International Journal of Modern Physics C
Vol. 21, No. 3 (2010) 365–382

DOI: 10.1142/S0129183110015166

World Scientific
www.worldscientific.com

FAST, PARALLEL AND SECURE CRYPTOGRAPHY ALGORITHM USING LORENZ'S ATTRACTOR

ANDERSON GONÇALVES MARCO
Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Av. do Trabalhador Sãocarlense 400
13560-970 São Carlos, São Paulo, Brazil
lock@grad.icmc.usp.br

ALEXANDRE SOUTO MARTINEZ
Universidade de São Paulo
Faculdade de Filosofia Ciências e Letras de Ribeirão Preto
Av. Bandeirantes 3900
14040-901 Ribeirão Preto, SP, Brazil
National Institute of Science and Technology in Complex Systems
City, Brazil
asmartinez@usp.br

ODEMIR MARTINEZ BRUNO
Universidade de São Paulo
Instituto de Física de São Carlos
Av. do Trabalhador Sãocarlense 400
13560-970 São Carlos, São Paulo, Brazil
bruno@ifsc.usp.br

Received 17 November 2009
Accepted 26 January 2010

A novel cryptography method based on the Lorenz's attractor chaotic system is presented. The proposed algorithm is secure and fast, making it practical for general use. We introduce the chaotic operation mode, which provides an interaction among the password, message and a chaotic system. It ensures that the algorithm yields a secure codification, even if the nature of the chaotic system is known. The algorithm has been implemented in two versions: one sequential and slow and the other, parallel and fast. Our algorithm assures the integrity of the ciphertext (we know if it has been altered, which is not assured by traditional algorithms) and consequently its authenticity. Numerical experiments are presented, discussed and show the behavior of the method in terms of security and performance. The fast version of the algorithm has a performance comparable to AES, a popular cryptography program used commercially nowadays, but it is more secure, which makes it immediately suitable for general purpose cryptography

applications. An internet page has been set up, which enables the readers to test the algorithm and also to try to break into the cipher.

Keywords: Encryption/decription; chaos; Lorenz's attractor; nonlinear systems.

PACS Nos.: 05.45.-a, 05.45.Vx, 05.45.Gg, 05.45.Ra.

1. Introduction

Since ancient times, the art of encrypting messages using ciphers or breaking into them, has decided the destiny of the civilization. Throughout the history, in many situations, cryptography was directly responsible for the fortune or doom of kings and queens. The success of exchanging secret messages decided the future of many battles and even wars. Some of them were important and very famous, such as the Thermopiles battle (Greece — 490 BC), where according to the legend, the Spartans were notified of the Persian invasion by a crypto message hidden in earthenware, which enabled the Spartans to plan a defense strategy before the Persian attack. Although the secret of the information plays an essential role in military strategy, it was only during the 20th century wars that its real importance become obvious and popular. With the advent of radio, it was possible to exchange vital strategy information by electromagnetic waves. Since there are no boundaries for radio waves, the cryptography power appeared as a fundamental strategy element. It is responsible for ensuring that the enemy, although being able to receive the message, is not capable of understanding it. In fact, cryptography was decisive in the destiny of the first and second world war. Although cryptography is an important military and political information tool, these uses overcame the war times. In the digital ages, cryptography ensures security of business and bank transactions over the internet, and it can also ensure privacy of private messages, making the modern computer, the internet and wireless networks secure and usable technologies.

The objective of cryptography consists of encrypting a message, file, document or image. To understand this procedure, consider the following definitions. One thinks of the object to be encrypted as a string (vector) of size n_p. The components of this vector is transformed to decimal ASCII representation (integers ranging from 0 to $2^8 - 1$) forming the *plaintext*, which is represented by $\mathbf{P}$, with size n_p. Also, one needs a *password* string, which is transformed to decimal ASCII representation forming $\boldsymbol{\pi}$, with size n_π. The plaintext $\mathbf{P}$ is transformed into the ciphertext $\mathbf{C}$ via an *encode function* $\mathbf{C} = E_{\boldsymbol{\pi}}(\mathbf{P})$, which is parametrized by the password $\boldsymbol{\pi}$. To retrieve the plaintext from the cipher, one uses the *decode function* $\mathbf{P} = D_{\boldsymbol{\pi}}(\mathbf{C})$, where $D_{\boldsymbol{\pi}} = E_{\boldsymbol{\pi}}^{-1}$ is the inverse of the encode function and parametrized by $\boldsymbol{\pi}$. Perhaps the simplest encode function is the monoalphabetic (Cesar) cipher $E_\pi(P_i) = (P_i + \pi) \bmod 2^8$, where 2^8 is the length of our alphabet, P_i is a component of $\mathbf{P}$ and π is the integer associated to the password. The associated decode function is: $D_\pi(C_i) = (C_i - \pi) \bmod 2^8$. Here, we call attention to the reversibility property of the mod operation. Care must be paid because the signal in one of the

arguments of the mod operation is reversed, for instance, $C = A + B \bmod D$ and $A = C - B \bmod D$.

The current cryptography methods are based on discrete mathematics (symmetrical key)[1–4] and number theory (asymmetrical key).[5–7] These algorithms use elementary mathematical as ciphers and have to increase the computing complexity to ensure a secure crypto. Taking this into account, they carry out many bit-to-bit operations and permutation between neighboring elements, or they have to use very long keys (asymmetrical algorithms — nowadays RSA algorithms, for instance, use keys varying between 2^{10} and 2^{11} bits) to hide information. However, for both strategies, the increase in the computing complexity and the size of the keys only ensure that the cipher is secure for a certain time. The constant increase in computer power and the evolution of the cryptoanalyses algorithm have made it easy to break into. Nowadays, although there are algorithms which are capable to breaking into this kind of ciphers, they take too long, making them impractical.[8–10] The increase in computational power makes algorithms based on elementary mathematics weak. One example is the DES method,[9] which was popular and largely used in the past. However, the algorithm became too weak and is no longer in use nowadays.

To make the ciphers more secure, some new mathematical methodologies have to be considered and consequently new cryptography approaches arise. In this context, the chaos theory and the complex dynamical systems appear as very attractive alternatives to develop cryptography methods.[11] Chaotic systems have some characteristics that make them valuable for cryptography, such as:

(i) complex numerical patterns,
(ii) unpredictably for unknown initial conditions,
(iii) strong dependence on the initial conditions,
(iv) based on relatively simple equations, and
(v) determinism.

The cryptography algorithms based on chaos generally explore the symmetrical approach.

Cryptography algorithms using chaos appeared in the 90's. The first ones shuffle continue time signals using chaos[12–15] and also digital signals.[16] At the end of the decade, chaos cryptography applied to digital data was introduced. In 1998, Baptista[17] proposed the use of the logistic map as a pseudorandom generator to make a cipher capable of dealing with bit sequences. The Baptista's algorithm was improved by Wong and collaborators[18] making it faster and more secure. The block cryptograph was incorporated in the logistic map in 2005 by Xiang *et. al.*[19] as an enhancement of the previous methods. As far as we are aware, there are two proposals in the literature using the Lorenz's attractor as basis of cryptography methods. The first one[20] was proposed in 1998, but it is very simple and weak. In the second one,[21] the Lorenz's attractor is used to generate pseudorandom numbers in the same sense as the logistic map based ciphers.

In this work a novel cryptography method based on the Lorenz's attractor is introduced. The proposals of the algorithm are to be secure and fast, making it practical for general use. In cryptography theory, the strongest cipher (impossible to break) is reached when the pad has the same length as the message (OTP — One-time pad cipher).[22] We use the Lorenz's attractor to compute a virtually infinite pad exploring the chaos. The complexity of the attractor and its determinism, makes it possible to compute chaotic and long number sequences. What makes the OTP impossible to be broken in is the fact that one does not know the pad. For instance, if one uses a book to write a OTP cipher, the cipher is decoded when the book is discovered. Since we use the Lorenz's attractor to compute a pad, although it is difficult, one can compute the Lorenz's trajectory and discover the cipher pad. To make this task harder to perform, we have added an operation mode in chaos, which we call the chaotic operation mode. The chaotic operation mode is a technique, inspired by traditional cryptography (operation mode[23]), which provides an iteration between the password, already coded message and chaotic system to make a complicated walk over the attractor.

The presentation of this paper starts with a discussion and a description of the chaotic operation mode (Sec. 2). The cryptographic algorithm is proposed in detail in Sec. 3 and the results are shown in Sec. 4, illustrating the behavior of the proposed method. In Sec. 6, the paper ends with a discussion about the method.

2. Chaotic Operation Mode

Cipher-block algorithms are able to carry out bit-to-bit operations between the neighbors of the plaintext and the password. This approach has a disadvantage of making similar ciphertext sequences for similar blocks of the plaintext, consequently making a weak cipher. The operation mode was created to correct this fault, whose main idea is to make the blocks independent. Consequently, the ciphertext is encrypted combining the information of the password plus the previous plaintext sequence. It can ensure that similar blocks of the plaintext are codified as different ciphertext, making the cipher stronger.

One of the most popular operation mode algorithms is Cipher-Block Chaining (CBC).[23] Each component of the plaintext $\mathbf{P}$, of size n_p is *XORed* with the previous ciphered element: $C_i = E_{\boldsymbol{\pi}}(P_i \oplus C_{i-1})$, where C_0 is an arbitrary value used to encode the first component of $\mathbf{C}$ that can also be used as an extra password. This value has an important role in the crypt process, since without it one cannot decode the message. The value of C_0 is generated from the password and it is desirable to include an additional parameter which may be different for each user. The symbol $\oplus$ represents the bitwise XOR logical operation. We notice that the size of $\mathbf{C}$ is $n_c = n_p + 1$ and that the elements of $\mathbf{P}$ and $\mathbf{C}$ are integers ranging from 0 to $2^8 - 1$ representing the decimal ASCII code. Here, the encode function $E_{\boldsymbol{\pi}}$ represents any symmetrical cryptography algorithm[1–4] using password $\boldsymbol{\pi}$. The operation mode uses information from the

previous coded sequence (C_{i-1}) to code C_i to ensure that similar sequences in $\mathbf{P}$ do not have the same sequences in $\mathbf{C}$. The inverse process also combines previous coded sequence to decode: $P_i = D_{\boldsymbol{\pi}}(C_i \oplus C_{i-1})$, where $D_{\boldsymbol{\pi}}$ is the decode function.

We observe that encryption and decryption operation can be carried out by the same procedure due to the reversible property of the bitwise XOR logical operator. To illustrate the reversibility property, consider two bit sequences $\mathbf{P}$ and $\boldsymbol{\pi}$ and the resulting bit sequence $\mathbf{C} = \mathbf{P} \oplus \boldsymbol{\pi} = \boldsymbol{\pi} \oplus \mathbf{P}$. One then has $\mathbf{P} = \mathbf{C} \oplus \boldsymbol{\pi} = \boldsymbol{\pi} \oplus \mathbf{C}$ and $\boldsymbol{\pi} = \mathbf{P} \oplus \mathbf{C} = \mathbf{C} \oplus \mathbf{P}$. There are other operators that allow the reverse operation, for instance, as mentioned in the introduction, the mod combined with a sign inversion in the argument. One interesting cryptographic scheme uses cellular automata.[24] In this case, the cellular automata must be reversible to preserve the information of the initial states.[25–27] For a cellular automaton to be reversible, the global function must be bijective to have an inverse. Local functions that involve only bitwise XOR logic operation lead to a linear global function, therefore, a bijective function (with non-singular transition matrix).[28,29]

Since the preserved patterns of the plaintext in the ciphertext may give clues for cipher analysis and the cipher-block chain is not strong enough, here, we propose to use dynamical systems in the chaotic regime to accomplish this task. We call this procedure the chaotic operation mode.

The chaotic operation mode has the same basis of the traditional operation mode. The message, password and previous code are combined to achieve a strong cryptography not preserving repetitive patterns of the original message. While the operation mode combines the message and the previous coded message using elementary operations, the chaotic operation mode uses a chaotic system to compute this combination. This approach improves the cryptography, making the analysis of it more difficult.

In the following, we show how to use a dynamical system in cryptography. Firstly consider $n_l = n_c$ vectors $\mathbf{r}$, with initial condition $\mathbf{r}_0$. Notice that $\mathbf{r}$ is a vector, where each component represents a variable of the Lorenz attractor so that it is a three-dimensional vector. Each vector is obtained by the iteration: $\mathbf{r}_i = \mathbf{F}[\mathbf{r}_{i-1} + \mathbf{M}(P_i)]$, where $\mathbf{F}(\mathbf{x})$ is a given vectorial function of a vector $\mathbf{x}$ which represents the dynamical system. The components of the vectorial function $\mathbf{M} = (m_1, m_2, m_3)$, with $0 \leq m_i \leq 0.255$, with $i = 1, 2, 3$, transforms the argument P_i (an integer between 0 and 255) depending on the iteration step. Notice that the plaintext characters alter step-by-step the iteration process, which in a parameter region of chaos gives rise to the chaotic operation mode. The elements of the ciphertext are obtained from: $C_i = E_{\boldsymbol{\pi}}[P_i + f(\mathbf{r}_{i-1})]$, which encrypts the scalar P_i, using bit-to-bit operations with bits extracted from the vectors $\mathbf{r}$. The function $f(\mathbf{r})$ extracts a value from the vector $\mathbf{r}$ of the dynamic system. The decode process is simply achieved with the inverse operation: $P_i = D_{\boldsymbol{\pi}}[C_i + f(\mathbf{r}_{i-1})]$.

Recently a security fault in the logistic map based algorithm that could be extended to other chaos system was demonstrated.[30] It is based on the return of the map that can be explored to break into the cipher. The chaotic operation mode solves this security fault of the chaos based cryptography algorithms, since it combines the password and plaintext with the chaos system, making it more secure.

When using the chaotic operation mode we call attention to the fact that if a message is altered we are able to detect it. In this way we guarantee its authentication. Besides, we can also guarantee the integrity of cipher. The chaotic operation mode can guarantee the authentication because a minimal change in C_i leads to a huge modification in the Lorenz's path, that can be easily identified, assuring integrity and consequently its authenticity.

3. Lorenz-Based Cryptography Algorithm

We propose a full method that combines the dynamic system $\mathbf{F}(\mathbf{x})$ with the encode function $E_{\boldsymbol{\pi}}$. In our case, the dynamical system is the Lorenz's attractor that generates a trajectory which is combined with the message, password with the chaotic operation mode. We start defining the quantities used in the algorithm and then we carefully describe it.

3.1. *Definitions*

The proposed algorithm uses the Euler differential discretization method to numerically compute the Lorenz's equation system (other numerical methods could be considered) $\mathbf{r}_{i+1} = \mathbf{F}(\mathbf{r}_i)$, with $\mathbf{r} = (x, y, z)$ and:

$$\begin{aligned} x_{i+1} &= [(1-\sigma)x_i + \sigma y_i]\delta t \\ y_{i+1} &= (\rho x_i - 2y_i - x_i z_i)\delta t \\ z_{i+1} &= [x_i y_i + (1-\beta)z_i]\delta t\,. \end{aligned} \tag{1}$$

The parameters $0 < \delta t \leq 0.027$, $\sigma = 10.0$, $\rho = 28.0$ and $\beta = 8/3$ are values where the system gives rise to chaotic sequences. The number of iterations n_{it}, along the Lorenz's attractor, determines the speed of the processing of the algorithm. Here we have used $n_{it} = 3000$.

The password $\boldsymbol{\pi}$ is converted to decimal ASCII values, with size n_π. Notice that n_π has a limit, this is because of the representation of a real number in double precision in a given computer. The quantity $\xi = 52$, according to the IEEE754 convention, is the number of bits of the mantissa to represent a double precision number.

The first stage of the algorithm is devoted to the conversion of the password into a coordinate of the Lorenz's attractor space. The vector $\boldsymbol{\pi}$, of size n_π, is converted into a vector $\mathbf{a} = (a_1, a_2, a_3)$, with the following components:

$$a_1 = \begin{cases} \sum_{i=1}^{L} \pi_i 2^{8(i-1)} & \text{if } n_\pi \bmod 3 = 0 \\ \sum_{i=1}^{L} \pi_i 2^{8i} + \pi_{3L+1} & \text{if } n_\pi \bmod 3 \neq 0 \end{cases} \tag{2}$$

$$a_2 = \begin{cases} \sum_{i=L+1}^{2L} \pi_i 2^{8[-L+(i-1)]} & \text{if } n_\pi \bmod 3 \neq 2 \\ \sum_{i=L+1}^{2L} \pi_i 2^{8(-L+i)} + \pi_{3L+2} & \text{if } n_\pi \bmod 3 = 2 \end{cases} \tag{3}$$

$$a_3 = \sum_{i=2L+1}^{3L} \pi_i * 2^{8[-2L+(i-1)]}\,, \tag{4}$$

where $L = \text{floor}(n_\pi/3)$. The vector $\mathbf{a}$ is then converted to $\mathbf{a}' = (g(a_1), g(a_2), g(a_3))$, where $g(x) = x/10^{\text{ceil}}(\log[2^{8(L+1)}])$ so that $0 \leq a'_i \leq 1$, with $i = 1, 2, 3$.

The starting point to move in the Lorenz's attractor is written as:

$$\mathbf{r}_0 = \mathbf{a}' + \boldsymbol{\lambda}\,, \tag{5}$$

with $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3)$. The components of $\boldsymbol{\lambda}$ are chosen to have values in the following range: $-15.67 < \lambda_1 < 16.01$, $-11.28 < \lambda_2 < 16.01$ and $0.090 < \lambda_3 < 62.000$. These ranges guarantee stable chaotic phases.

Firstly, compute $\boldsymbol{\mu} = (\mu_1, \mu_2, \mu_3)$, so that: $\mu_1 = (a_1 + a_2 + a_3) \bmod 3$, $\mu_2 = (a_1 * a_2 + a_3) \bmod 3$ and $\mu_3 = (a_1 + a_2 * a_3) \bmod 3$. Compute now, $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3)$, so that:

$$\alpha_i = \begin{cases} x_n\,, & \mu_i = 0 \\ y_n\,, & \mu_i = 1 \\ z_n\,, & \mu_i = 2 \end{cases}\,, \tag{6}$$

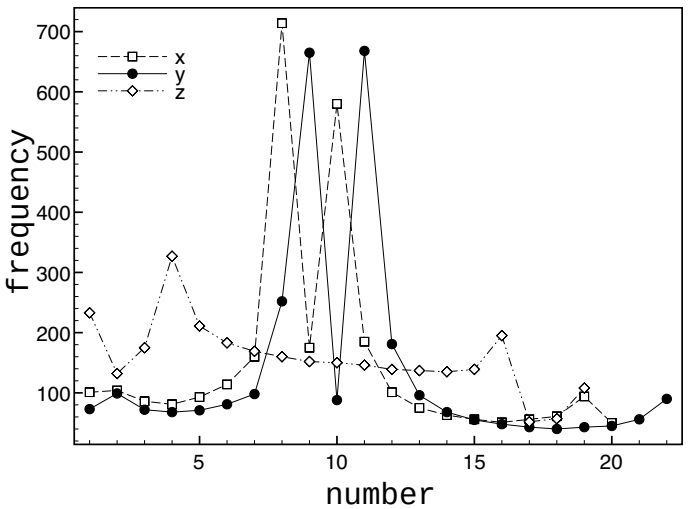
with $n = 0$. Now, choose values for $\mathbf{k} = (k_1, k_2, k_3)$, with the components of this vector in the range: $2 < k_i \leq \text{floor}[(\xi - 14)/8]$, so that k_i being an integer. Finally, compute $\boldsymbol{\Omega} = (\Omega_1, \Omega_2, \Omega_3)$ so that:

$$\Omega_i = \text{hash}(i\mathbf{a}) \mod k_i\,, \tag{7}$$

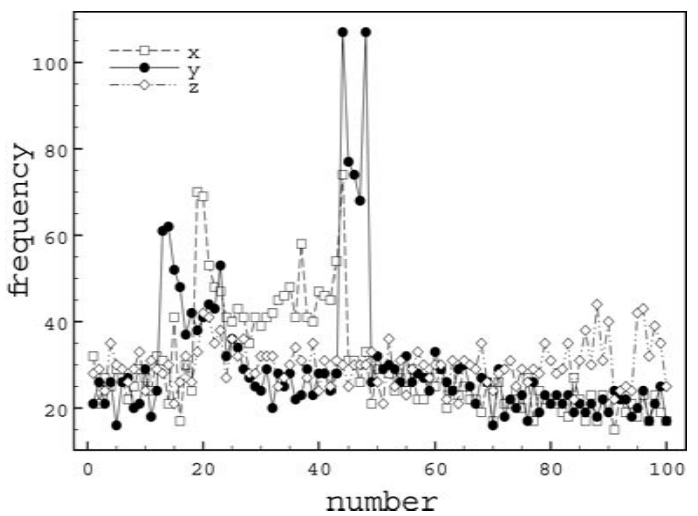
where hash($\mathbf{a}$) is a hash function such as MD5 or SHA. These values have been chosen to obtain the less significant bytes in the coordinates of the Lorenz's attractor. This contrasts to the procedure presented in Refs. 20 and 21 where the integer and first digits from the decimal part are considered. The procedure we propose makes the cipher frequency distribution uniform not giving clues for undesirable decodification as shown in Sec. 4. In Fig. 1 we depict the frequency distribution of the integer part as well as the decimal part of the Lorenz's attractor. In Fig. 1(a), we show that the integer part of coordinates of the Lorenz's attractor are concentrated in the middle of the graph varying in a short range. In Fig. 1(b), we show

that the first digits of the decimal part of the coordinates of the Lorenz's attractor are concentrated. In Figs. 1(c) and 1(d), we show that the frequency distribution is practically uniform for the 3rd, 4th, 5th and 6th pairs of digits. We call attention to the fact that the low significant digits are the head of our algorithm.

To encrypt a plaintext, we propose a new encode function. This function combines an integer (a decimal ASCII of the plaintext) and a float point (obtained

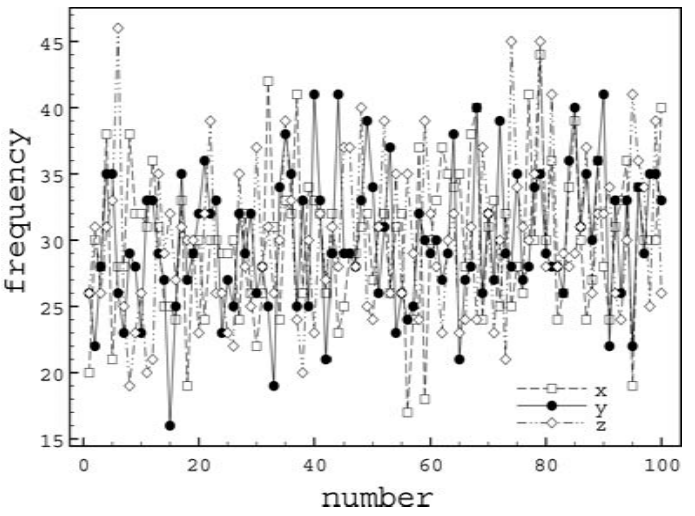


(a)

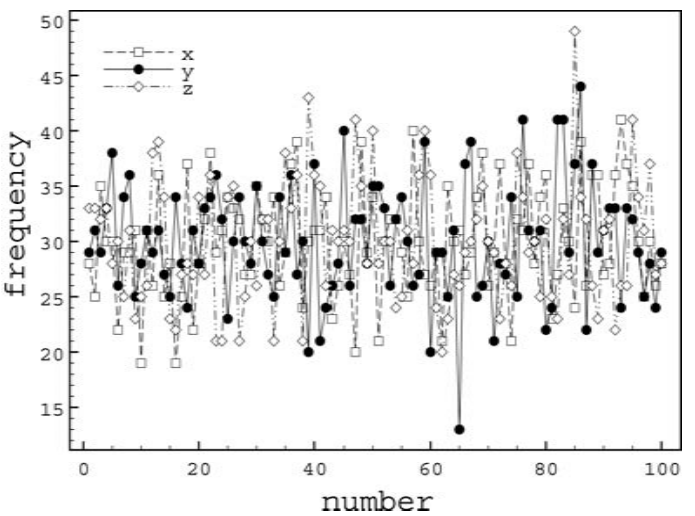


(b)

Fig. 1. Frequency distribution of the integer part as well as the decimal part of the Lorenz's attractor; (a) integer part, (b) first two digits of the decimal part, (c) third and fourth digits of the decimal part, and (d) fifty and sixth digits of the decimal part.



(c)



(d)

Fig. 1. (*Continued*)

from the trajectory in the Lorenz's attractor) in the argument to produce an integer, between 0 and 255, to represent the coded character. For this reason one has a structure as: $E_{\boldsymbol{\alpha},\boldsymbol{\Omega}}(P_i) = [P_i + f(\boldsymbol{\alpha},\boldsymbol{\Omega})] \bmod 2^8$.

Let us now concentrate in $f(\boldsymbol{\alpha},\boldsymbol{\Omega})$, with $\boldsymbol{\alpha}$ and $\boldsymbol{\Omega}$ are quantities related to the password $\boldsymbol{\pi}$ and depend on the Lorenz's attractor. Here we only use the components α_1 and α_2, which are float point numbers and Ω_1 and Ω_2, which are integer numbers.

The algorithm to obtain these quantities in the successive iterations is presented in Sec. 3.2, where we show that α_3 and Ω_3 are auxiliary quantities.

Our objective is the creation of a ternary operation, which operates on a character code and on two float point numbers so that an inverse operation exists, combined with the password $\boldsymbol{\pi}$, and one can easily perform the decode operation. To accomplish this task, consider a float point number α, so that its mantissa has ξ bits and we convert it to an integer multiplying it to 10^{ν}, where $\nu = \text{floor}(\log 2^{\xi-6})$ with the function floor(x) taking the integer part of x. The product floor$(\alpha 10^{\nu})$ is an integer with 16 bytes. Recall that the integer 255, is a byte of 1's. To move this eight 1's to the left, one multiplies it by 2, so that, multiplying 255 to 2^8, one displaces 255 of one byte. To displace 255 of Ω bytes, one must multiply it to $2^{8\Omega}$. The number $255 \times 2^{8\Omega}$ is a mask. To obtain the Ωth byte of the quantity floor$(\alpha 10^{\nu})$, apply the logical AND operation, represented by $\wedge$, with the mask $255 \times 2^{8\Omega}$. A resulting large float number is obtained dividing the above result by $2^{8\Omega}$. Consider the function:

$$R_{\nu}(\alpha, \Omega) = \frac{\text{floor}(\alpha 10^{\nu}) \wedge (255 \times 2^{8\Omega})}{2^{8\Omega}}\,. \tag{8}$$

Now we are able to write the encode function as:

$$y = E_{\boldsymbol{\alpha},\boldsymbol{\Omega}}(x) = \left[x + \sum_{i=1}^{2} R_{\nu}(\alpha_i, \Omega_i)\right] \mod 2^8\,, \tag{9}$$

and the function used to decode the information as:

$$x = D_{\boldsymbol{\alpha},\boldsymbol{\Omega}}(y) = \left[y - \sum_{i=1}^{2} R_{\nu}(\alpha_i, \Omega_i)\right] \mod 2^8\,. \tag{10}$$

To prepare data for the following steps set: $\boldsymbol{\alpha}' = \boldsymbol{\alpha}$, $\boldsymbol{\mu}' = \boldsymbol{\mu}$ and $\boldsymbol{\Omega}' = \boldsymbol{\Omega}$. We call attention to the reversibility of the mod operation of Eqs. (9) and (10). We have chosen to use the mod operator instead of the bitwise XOR operator because its full reversibility asks for an additional operation (inversion of the argument). Another important aspect is that the reversibility of mod is not so obvious as a simple change of the state of bits. These two points combined make the codification more difficult to be broken in with mod.

We have chosen to use the mod operator, which gives the rest of a division, instead of the bitwise XOR operator because to have a full reversibility, an additional operation (inversion of the argument) is necessary in mod operation. This makes the codification more difficult to be broken in.

3.2. *Successive iterations*

The process of encrypting and decrypting a message can be divided into three steps, which are shown below. The dynamic system iterates the components of **P** or **C**.

Some parts of the algorithm are different for the first iteration. This is necessary because the method uses variables calculated in the previous iteration.

Step 1. The function E (encryption) or D (decryption) is used to compute P_i or C_i: $C_i = E_{\boldsymbol{\alpha},\boldsymbol{\Omega}}(P_i)$, for encrypting and $P_i = D_{\boldsymbol{\alpha},\boldsymbol{\Omega}}(C_i)$, for decrypting, where $E_{\boldsymbol{\alpha},\boldsymbol{\Omega}}$ and $D_{\boldsymbol{\alpha},\boldsymbol{\Omega}}$ are given by Eqs. (9) and (10).

Step 2. The chaotic operation mode is carried out. Consider the quantity Ω_3, the password $\boldsymbol{\pi}$, and $\Theta = P_i/10^{3+\Omega_3}$. The Lorenz's attractor is calculated to shuffle the elements and ensure that there are no possibility of similarity of cipher blocks when there is a similarity in the blocks of the plaintext. The chaotic operation mode changes one of the Lorenz's parameters, adding Θ to the x component if $\mu_3 = 0$, to y if $\mu_3 = 1$ or to z, if $\mu_3 = 2$.

Step 3. The quantities μ_i are calculated as: $\mu_i' = [\mu_i + R_\nu(\alpha_i, \Omega_i)] \bmod 3$ where R_ν is given by Eq. (8). The components of $\boldsymbol{\alpha}$ are given by Eq. (6). Consider now k_3 an integer in the range $0 < k_3 < \nu - 2$ and calculate $\Omega_i = [\Omega_i' + R_\nu(\alpha_{[(i+2) \bmod 3]+1}, \Omega_i')] \bmod k_i$, where R_ν is given by Eq. (8). Finally set: $\boldsymbol{\alpha}' = \boldsymbol{\alpha}$, $\boldsymbol{\mu}' = \boldsymbol{\mu}$ and $\boldsymbol{\Omega}' = \boldsymbol{\Omega}$, $\mathbf{r}_n = \mathbf{r}_n + \mathbf{a}'$.

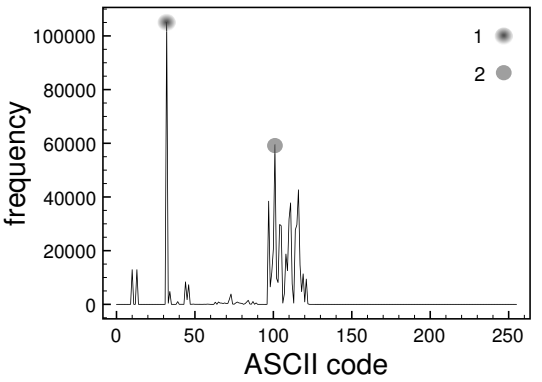
We stress that this algorithm may be adapted to use other dynamical system that presents chaos as the maps obtained from discrete population models.[31]

4. Experiments and Results

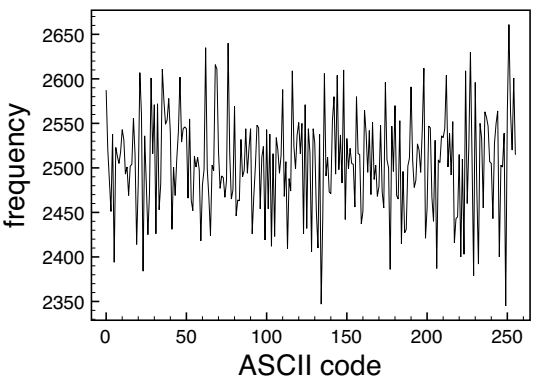
Perhaps, there is no cipher that cannot be broken into, and never will be, at least, using conventional computers and numerical mathematics. History demonstrates that, many ciphers, which were considered invincible were in fact broken into. Indeed, the cryptography and cryptoanalysis are in constant conflict, in which there are no permanent winners. New ciphers are always appearing and new cryptoanalysis methods emerging to break into them (the red queen effect). As a result, it is hard to predict, how difficult it is to break into a cipher. Nevertheless, there are some metrics that can help the analysis and can estimate how much the cipher makes the ciphertext shuffled and unintelligible, allowing for a supposition about how strong the cipher is.

In the following, two experiments with the proposed algorithm are presented. Firstly, the algorithm encrypts a text message and an image. The analysis of the shuffled capability of the method is presented. Besides the security, the computer performance is another important point for a cipher to be able to be used in real world applications. Secondly, the results of the performance for the two versions of the sequential and parallel algorithm are presented. The results are compared with a popular AES method, which is used commercially nowadays.

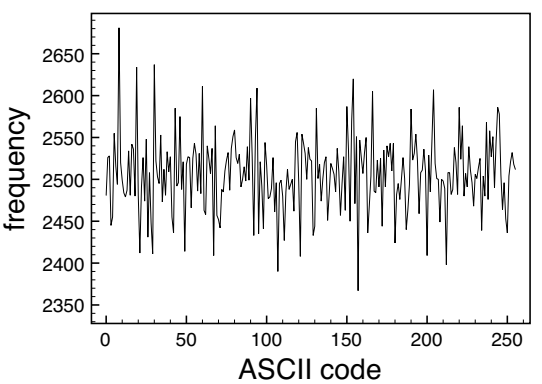
When a ciphertext is yielded, several analysis estimating the strength of the cryptography are made. In the text experiment, the book "The Return of Sherlock



(a)



(b)



(c)

Fig. 2. The frequency analysis of the book "The Return of Sherlock Holmes" using slightly different passwords. (a) Histogram of the plaintext the circle 1 represents the frequency of character "space" and the circle 2 represents the frequency of character "e," (b) histogram of the ciphertext using "123456" as a password, and (c) ciphertext histogram using "123457" as a password.

Holmes" was encrypted. Long texts, such as books, are easier to be broken into when compared with short messages since statistical patterns may be explored.

One of the most well known methods of cryptoanalysis is the frequency analysis. It is quite simple and consists of carrying out a histogram, which presents the ASCII characters frequency, giving tips about their frequency pattern for the cryptoanalysis. The frequency analysis is carried out and displayed in Fig. 2. It illustrates the histogram of the considered book, comparing the plaintext [Fig. 2(a)] with the ciphertext [Figs. 2(b) and 2(c)]. Notice that, the distribution is almost constant in the ciphertext, like a white noise, which illustrates a good performance for the crypto. Figures 2(b) and 2(c) present the book's histogram encrypted with two slightly different passwords: "123456" and "123457." The two histograms have different plots, illustrating the way the cryptography result depends on the password.

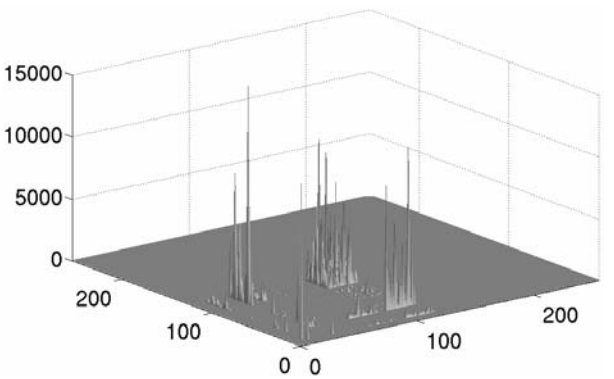
The Shannon's entropy is another approach to estimates how unintelligible the ciphertext is. For an eight-bit codification the text entropy is $S = -\sum_{i=0}^{2^8-1} p_i \log_2(p_i)$, where p_i is the relative frequency of the ith ASCII character. The maximum entropy value is obtained when all the p_i are the same ($p_i = 1$; 2) leading to $S_{\max} = 8$. The entropy of the book's plaintext is 4.5 while the entropy of the ciphertext attains its maximum value $S = S_{\max} = 8$. The entropy achieved by the cryptography is the same as the theoretical prediction for the system. This result shows that the ciphertext has a very low level of redundancy or predictability.

In Fig. 3 auto-correlation matrices are shown, comparing the plaintext [Fig. 2(a)] with the ciphertext [Fig. 2(b) and a white noise signal with the same length of the book (c)].

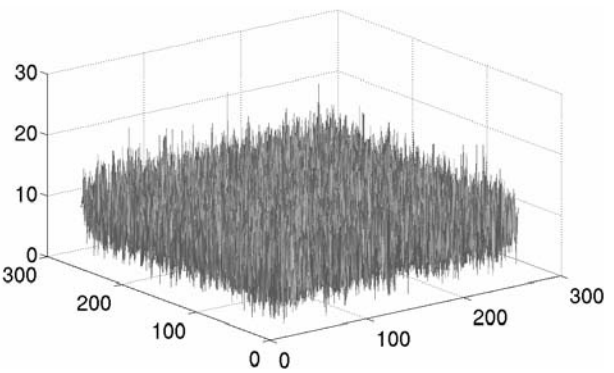
In the second experiment, the cipher was evaluated encrypting an image. Images are also appropriate to cryptoanalysis, as they have strong global and local patterns and redundancy. The experiment is to compare three images: the original one, the cipher image and white noise image. The frequency analysis has been performed and compared it to 2D Fourier power spectrum of the images (see Fig. 4). Notice that while the original image histogram has a lot of information concerning the image and both the cipher image and noise histograms are similar. This fact shows how unintelligible the encrypted image is. Unlike the spectrum of the original image, the spectrum of the noise and the cipher image does not present any frequency information, which demonstrates that the information cannot be achieved in the cipher image.

5. Fast and Parallel Version

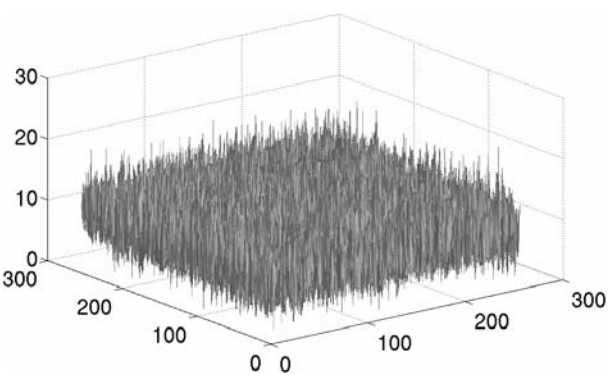
Performance is the most important obstacle that prevents practical use of chaos based cryptography algorithms. In fact, they need float point and numerous calculations, which make the chaos cryptography programs much slower than the traditional ones. To make the proposed algorithm feasible, we have improved its performance using parallel computation, which is in current use nowadays via the



(a)



(b)



(c)

Fig. 3. The auto-correlation matrices of the book "The Return of Sherlock Holmes." (a) Auto-correlation matrix of the plaintext, (b) auto-correlation matrix of the ciphertext, and (c) auto-correlation matrix of a white noise signal.

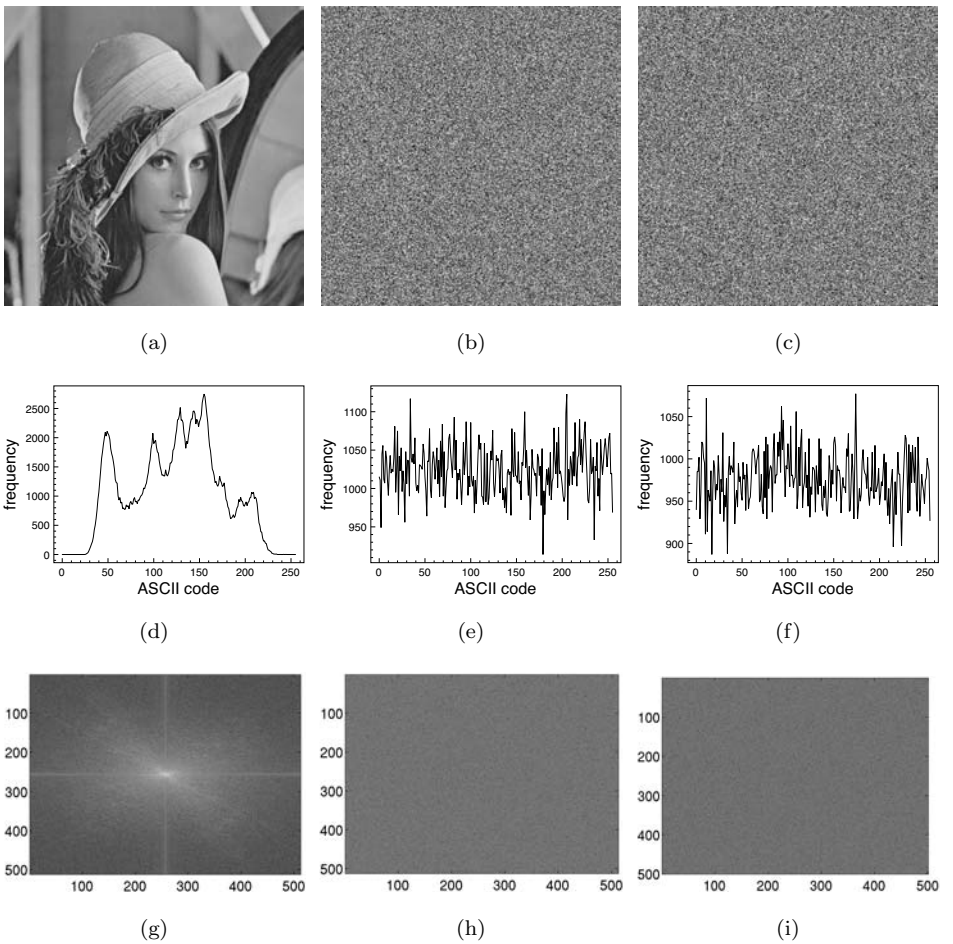
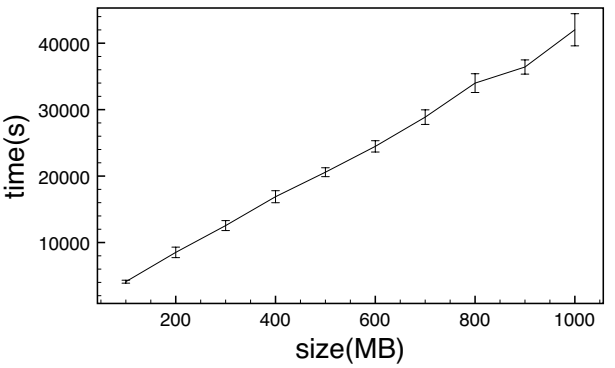


Fig. 4. Comparison of a plain image, cipher image and white noise. In the first row, the matrices analyzed are shown: (a) plain image, (b) cipher image, and (c) white noise. In the second, the histogram analysis of the: (d) plain image, (e) cipher image, and (f) white noise. And in the last row, there is the Fourier power spectrum of: (g) plain image, (h) cipher image, and (i) white noise.

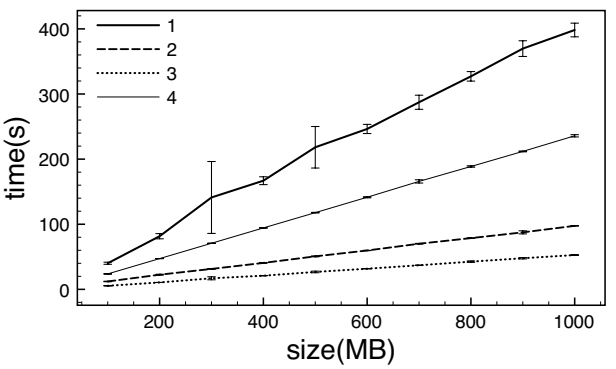
increase of multiple cores in the computer processor or with the use of graphical cards with multiple GPUs (Graphical Processing Unit).

The nature of our algorithm allows an intuitive parallelization. Consider a message of size n and a machine with p processors. The parallel algorithm splits the message into p parts and executes each part into a different processor. For each process a particular initial condition is selected.

The computer performance of the cryptography algorithm is an important item to make it feasible for real applications. The algorithm has been implemented in Java and also in C (fast version). The choice of Java was made due to its portability and capability to run on multiple operational systems and computer hardware (also small devices such as mobile phones). Two versions of the algorithm have



(a)



(b)

Fig. 5. Performance graphs. (a) Sequential performance of the proposed algorithm size in MBytes versus time in seconds. (b) Comparison of the performance of the proposed algorithm in the GPU version and the AES cryptography method. Line 1 represents the performance of the proposed algorithm in a Nvidia geforce 8800 GT. Line 2 represents the performance of the AES algorithm in an Intel Pentium 4 3.4 Ghz. Line 3 represents the performance of the AES algorithm in an Intel QuadCore Q8200 and line 4 represents the performance of the proposed algorithm in a Nvidia GTX 285.

been implemented: the full proposed method as described in Sec. 3 and its parallel version described above. In the parallel version, the chaotic operation mode has been adapted to make the algorithm easy to be parallelized. This version is weaker than the original version, but it is faster and also highly parallelized. There is also a performance comparison of this implementation with the popular AES cryptography method[4] shown in Fig. 5.

Figure 5(a) shows the plot of the time consumption versus the size of the file encrypted for the sequential strong version. The machine used was a Pentium 4–3.40 GHz, with 3 GB of memory and which runs Gentoo Linux distribution and a Intel QuadCore Q8200 with 4 GB of memory and which runs Debian Linux 5.0 distribution. The algorithm has an average rate of approximately 1.5 MB/min. Note

that the program was implemented in Java and can achieve a performance of three or four times faster if it is built using the C language. Although this performance is very slow, it can be considered usable for high security purpose tasks.

To improve the performance of the algorithm, a simpler (but still strong), fast and parallel version has been developed (Subsec. 5). The fast parallel version was implemented in C and, to explore mass computer parallelism, the GPU (Graphics Processing Unit) architecture was adopted (8800 GT graphical card) and the CUDA (www.nvidia.com), a GPU programing tool for C, was used. Figure 5(b) presents the performance of the program compared with the AES. Both programs run on the same computer (described previously). The performance of the chaos based algorithm, which is approximately (approximate 2.5 MB/s in a Nvidia Geforce 8800 GT and approximate 4.3 MB/s in a Nvidia GTX 285) is close to the AES (approximate 8.3 MB/s in an Intel Pentium 4 3.4 Ghz and approximate 18 MB/s in a Intel Quad-Core Q8200). The great performance of the fast and parallel implementation allows immediate use of the chaos cryptography algorithm in real life applications, which has an advantage of being more secure then the traditional cryptography approach.

6. Conclusion

In this paper, a new cryptograph algorithm based on the Lorenz's attractor and a novel method to carry out the operation mode with chaos (Chaotic Operation Mode) were proposed. The proposed algorithm in combination with the Chaotic Operation Mode achieves a strong cipher. The novel method has been implemented in Java (sequential strong version) and also in CUDA (parallel fast version) and experiments present the performance of cryptography and as well as the time of the algorithm has been presented and discussed. The performance of the method and the comparison with the AES algorithms demonstrate that the method is suitable and prepared for real life applications. The analysis of the algorithm and of the results, we notice that the algorithm is strong and perhaps very difficult to break into. However, it is very difficult to determine in fact, how hard it is to break into a cipher. Besides the set of experiments conducted and presented, a web version of the algorithm has been implemented. We would like to invite cryptoanalysts to try to break into the proposed cipher, and help us to determine the reliability of the method and of course, to help develop new methods. Cryptology is a dynamic science which is forever changing. The cryptography page can be found at http://www.mandelbrot.ifsc.usp.br/crypto_lorenz, and the reader can find an interface for encode and decode messages and files.

Acknowledgments

A. G. M. acknowledges support from CNPq (I. C. Grant). A. S. M. acknowledges the support of CNPq (303990/2007-4, 476862/2007-8). O. M. B. would like to thank CNPq (306628/2007-4).

References

1. E. Biham, R. Anderson and L. Knudsen, *In Fast Software Encryption '98* (Springer-Verlag, New York, 1998), pp. 222–238.
2. B. Schneider, J. Kelsey, D. Whiting, D. Wagner, C. Hall and N. Ferguson, *First Advanced Encryption Standard (AES) Conference*, 1998.
3. X. Lai and J. L. Massey, *A Proposal for a New Block Encryption Standard* (Springer-Verlag, New York, 1991), pp. 389–404.
4. J. Daemen and V. Rijmen, *The Design of Rijndael: AES — The Advanced Encryption Standard* (Springer-Verlag, New York, 2002).
5. M. Diffie and W. Hellman, *Information Theory, IEEE Transactions* **22**, 644 (1976).
6. L. A. R. L. Rivest and A. Shamir, *Commun. ACM* **21**, 120 (1978).
7. N. Koblitz, *Math. Comput.* **48**, 203 (1987).
8. M. Matsui and A. Yamagishi, *EUROCRYPT* (1992), pp. 81–91.
9. E. Biham and A. Shamir, *CRYPTO '92: Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology* (Springer-Verlag, London, UK, 1993), pp. 487–496.
10. J. P. Nicolas Courtois, *LNCS* **2501**, 267 (2002).
11. X.-Y. Wang and X.-J. Wang, *Int. J. Mod. Phys. C* **19**, 813 (2008).
12. L. Pecora and T. Carroll, *Phys. Rev. Lett.* **64**, 821 (1990).
13. L. Kocarev, K. S. Halle, K. Eckert, L. O. Chua and U. Parlitz, *Int. J. Bifurcation and Chaos* **2**, 709 (1992).
14. U. Parlitz, L. Kocarev, T. Stojanovski and H. Preckel, *Phys. Rev. E* **53**, 4351 (1996).
15. S. Bu and B. Wang, *Chaos Soliton and Fractals* **19**, 919 (2004).
16. K. Murali, *Phys. Lett. A* **272**, 184 (2000).
17. M. Baptista, *Phys. Lett. A* **240**, 50 (1998).
18. W.-K. Wong, L.-P. Lee and K.-W. Wong, *Comput. Phys. Commun.* **138**, 234 (2001).
19. T. Xiang, X. Liao, G. Tang, Y. Chen and K. W. Wong, *Phys. Lett. A* **349**, 109 (2005).
20. R. He and P. G. Vaidya, *Phys. Rev. E* **57**, 1532 (1998).
21. A. Ali-Pacha, N. Hadj-Said, A. M'Hamed and A. Belgoraf, *Chaos Soliton and Fractals* **33**, 1762 (2007).
22. C. Shannon, *Bell System Tech. J.* **28**, 656 (1949).
23. M. Dworkin, Recommendation for block-cipher modes of operation methods and techniques, NIST Special Publication 800-38A.
24. G. M. B. Oliveira, A. R. Coelho and L. H. A. Monteiro, *Int. J. Mod. Phys. C* **15**, 1061 (2004).
25. J. C. S. T. Mora, *Int. J. Mod. Phys. C* **13**, 837 (2002).
26. J. C. S. T. Mora, S. V. C. Vergara, G. J. Martínez and H. V. McIntosh, *Int. J. Mod. Phys. C* **14**, 379 (2003).
27. J. C. S. T. Mora, M. G. Hernández, G. J. Martínez and S. V. C. Vergara, *Int. J. Mod. Phys. C* **14**, 1143 (2003).
28. A. M. del Rey and G. R. Sánchez, *Int. J. Mod. Phys. C* **17**, 975 (2006).
29. A. M. del Rey and G. R. Sánchez, *Int. J. Mod. Phys. C* **20**, 1081 (2009).
30. X.-Y. Wang, C.-F. Duan and N. Gu, *Int. J. Mod. Phys. B* **22**, 901 (2008).
31. A. S. Martinez, R. S. González and A. L. Espindola, *Physica A* **388**, 2922 (2009).