	49999

4.5. Autocorrelation test

The idea of this test is to check if there is a correlation between the bits of a given sequence and the bits of a shifted version of the same sequence. Let $\{\theta_i\}_{i=0}^{N-1}$ be a given binary sequence of length N, then if we consider a shift of size d we will compare the bit θ_i with θ_{i+d}. If there is no correlation then we would expect θ_i to be equal to θ_{i+d} half the time.

Beside these randomness tests, a well-know test on stream ciphers is the linear complexity test. Let $\{\theta_i\}_{i=0}^{\infty}$ be a given binary sequence. The linear complexity is the length of the shortest linear feedback shift register [2] that can generate the sequence $\{\theta_i\}_{i=0}^{\infty}$. The importance of this attack is based on the fact that, if a sequence has linear complexity m, then from the knowledge of $2m$ successive bits of that sequence the whole sequence can be regenerated on a LFSR [2]. Thus, a sequence intended for use in cryptographic applications must possess high linear complexity, about half the length of the used sequence. Experimental results have demonstrated that sequences produced by the two proposed generators achieve linear complexity similar to that of a truly random sequence. That is, about half its length ± 1.

Table 3
Statistical tests on the sequence $\{\beta_i\}_{i=0}^{P-1}$ with different initial states and with $P = 100{,}000$

Tests		$x_0 = 0.123, x'_0 = 0.135,$ $\lambda = 3.99999, \lambda' = 4,$ $\{\beta\}_{i=0}^{P-1}$	$x_0 = 0.786, x'_0 = 0.147,$ $\lambda = 3.99997, \lambda' = 3.99998,$ $\{\beta\}_{i=0}^{P-1}$
Frequency		P	P
Serial		P	P
Poker	l=3	P	P
	l=4	P	P
	l=5	P	P
	l=7	P	P
Runs		P	P
Autocorrelation	d=1	P	P
	d=2	P	P
	d=3	P	P
	d=4	P	P
	d=5	P	P
	d=6	P	P
	d=7	P	P
	d=8	P	P
	d=9	P	P
	d=10	P	P
	d=11	P	P
	d=12	P	P
	d=13	P	P
	d=14	P	P
	d=15	P	P
	d=16	P	P
	d=17	P	P
	d=18	P	P
Linear Complexity		49999	50000

The FIPS 140-1 is another well-known statistical suite [38]. It is issued by the National Institute of Standards and Technology (NIST). This suite together with Beker and Piper's suite have been applied to output sequences generated by *LOGMAP*1 and *LOGMAP*2.

In the next section, some experimental results are presented to demonstrate the high linear complexity and good statistical distribution properties of sequences generated by the proposed generators.

5. Experimental results

In this section we list some results of the statistical tests described in [37] which we applied to the first 100,000 bits of each generated sequence $\{\alpha_i\}_{i=0}^{\infty}$ and $\{\beta_i\}_{i=0}^{\infty}$ as described in (5) and (8), respectively. We compare these results with those of the sequence $\{z_i\}_{i=0}^{\infty}$ produced by the logistic map (3). In each of these experiments the initial values x_0, x'_0 and the parameters λ, λ' are chosen at random, where $x_0, x'_0 \in (0,1)$, and $3.99996 < \lambda, \lambda' \leqslant 4$. For all these tests we use $\sigma = 0.05$ as the significance level. Each of these sequences is shown to possess high linear complexity similar to that of a random sequence. The expected value of the linear complexity of a random sequence of length 100,000 is approximately 50,000. If a given sequence passes all the mentioned tests, or all the tests except 5% of the autocorrelation tests then it is to be accepted to use as a keystream for stream ciphers applications, otherwise the sequence is to be rejected.

Several sequences $\{\alpha_i\}_{i=0}^{P-1}$ and $\{\beta_i\}_{i=0}^{P-1}$ of length $P = 100{,}000$ produced by *LOGMAP*1 and *LOGMAP*2, whose initial values $x_0, x'_0 \in (0,1)$ and parameters $\lambda, \lambda' \in (3.99996, 4]$ are randomly chosen, have been put forward for testing using the FIPS 140-1 and Beker, Piper's suites. As a result it turned out that over 95% of these sequences passed the tests of these suites. In comparison with the sequence $\{z_i\}_{i=0}^{P-1}$ it turned out that the last sequence failed a much higher number of tests.

Tables 1–3 include some examples of these experiments. In Tables 1 and 2 we apply some of the statistical tests on the sequences $\{\alpha_i\}_{i=0}^{P-1}$ and $\{z_i\}_{i=0}^{P-1}$. In Table 3 we apply the same tests on the sequence $\{\beta_i\}_{i=0}^{P-1}$. In these tables "**P**" denotes that the sequence has passed the corresponding test and "**F**" denotes that the sequence has failed the corresponding test.

6. Conclusion

In this paper, we have proposed two binary pseudorandom generators based on logistic chaotic maps for cryptographic applications. The proposed generators have strong cryptographic properties such as high sensitivity to initial values and parameters of the logistic maps together with the uncorrelation, random-like and unpredictability. Several statistical tests have been applied on the binary sequences produced by both generators to show that over 95% of these sequences have good statistical distribution properties and high linear complexity. These properties suggest a strong similarity of these sequences to random sequences. Possible cryptanalysis techniques for these sequences will be the subject of future work.

References

[1] Menezes AJ, van Oorschot PC, Vanstone SA. Handbook of applied cryptography. CRC Press; 1997.
[2] Golomb SW. Shift register sequences. Aegean Park Press; 1982.
[3] Hayes S, Grebogi C, Ott E. Communicating with chaos. Phys Rev Lett 1993;70(20):3031–4.
[4] Pecora LM, Carroll TL. Synchronization in chaotic systems. Phys Rev Lett 1990;64:821–4.
[5] Cuomo KM, Oppenheim AV. Circuit implementation of synchronized chaos with applications to communications. Phys Rev Lett 1993;71:65–8.
[6] Kocarev L, Parlitz U. General approach for chaotic synchronization with applications to communication. Phys Rev Lett 1995;74:5028–31.
[7] VanWiggeren GD, Roy R. Communicating with chaotic lasers. Science 1998;279:1198–200.
[8] Boccaletti S, Kurths J, Osipov G, Valladares DL, Zhou CS. The synchronization of chaotic systems. Phys Rep 2002;366(1–2):1–101.
[9] Perez G, Cerdeira H. Extracting messages masked by chaos. Phys Rev Lett 1995;74:1970–3.
[10] Short KM, Parker AT. Unmasking a hyperchaotic communication scheme. Phys Rev E 1998;58:1159–62.
[11] Zhou C, Lai CH. Extracting messages masked by chaotic signals of time-delay systems. Phys Rev E 1999;60(1):320–3.
[12] Wang S, Kuang J, Li J, Luo Y, Lu H, Hu G. Chaos-based secure communications in a large community. Phys Rev E 2002;66:065202.

[13] Chen G, Mao Y, Chui C. A symmetric image encryption scheme based on 3D chaotic cat maps. Chaos, Solitons & Fractals 2004;21(3):749–61.
[14] Lu H, Wang S, Li X, Tang G, Kuang J, Ye W, et al. A new spatiotemporally chaotic cryptosystem and its security and performance analyses. Chaos 2004;14(3):617–29.
[15] Machado RF, Baptista MS, Grebogi C. Cryptography with chaos at the physical level. Chaos, Solitons & Fractals 2004;21(5):1265–9.
[16] Tang G, Liao X, Chen Y. A novel method for designing S-boxes based on chaotic maps. Chaos, Solitons & Fractals 2005;23(2):413–9.
[17] Xiao D, Liao X, Wong K. An efficient entire chaos-based scheme for deniable authentication. Chaos, Solitons & Fractals 2005;23(4):1327–31.
[18] Huang F, Guan ZH. Cryptosystem using chaotic keys. Chaos, Solitons & Fractals 2005;23(3):851–5.
[19] Huang F, Guan ZH. A modified method of a class of recently presented cryptosystems. Chaos, Solitons & Fractals 2005;23(5):1893–9.
[20] Gao H, Zhang Y, Liang S, Li D. A new chaotic algorithm for image encryption. Chaos, Solitons & Fractals 2006;29(2):393–9.
[21] Ali-Pacha A, Hadj-Said N, M'Hamed A, Belgoraf A. Lorenz's attractor applied to the stream cipher (Ali-Pacha generator). Chaos, Solitons & Fractals 2007;33(5):1762–6.
[22] Kotulski Z, Szczepanski J. Discrete chaotic cryptography. Annalen der Physik 1997;6(5):381–94.
[23] Baptista MS. Cryptography with chaos. Phys Lett A 1998;240:50–4.
[24] Alvarez E, Fernandez A, Garca P, Jimenez J, Marcano A. New approach to chaotic encryption. Phys Lett A 1999;263:373–5.
[25] Wong KW, Ho SW, Yung CK. A chaotic cryptography scheme for generating short ciphertext. Phys Lett A 2003;310:67–73.
[26] Lee P, Pei S, Chen Y. Generating chaotic stream ciphers using chaotic systems. Chin J Phys 2003;41:559–81.
[27] Pareek NK, Patidar V, Sud KK. Discrete chaotic cryptography using external key. Phys Lett A 2003;309:75–82.
[28] Pareek NK, Patidar V, Sud KK. Cryptography using multiple one-dimensional chaotic maps. Commun Nonlinear Sci Numer Simulat 2005;10(7):715–23.
[29] Xiang T, Liao X, Tang G, Chen Y, Wong K. A novel block cryptosystem based on iterating a chaotic map. Phys Lett A 2006;349(1–4):109–15.
[30] Li P, Li Z, Halang Wo, Chen G. A stream cipher based on a spatiotemporal chaotic system. Chaos, Solitons & Fractals 2007;32(5):867–1876.
[31] Ulam SM, von Neumann J. On combination of stochastic and deterministic processes. Bull Am Math Soc 1947;53:1120.
[32] Smaoui N, Kostelich E. Using chaos to shadow the quadratic map for all time. Int J Comput Math 1998;70:117–29.
[33] Li S, Mou X, Cai Y. Pseudo-random bit generator based on couple chaotic systems and its applications in stream-cipher cryptography. In: Proceedings of INDOCRYPT 2001. Lecture notes in computer science, vol. 2247. Springer-Verlag. p. 316–29.
[34] Kocarev L, Jakimoski G. Logistic map as a block encryption algorithm. Phys Lett A 2001;289(4–5):199–206.
[35] Schuster HG. Deterministic chaos. Weinheim: Physik Verlag; 1984.
[36] Kotulski Z, Szcepanski J, Gorski K, Gorska A, Paszkiewicz A. On constructive approach to chaotic pseudorandom number generators. In: Proceedings of RCMCIS 2000, Zegrze. p. 191–203.
[37] Beker H, Piper F. Cipher systems: the protection of communications. New York: van Nostrand Reinhold; 1982.
[38] NIST. Federal Information Processing Standards Publication (FIPS140-1). Security requirements for cryptographic modules; 1994.
[39] Maclaren MD, Marsaglia G. Uniform random number generators. J Assoc Comput Mach 1965;12(17):83–9.
[40] Ott E, Grebogi C, Yorke JA. Roundoff-induced periodicity and the correlation dimension of chaotic attractors. Phys Rev A 1988;38:3688–92.
[41] Phatak SC, Rao SS. Logistic map: a possible random number generator. Phys Rev E 1995;51:3670.
[42] Bose R, Banerjee A. Implementing symmetric key cryptography using chaos functions. In: Seventh international conference on advanced communications and computing (ADCOM), University of Roorkee, Roorkee, India; 1999. p. 318–21.
[43] Deane JH, Jefferies DJ. Chaotic dynamics and forbidden words. In: Complex 2000 conference, Dunedin, New Zealand, November 2000. www.ee.surrey.ac.uk/Personal/D.Jefferies/ps/forbidden.ps.