Chaos-Based Key Stream Generator Based on Multiple Maps Combinations and its Application to Images Encryption

Kamel Faraoun
Département d'informatique, UDL University, Algeria

Abstract: *In recent years, growing number of cryptosystems based on chaos have been proposed. However, most of them encounter some problems such as: low level of security and small key space. The key stream generator is the key design issue of an encryption system. It directly determines security and efficiency, but most of the proposed key streams are binary valued, and suffer from short period and limited key space. In this paper, we propose an n-ary key stream generator, based on hierarchical combination of three chaotic maps. We demonstrate that the produced key streams have good statistical properties, such as uniform distribution, δ-like auto-correlation function, near-zero cross-correlation and very height sensitivity to initial conditions, under precision restricted condition. An image cryptosystem is constructed using the proposed approach and proven to be enough secure to resist various attacks. Complexity is analysed and an effective acceleration of chaos-based image cryptosystems is shown to be achievable.*

Keywords: *Chaotic maps, images cryptography, key stream generation, security.*

Received December 1, 2008; accepted January 27, 2009

1. Introduction

In recent years, owing to the frequent flow of digital images across the world over transmission media, it has become essential to secure them from leakages. Many applications as military image databases, confidential video conferencing, medical imaging systems, TV cable, online personal albums, etc. require reliable, fast and robust security system to store and transmit digital images. Most conventional ciphers, such as Data Encryption Standard (DES), International Data Encryption Algorithm (IDEA), Advanced Encryption Standard (AES), Linear Feedback Shift Register (LFSR), *etc*. [1, 2] with high computational security consider plaintext as either block cipher or data stream and are not suitable for image/video encryption in real time because of long execution time due to large data volume and strong correlation among image pixels. The implementation of traditional algorithms for image encryption is even more complicated when they are realized by software.

In the last decades chaotic cryptography has received considerable attention when many researchers pointed out the existence of a strong relation between chaos and cryptography. Actually, both digital and analog chaotic encryption methods have been proposed and analyzed [3-14]. The main advantage using chaos lies in the observation that a chaotic signal looks like noise for the unauthorized users. Secondly, some interesting properties, such as mixing and sensitivity to initial conditions, can be connected with those of good ciphers, such as confusion and diffusion [7]. Moreover, generating chaotic signal is often of low cost with simple iterations, which makes it suitable for the construction of stream ciphers. Chaotic stream ciphers use chaotic systems to generate pseudorandom key stream to encrypt the plaintext element by element. Different chaotic systems have been utilized to generate such key streams: Forré proposed 2-D Hénon attractor [15], Pareek et al. used generalized logistic map [16] when Behnia *et al*. introduced Piecewise Linear Chaotic Map (PWLCM) [5]. The key streams can then be generated from the outputs of considered chaotic systems by different post-processing methods. This is done by extracting some bits from chaotic orbits determined by the interval reached by chaotic orbits, by cascading multiple chaotic systems [8], or by coupling chaotic systems [17].

Generally, a stream cipher algorithm expands a given short random key into a pseudo-random key stream. Encryption by a stream cipher uses a sequence of random numbers to mask a sequence of plaintext of the same length, bit by bit. Although, strictly speaking, using truly random implementation is impossible. In fact, generating truly random number sequence with deterministic algorithms is in a state of sin, following John von Neumann.

In practice, pseudorandom numbers are used instead. The main problem becomes then to generate pseudorandom numbers with "good" properties that meet the need of a key stream. A commonly used Pseudorandom Number Generator (PRNG) is the Linear Congruential Generator (LCG). However, chaotic systems can generate orbits that are not distinguishable from truly random orbits (e.g., they have broad power spectra, and are extremely sensitive to small changes of initial conditions). Accordingly, Chaotic Pseudorandom Number Generators (CPRNG) have attracted more and more attention [18]. In this paper, a new approach is suggested for fast and secure image encryption. We use a special combination of chaotic maps to construct a new key stream generator which is suitable to use for cryptographic systems, and present very interesting properties: uniform distribution, statistical pseudo- randomness and sensitivity to initial conditions variation.

We show that a combination of simple chaotic maps, can lead to a very complex behavior that implies "good" pseudo random sequence. The proposed key stream generator has also a large key space and is very sensitive to initial conditions variation. To demonstrate its efficacy, we employ the so generated key stream generated to construct a diffusion-confusion cryptographic system and apply it to encipher grey level digital images.

The paper is organized as follows. In section 2 the new general chaos-based key stream generator is presented. In sections 3 and 4 principal experimental results are analyzed. Conclusions and summery of most important points are given in section 5.

2. Proposed Approach

2.1. Key Stream Generation

Without loss of generality, let assume that we have a plaintext $P = \{p_1, p_2, \ldots, p_l\}$, a cipher-text $C = \{c_1, c_2, \ldots, c_l\}$, and a key stream $K = \{k_1, k_2, \ldots, k_l\}$, all of length l, where $p_i \in \{0,1,\ldots,n\}$, $c_i \in \{0,1,\ldots,n\}$ and $k_i \in \{0,1,\ldots,n\}$. For any p_i, k_i, there exists a c_i, such that $c_i = E(p_i \oplus k_i)$, and for any c_i, k_i, there exists p_i, such that $p_i = D(c_i \oplus k_i)$ where $E(\cdot)$ and $D(\cdot)$ are the encryption and the decryption functions respectively. Our goal in this work is to design an n-valued keystream generator, using a special combination of chaotic maps. Let's consider a one-dimensional non linear chaotic map $\Gamma_X:I\rightarrow I$, such that $I\subset\Re$, and its corresponding differences equation:

$$X_{m+1}= \Gamma_x(X_m) \tag{1}$$

Given an initial value X_0, $\{X_m, m=1, 2, \ldots\}$ is the corresponding chaotic orbit. Γ_X is a continuous mapping that verifies the mixing propriety, the topological transitivity and the density of periodic points in I. With a proper choice of the initial condition X_0, the generated orbit will be bounded in a limited region that corresponding to the attractor of the system described by equation 1. Let consider X_{max} and X_{min} the upper and the lower boundaries of the attractor and then partition the region $[X_{min}, X_{max}]$ into N disjoint equal sub-regions $\{R_i, 1 \leq i \leq N\}$ such that :

$$[X_{min},X_{max}]=\bigcup_{i=1}^{N} I_i \quad I_i\cap I_j=\varnothing \quad for\ i\neq j \tag{2}$$

A random n-ary sequence S of length N $\{S_i, 1 \leq i \leq N\}$ is then generated, and a one to one mapping is created between each element S_i and the region R_i. The sequence values belong to the set $\{1, 2,\ldots, n\}$ taken with a uniform selection probability. So the number of regions N must be a multiplicand of n to ensure that all values are present with the same proportionality in the sequence, hence ensuring a uniform distribution of the final generated stream. Originally, the association between the sequence elements $\{S_i, 1 \leq i \leq N\}$ and the regions $\{R_i, 1 \leq i \leq N\}$ is at an agreed setting, for example, we can set the original sequence to an ordered sequence of N value such that

$$S_i=(i \bmod n)\ for\ i=1\ldots N. \tag{3}$$

The association will then be:

$$S_1\rightarrow R_1,\ S_2\rightarrow R_2,\ldots\ldots\ldots,S_N\rightarrow R_N \tag{4}$$

When the chaotic equation 1 is iterated, X_i values will be distributed chaotically in the system attractor ($[X_{min}, X_{max}]$) in different manners according to the initial value X_0. At each iteration step t, one can chooses the S_i value corresponding to the region R_i such that $X_t\in R_i$ as the output of the stream generator at the time t. It has been shown that usually such approach leads the key stream to fall rapidly into a short period [19], which will degrade the randomness quality of the stream. To avoid such behavior, the one to one mapping between the sub regions R_i and the random sequence of elements S_i is changed dynamically after each Δ iterations of the map (1). We use the orbit of another chaotic map $\Gamma_Y:I\rightarrow I$ with corresponding differences equation:

$$Y_{m+1}= \Gamma_Y(Y_m) \tag{5}$$

As a pseudo- random sequence to generate the dynamical association. Let Y_0 be a predetermined value from the interval I used as initial condition for equation 5. Dropping the first N_0 iterations of equation 5 we can get its corresponding chaotic orbit:

$$Y_{N0+1}, Y_{N0+2}, \ldots\ldots, Y_{N0+N} \tag{6}$$

With the same length as the sequence S, equal to the number of sub regions R_i. Equation 6 is then rearranged in a decreasing order to obtain a new sequence:

$$Y'_1, Y'_2, \ldots\ldots, Y'_N \tag{7}$$

Such that $Y'_j=Y_{N0+i}$ if Y_{N0+i} is located in the jth position after sorting. The sequence S is then rearranged using the equation 7 and new associations are created like the following:

$$S_{Y'1}\rightarrow R_1,\ S_{Y'2}\rightarrow R_2,\ldots\ldots\ldots,S_{Y'N}\rightarrow R_N \tag{8}$$

This process is repeated after each Δ iterations of the map (1). To ensure randomness, the initial value of equation 5 is changed each time the sequence is generated. In this work, we choose to set the initial value Y^K_0 after each K*Δ iteration to:

$$Y^K_0=Fract(Y_0+X_{K*\Delta}) \tag{9}$$

where Y_0 is a predetermined value from I, $X_{K*\Delta}$ is the last obtained value of the map (1), and Fract(x): give the fractional part of a real, number x.

So the initial condition of the map (5) will take respectively the values $Y^1_0, Y^2_0, \ldots, Y^p_0$, when p depends on the keystream length l and the Δ parameter (p is the number of times the association between the sequence elements S_i and the regions R_i is recreated) . The parameter Δ greatly influences the resulting key stream. Accordingly, we propose to change it dynamically during the iterations instead of fixing its value. We actually find that this enhances the key stream randomness and sensibility to initial parameters. Let use a third chaotic map Γ_Z:I→I with corresponding differences equation:

$$Z_{m+1}=\Gamma_Z(Z_m) \tag{10}$$

Using predetermined initial value Z_0, the generated orbit $\{Z_m, m=1,2,\ldots\ldots\}$ will serve to produce different values of Δ using the formula :

$$\Delta_i=floor\ (Z_{N0+i+1} * 10^{\alpha}) \tag{11}$$

where Δ_i is the number of iterations performed before changing the association, and Z_{N0+i} is the ith value obtained by equation 9 after dropping the first N_0 iterations.

The exponent α is a parameter that depends on the size of the generated stream, and determines the frequency of dynamic association generation. To make good compromise between the execution time and the efficacy of the generated stream randomness, we choose to set this exponent to:

$$\alpha = Floor(\log_{10}(stream\_size))-2 \tag{12}$$

Equation 12 has been determined experimentally and proved to give most appropriate result. Using the maps and the parameters presented above, a keystream of length l can be generated as follows:

1. Iterate N_0 times the maps (1) and (10) for a given values X_0 and Z_0;
2. Set Δ_0 =floor(Z_{N0+1}*10^4) and start iterating (5) from Y^1_0 computed using equation 9 to generate N-value orbit;
3. Rearrange the sequence S using the Γ_Y produced orbit and create the association with regions $\{R_i, 1 \le i \le N\}$;
4. Iterate (1) for Δ_0 time and produce a key stream element at each iteration using the association and the Γ_X produced orbit :

$$k_i=S_j \text{ such that } X_i \in R_j \tag{13}$$

5. Compute the new Δ_1 using equation 11, Y^1_0 using equation 9 ;
6. Repeat steps 3 and 4 until we get the desired stream length.

The block diagram of the proposed algorithm is illustrated in Figure 1. This algorithm can be used by choosing any combination of chaotic maps Γ_X, Γ_Y and Γ_Z that verifying the mixing property and sensibility to initial conditions. Furthermore, different values of n can be used to produce keystreams with different scales (e.g., binary if n=2). In our experiments, we choose the logistic map defined by:

$$\Gamma(x)=\mu.x.(1-x) \tag{14}$$

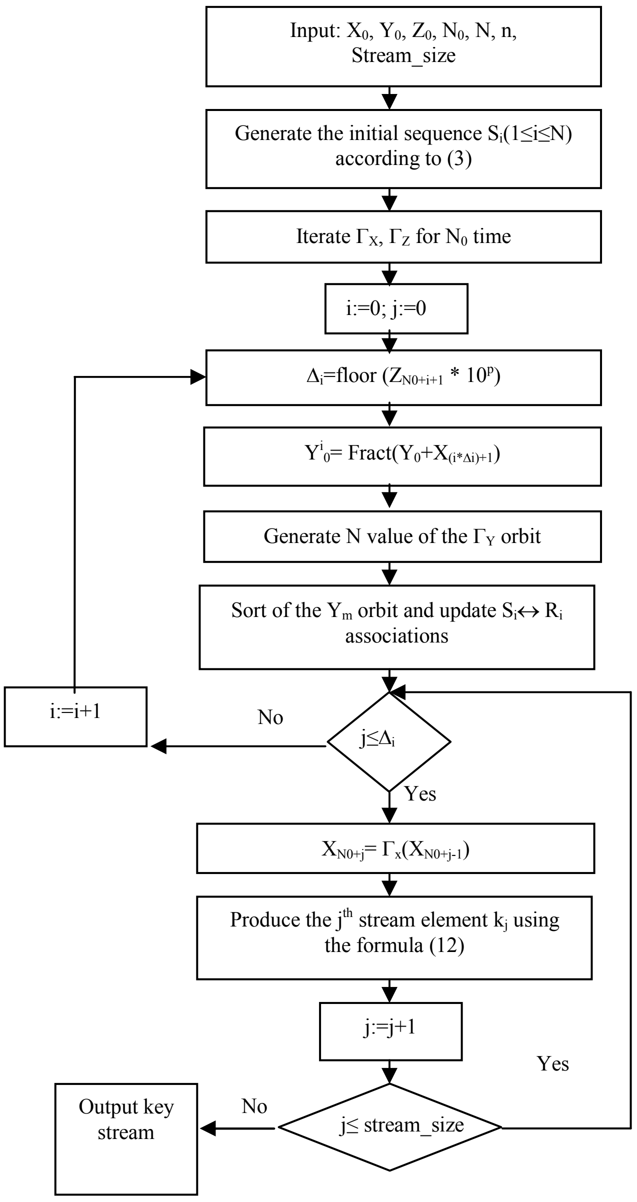

Input: X_0, Y_0, Z_0, N_0, N, n, Stream_size
Generate the initial sequence $S_i(1\le i\le N)$ according to (3)
Iterate Γ_X, Γ_Z for N_0 time
i:=0; j:=0
Δ_i=floor (Z_{N0+i+1} * 10^p)
Y^i_0= Fract($Y_0+X_{(i*\Delta i)+1}$)
Generate N value of the Γ_Y orbit
Sort of the Y_m orbit and update $S_i\leftrightarrow R_i$ associations
i:=i+1
No
$j\le\Delta_i$
Yes
$X_{N0+j}=\Gamma_x(X_{N0+j-1})$
Produce the jth stream element k_j using the formula (12)
j:=j+1
Yes
Output key stream
No
j≤ stream_size

Figure 1. Diagram block of the proposed key stream generation approach.

To represent Γ_X, Γ_Y and Γ_Z. This map has chaotic behavior [20] in the interval I = [0, 1] when $\mu \in [3.57, 4]$. Table 1 summarizes the different parameters used to experiment the proposed approach. Variable parameters can be used as the key of generated key stream.

Table 1. Different parameter of the key stream generation process.

Parameter	Description	Value
μ	Logistic map seed	4.0
n	Stream output maximum bound	256
X_{min}	Attractor lower bound	0
X_{max}	Attractor upper bound	1
X_0, Y_0, Z_0	Initial condition of Γ_X, Γ_Y and Γ_Z	Variable
N_0	Iterations initial steps	Variable
N	Number of Regions	Variable
Stream_size	Size of generated stream	Variable

2.2. Design of the Encryption / Decryption System

Based on the approach explained above, one can construct an encryption system using the generated key stream for a diffusion process. We propose in the following a simple confusion/diffusion system to encrypt grey level digital images. The general schema of the proposed cryptosystem is illustrated in Figure 2.

2.2.1. Confusion Stage

The confusion process is realized solely by permuting all pixels by an invertible discretized 2D standard map, without mixing their values [6]. As the corner pixel (x = 0, y = 0) is not permuted at all under the standard map, a random scan couple (r_x, r_y) is included to permute this pixel with another one. The resulting modified standard map equations are given by the following:

$$\begin{cases} x_{k+1} = (x_k + r_x + r_y + y_k) \bmod M \\ y_{k+1} = \left(y_k + r_y + K_C \sin\left(\frac{2\pi x_{k+1}}{M}\right)\right) \bmod M \end{cases} \quad (15)$$

where (x_k, y_k) and (x_{k+1}, y_{k+1}) are respectively the original and the permuted pixel position of an M×M image. The standard map parameter K_C is an integer number.

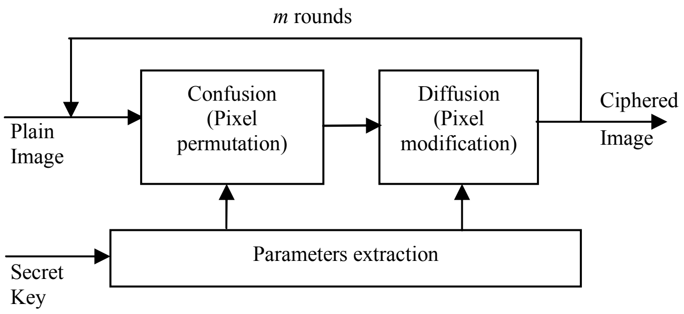


Figure 2. General architecture of the proposed encryption system.

2.2.2. Diffusion Stage

The permuted image of size M×M is first transformed to a one dimensional array of length M*M. If we suppose that the diffusion stage is performed for m times, a key stream of length m*M*M is generated using the approach presented in the section 2.1. At each iteration step, the result of the permutation process is combined with the corresponding element of the key stream as follows:

$$\begin{cases} C_{-1} = K_d \\ C_i = (((p_i + k_i) \bmod 256) \oplus C_{i-1}) \oplus k_i \end{cases} \quad (16)$$

where p_i is the value of the ith pixel of the permuted image, k_i is the ith element of the keystream, c_{i-1} and c_i are the value of the (i-1)th and the ith pixel of the diffused image, respectively. The seed of the diffusion function is c_{-1} obtained from the diffusion key K_d.

2.3. Key Scheming

The key directly used in the proposed encryption scheme is a vector of 7 parameters including diffusion and confusion ones, the three real values X_0, Y_0, Z_0, the integer values K_C, N_0, N and the iterations count m. Real values are coded on 51 bit to ensure a precision of 10^{-15}. We use 51 bit to code K_C, 16 bits to code N_0 and 5 bits for both N and m. These lead to a key size of 230 bit, making the key space as large as 2^{230} possible combination. This is larger than the acknowledged most security AES standard.
K_d, r_x and r_y are directly derived from the 230 bit user key. K_d is coded on 8 bits, when r_x and r_y codification size depend on the image size M (ex: 8 bits for a 256x256 image).

3. Key Stream Properties Analysis

In what follows, different experiments are performed to test the statistical properties of the key stream outputted, and its sensibility to initial conditions. All tests are performed on a 3GHz Intel Pentium (IV), with 1Go memory size and 80Go hard-disk capacity. We use the precision of 10^{-15} easily realized on today's personal computers.

3.1. Keystream Distribution

From the point of view of strict cryptography, chaotic sequences have to satisfy uniform distribution which is most important to prevent any kind of statistical attack. To prove the pseudo-uniformity of the keystream, we use the chi-square test [21] and the Kolmogorov-Smirnov test [22] on 300 generated instances of the key stream with size 10^7, using random combinations of parameters. The chi-square test is applied using:

$$\chi^2_{test} = \sum_{i=1}^{n} \left(\frac{(o_i - e_i)^2}{e_i} \right) \tag{17}$$

where n is the number of levels in the output key stream, o_i, and e_i are respectively the occurrence observed and expected frequencies of each level. When using a significance level of 0.05, we find that $\chi^2_{test} < \chi^2_{255,0.05}$, so the null hypothesis is not rejected and the distribution of the key stream is uniform. The Kolmogorov-Smirnov test is used to test the null hypothesis that the population distribution from which the data sample is drawn is a uniform distribution.

The Kolmogorov-Smirnov statistic for a given function F(x) is:

$$D = \sup_x \left| F_k(x) - F(x) \right| \tag{18}$$

where $F_k(x)$ is the empirical distribution and F(x) is the mathematical expression of the uniform distribution. We have to test the null hypothesis H_0:"$F_k(x)=F(x)$ for all x", against the hypothesis H_1:"$F_k(x) \neq F(x)$ for some value of x". The used critical region of size $\alpha = 0.05$ corresponds to values of D greater than the 0.95 quantile 0.085, obtained from the Kolmogorov-Smirnov table for n=256 [23]. When computing D, we find that $D=0.0634 < 0.085$. So the hypothesis is accepted and the distribution is uniform. Figure 3 shows the obtained histogram of a key stream with size 10^7, obtained using $X_0=0.43384$, $Y_0=0.5728658$, $Z_0=0.229145$, $N_0=10000$ and N=10.

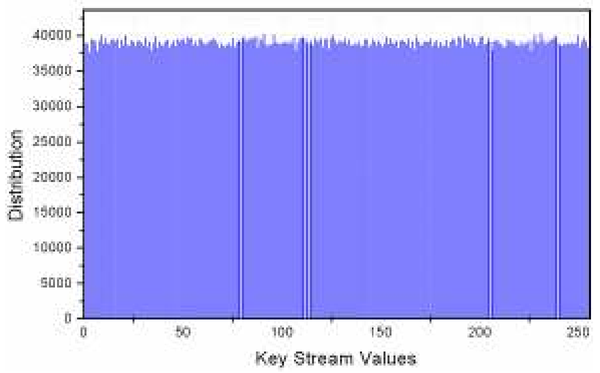


Figure 3. Histogram of the key stream generated using the key values: $X_0=0.43384$, $Y_0=0.5728658$, $Z_0=0.229145$, $N_0=10000$ and N=10.

3.2. Information Entropy

The entropy is the most outstanding feature of randomness [24]. Information theory is a mathematical theory of data communication and storage founded by Claude E. Shannon in 1949 [25]. It is well known that the entropy H of a symbol source S can be calculated as:

$$H(m) = \sum_{i=1}^{2^N - 1} P(m_i).\log_2\left(\frac{1}{P(m_i)}\right) \tag{19}$$

where $P(m_i)$ represents the probability of symbol m_i and the entropy is expressed in bits. Actually, given that a real information source seldom transmits random messages, in general, the entropy value of the source is smaller than the ideal one. If all the symbols have equal probabilities, then the entropy H(m) is equal to 8, corresponding to a truly random source. Let us consider the key stream generated with our approach and containing 256 different symbols and using the same parameters of Figure 1. The number of occurrences of each symbol block is recorded and the probability of occurrence is computed. We obtain the entropy H(m) = 7.999916. The value obtained value is very close to the theoretical one of 8, meaning that information leakage in the generated key stream is negligible and the encryption system can be trusted upon the entropy attack.

3.3. Sensitivity to Initial Conditions

High key sensitivity is required by secure cryptosystems, which means that the cipher text cannot be decrypted correctly although there is only a slight difference between encryption or decryption keys. This guarantees to some extent the security of a cryptosystem against brute-force attacks. In our case, the sensitivity is determined with respect to initial values of the different parameters. The cryptosystem will be enough secure to resist brute force attacks when sensitivity to initial parameters is increased. We use a measure of sensitivity analogous to that used in [26]. The change rate in the keystream K is computed by:

$$\begin{aligned} Cdr(p=p_0) &= \frac{Diff(K,K_1)+Diff(K,K_2)}{2*Size(K)}.100\% \\ Diff(K,K_1) &= \sum_{i=1}^{size(K)} Difp(K[i],K_1[i]) \\ Difp(K[i],K_1[i]) &= \begin{cases} 1 & if\ K[i]=K_1[i] \\ 0 & else \end{cases} \end{aligned} \tag{20}$$

where K is the keystream generated when $p=p_0$, K_1 is the key stream generated when $p=p_0+\Delta p$, and K_2 the key stream generated when $p=p_0-\Delta p$. Cdr is the keystream difference rate. Diff is a function computing the number of different keystream values. We studied the sensitivity of the keystream according to the parameters X_0, Y_0 and Z_0 with Δp set to 10^{-15}, and according to N_0, N with Δp set to 1. In all cases, the stream size was of 10^6 elements. At each time a value is assigned to a parameter, 300 random values are generated for other ones, and the resulting change rates are averaged.

Figures 4 and 5 illustrate the change rate with respect to the variation of real valued and integer parameters respectively. We can see that the value of Cdr is above 97% for all possible values witch ensure high sensitivity to initial parameters. The experimental results also demonstrate that the generated key streams have a δ-like Auto-Correlation (AC) function and a near-zero Cross-Correlation (CC) function, which are shown in Figure 6. We have used the keystream K generated with X_0=0.29384, Y_0=0.6254, Z_0=0.4587254, N_0=24000 and N=8.

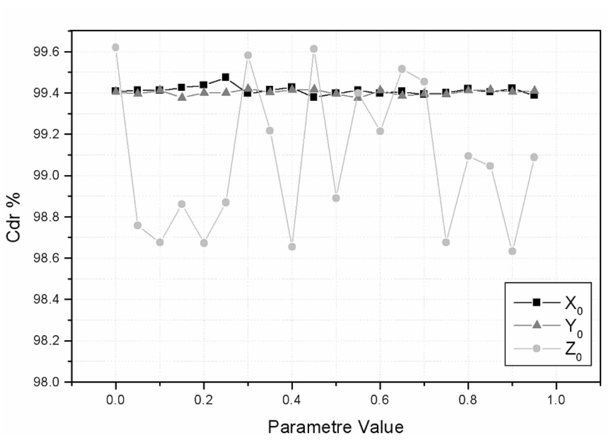
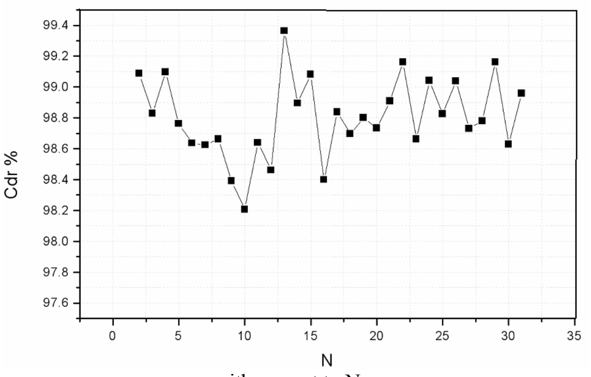
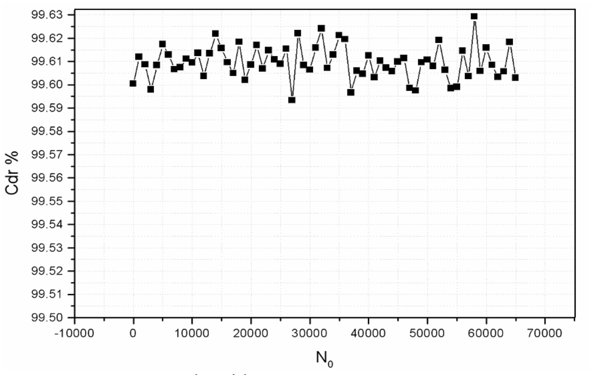


Figure 4. Key sensitivity test of the generated keystream with respect to x_0,y_0 and z_0 parameters.



a. with respect to N.



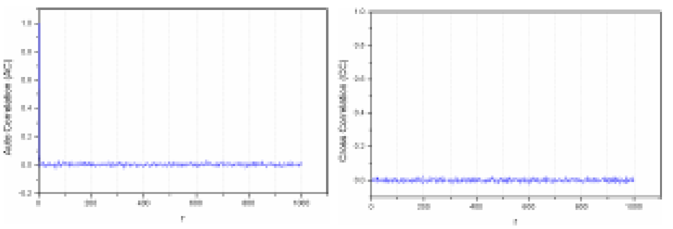
b. with respect to N_0.

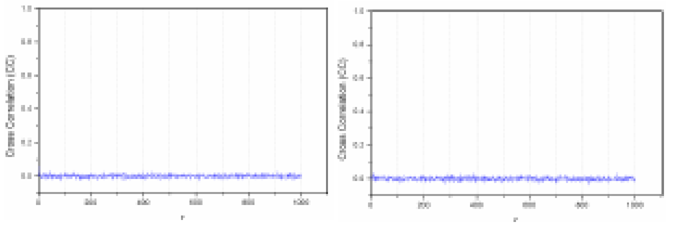
Figure 5. Key sensitivity test of the generated keystream .

CC is computed between K and the key streams: K_1 generated when varying X_0 by $\Delta X_0=10^{-14}$, K_2 generated when varying Y_0 by $\Delta Y_0=10^{-14}$, K_3 generated when varying Z_0 by $\Delta Z_0=10^{-14}$, K_4 generated when varying N_0 by ΔN_0=1 and K_5 generated when varying N by ΔN=1. The AC and CC functions are defined as:

$$ACr(K)=\frac{1}{Size(K)}\sum_{i=0}^{size(K)-1}(K[i]-\overline{K})(K[i+r]-\overline{K})$$
$$CCr(K_1,K_2)=\frac{1}{Size(K_1)}\sum_{i=0}^{size(K_1)-1}(K_1[i]-\overline{K1})(K_2[i+r]-\overline{K2}) \quad \text{such that } K_1\neq K_2 \tag{21}$$

where $\overline{K}$ is the mean value of the key stream k, K[i] is the ith element of K and r is the shift parameter.


a. Auto-correlation. b. Cross correlation with K_1.


c. Cross correlation with K_2. d. Cross correlation with K_3.

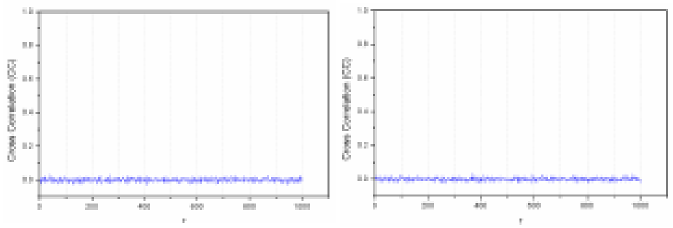
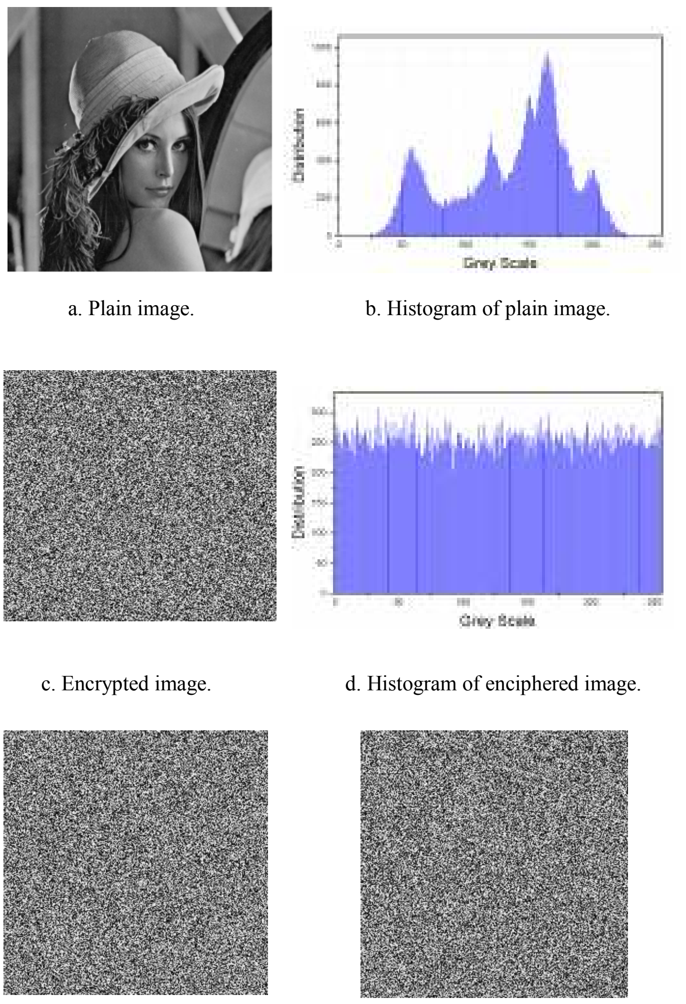

e. Cross correlation with K_4.. f. Cross correlation with K_5.

Figure 6. Correlation of the generated keystream.

4. Cryptosystem Security Analysis

Encryption and decryption results are given in this section demonstrating the efficiency of our proposed algorithm. We take the traditional 256×256 size 8 bits "Lenna" image as example. The positions permutation and the pseudorandom key stream used for grayscale substitution are generated following the scheme proposed in section 2.2. Original image and its histogram are shown in Figures 7(a) and (b), and the encrypted image and its histogram is shown in Figure 7(c) and (d). Initial parameters were: X_0= 0.567676469135816, Y_0= 0.60354653336181, Z_0= 0.339824617849325, N_0= 6067, N=11 K_C = 0.703756097336475 and m=8 corresponding to the 230 bit key: "1395F87B4BCB9514041766CE33B5BE7". In Figure 7(d) we can see the grayscale distribution of the encrypted image has good balance property, and the histogram is significantly different from that of the original image which is secure against known plaintext attack.



a. Plain image. b. Histogram of plain image.

c. Encrypted image. d. Histogram of enciphered image.

e. Image decrypted with one bit different of the correct key. f. Difference between (c) and (e).

Figure 7. Key sensitivity.

4.1. Key Sensitivity

Recall that secure cryptosystem requires not only a large key space but also a high key sensitivity. That is, a slight change in the key should cause some large changes in the ciphertext. This property makes the cryptosystem of high security against statistical or differential attacks. Since the user key is in 230 bits, the key space is about 1.7254 $.10^{69}$, which is sufficient to resist the brute-force attack with the current computer technology.

An encryption scheme has also to be key-sensitive, meaning that a tiny change in the key will cause a significant change in the output. Figure 7(e) shows an example of an enciphered image generated using a security key with only 1-bit difference. It can be observed that the difference values are very close to the expected value of pixel difference on two randomly generated images (99.609375%). Figure 8 shows the values of differences between the correct decrypted image and the different decrypted images with the 230 possible keys that differs in one bit from the correct one.

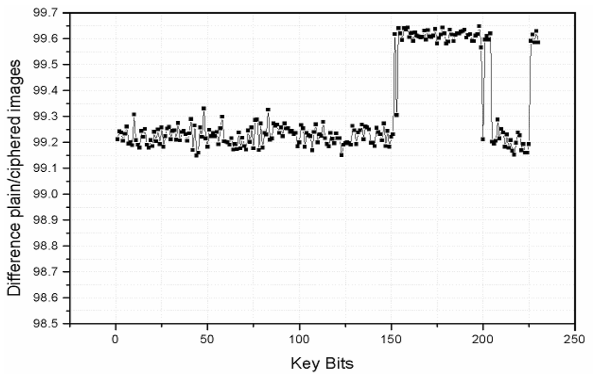


Figure 8. Differences between the corrected decrypted image and those decrypted with the 230 possible keys that differs in one bit from the correct key.

4.2. Correlation of Adjacent Pixels

To test the correlation properties of the enciphered image, we performed statistical analysis on the encryption algorithm. This is done by testing the correlation between two vertically adjacent pixels, two horizontally adjacent pixels, and two diagonally adjacent pixels, respectively.

We randomly select 1000 pairs of two adjacent pixels from the image then we calculate the correlation coefficient of each pair using the following discrete formulas [28]:

$$rxy = \frac{Cov(x,y)}{\sqrt{D(x)}\sqrt{D(y)}}$$

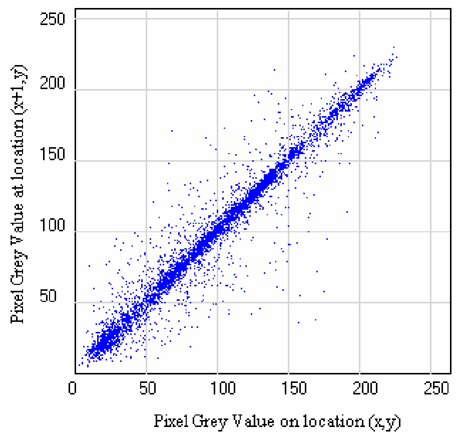
$$\text{where} \quad Cov(x,y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))(y_i - E(y)) \tag{22}$$

$$E(x) = \frac{1}{N}\sum_{i=1}^{N}x_i\,,\ D(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))^2$$

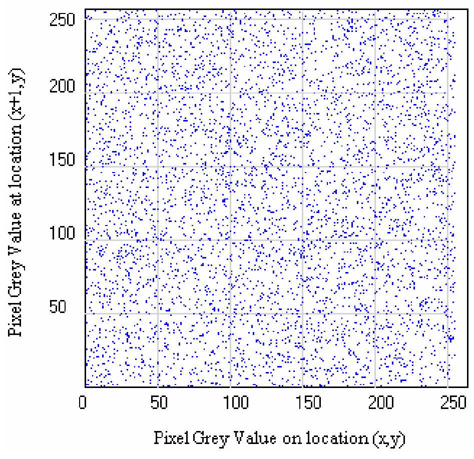
Here, E(x) is the estimation of mathematical expectations of x, D(x) is the estimation of variance of x, and cov(x, y) is the estimation of covariance between x and y. x and y are grey-scale values of two adjacent pixels in the image. Figure 9 shows the correlation distribution of two vertically adjacent pixels in the plain-image and those in the ciphered image. The average correlation coefficients are 0.9792062 and 0.0030121 respectively. Similar results for diagonal and vertical directions were obtained. These are shown in Table 2. It is clearly visible that correlation coefficients are very different when comparing plain and ciphered images.

Table 2. Correlation coefficients of adjacent pixels of plain image, ciphered image and a random one

	Plain image 'Lena'	**Ciphered image**	**Random image**
Horizontal	0.9537214	0.0046591	0.001562
Vertical	0.9792062	0.0030121	0.005922
Diagonal	0.9245871	0.0047814	0.004006



a. The plain Lena image.



b. The cipher image obtained using the proposed scheme.

Figure 9. Correlation analysis of two horizontally adjacent pixels.

4.3. Differential Attacks

It is clear that if one minor change in the plain image can cause a significant change in the ciphered-image, with respect to both diffusion and confusion, then a differential attack may become inefficient. To test the influence of one-pixel change on our whole encrypted image, two common measures are used: NPCR and UACI. The first one stands for the number of pixels change rate while one-pixel of plain image is changed, when the second is the unified average changing intensity that measures the average intensity of differences between the plain and ciphered image.

To calculate NPCR and UACI, consider two ciphered images C_1 and C_2, whose corresponding plain images have only one pixel difference. The gray values of each pixels in C_1 or C_2 is labeled $C_1(i, j)$ or $C_2(i, j)$ respectively. If we define a bipolar array D with the same size as image C_1 or C_2, with D(i, j) being determined by $C_1(i, j)$ and $C_2(i, j)$: if $C_1(i, j) = C_2(i, j)$ then D(i, j) = 0, otherwise D(i, j) = 1. The NPCR will be defined by:

$$NPCR=\left(\frac{1}{M^2}\sum_{i=1}^{M}\sum_{j=1}^{M}D(i,j)\right)\times 100\% \tag{23}$$

where M is the border size of both images C_1 and C_2, and NPCR measures the percentage of different pixel numbers between the two images. The UACI is defined by:

$$UACI=\frac{1}{M^2}\left(\sum_{i=1}^{M}\sum_{j=1}^{M}\frac{|C_1(i,j)-C_2(i,j)|}{255}\right)\times 100\% \tag{24}$$

Tests have been performed on the proposed scheme, about the one-pixel change influence on a 256 grey-scale image of size 256×256. We computed the average of both NPCP and UCAI when changes of each pixel of the image are applied with the 256 possible value of grey level, i.e. average is taken on the 256^3 possible situations. The encryption process was performed using the parameters presented in section 4.

The obtained values of NPCR and UCAI were of 99.51% and 33.45% respectively. These results show that even swiftly change in the original image will result in significant change in the ciphered one. The algorithm proposed has then good ability to anti differential attack.

In Figure 10, we show the progression of the NPCR and UCAI values with respect to the number of iterations (confusion-diffusion) m. The graphs show that both performance indices raise rapidly indicating good confusion and diffusion effects. It is clear that our proposed schema can achieve very accepted performances with relatively small iteration number, compared to other approaches.

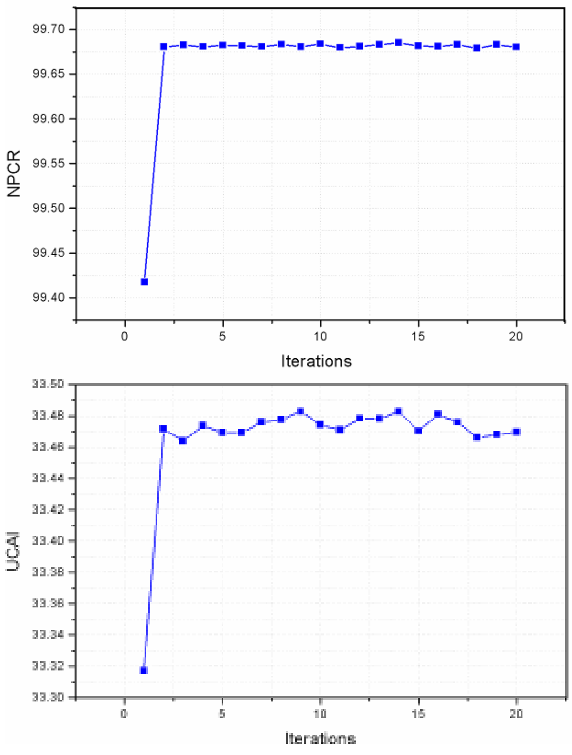


Figure 10. Progress of NPCR and UACI with respect to the number of confusion/diffusion iterations.

4.4. Computational Complexity Analysis

Compared with traditional block ciphers [28] such as DES, IDEA and NSSU, the proposed chaos-based cryptosystem has some distinct properties. Here, tests are performed on the encryption speed of the proposed chaotic cryptosystem. The comparison between different approaches is performed with respect to practical time complexity, when executed in the same environment and conditions. Most part of the implementation was realized using Intel Assembly instruction to guarantee the maximum of execution speed. The speed of the proposed encryption algorithm is much faster than most existing encryption ones. Its average speed is about 30 MByte/s with a Pentium IV 3GHz personal computer. Table 3 shows the result of the encryption speed and speed of some well-known encryption algorithms in Crypto++ Library [29] using the same machine. With such a speed, this image encryption scheme can be used in Internet applications over broadband network, where the encryption and decryption time have to be short compared to the transmission time. Table 4 gives an idea about the Encryption/Decryption time when varying iterations with corresponding NPCR and UACI values, using the image of Lena 256×256, and the key previously used in section 4.

Table 3. Encryption speed of the proposed scheme and some well-known algorithms.

Algorithm	Speed (MB/s)
Proposed Approach	29.785
DES	6.212
AES (192-bit key)	11.472
AES (256-bit key)	10.972

Table 4. Encryption /Decryption time for different iterations values with corresponding NPCR and UACI.

Iterations	Encryption Time (ms)	Decryption Time (ms)	NPCR %	UACI %
1	15	13	99.4325	33.3105
2	31	27	99.6752	33.47010
3	33	31	99.6764	33.46387
4	34	32	99.6758	33.47215
5	47	41	99.6746	33.47102
6	49	45	99.6801	33.47110
7	51	49	99.6808	33.47852
8	63	55	99.6812	33.48012
9	78	61	99.6806	33.48458
10	81	69	99.6813	33.47336
11	83	72	99.6823	33.47025
12	85	79	99.6820	33.47992
13	87	81	99.6819	33.47991
14	91	85	99.6826	33.48256
15	94	91	99.6824	33.47011
16	101	94	99.6831	33.48110
17	109	99	99.6829	33.47854
18	110	102	99.6833	33.47978
19	115	109	99.6829	33.47820
20	125	113	99.6841	33.48120

5. Conclusions

In this paper, an image encryption scheme based on chaotic maps combination is proposed. The system is in a stream-cipher architecture, where the pseudo-random keystream generator is constructed using three chaotic maps, serving the purpose of stream generation and random mixing, respectively. It is found that such a design can enhance the randomness and sensitivity to initial conditions even under finite precision implementation. We stress out that our approach is applicable to almost all chaotic maps with mixing property and that the achievement of the key streams with the desired long cycle length is almost easy. A detailed statistical analysis on both stream generation system and the encryption scheme is given. Experimental results, allow concluding that this algorithm outperforms existing schemes, both in term of speed and security. Having a high throughput, the proposed system is ready to be applied in fast real time encryption applications and suitable for practical use in the secure transmission of confidential information over the Web.

References

[1] Schneier B., *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley and Sons, New York, 1996.

[2] Daemen J. and Rijmen V., *The Design of Rijndael: AES - The Advanced Encryption Standard*, Springer-Verlag New York, Berlin, 2002.

[3] Kocarev L., Jakimoski G., Stojanovski T., and Parlitz U., “From Chaotic Maps to Encryption Schemes,” *in Proceedings of IEEE International Symposium Circuits and Systems*, pp. 514-517, 1998.

[4] Sang T., Wang R., and Yan Y., “Perturbance-Based Algorithm to Expand Cycle Length of Chaotic Key Stream,” *Electronics Letters*, vol. 34, no. 9, pp. 873-874, 1998.

[5] Behnia S., Akhshani A., Ahadpour S., and Mahmodi H., “A Fast Chaotic Encryption Scheme Based on Piecewise Nonlinear Chaotic Maps,” *Physics Letters A*, vol. 366, no. 4-5, pp. 391-396, 2007.

[6] Wong K., Kwok B., and Law W., “A Fast Image Encryption Scheme Based on Chaotic Standard Map,” *Physics Letters A*, vol. 372, no. 15, pp. 2645-2652, 2008.

[7] Kwok H. and Tang W., “A Fast Image Encryption System Based on Chaotic Maps With Finite Precision Representation,” *Chaos, Solitons and Fractals*, vol. 32, no. 4, pp. 1518-1529, 2007.

[8] He X., Zhu Q., and Gu P., “A New Chaos-Based Encryption Method for Color Image,” *Springer Berlin/Heidelberg*, vol. 4062, pp. 671-678, 2006.

[9] Gao T. and Chen Z., “A New Image Encryption Algorithm Based on Hyper-Chaos,” *Physics Letters A*, vol. 372, no. 4, pp. 394-400, 2008.

[10] Behnia S., Akhshani A., Mahmodi H., and Akhavan A., “A Novel Algorithm for Image

Encryption Based on Mixture of Chaotic Maps," *Chaos, Solitons and Fractals*, vol. 35, no. 4, pp. 408-419, 2008.
[11] Sun F., Liu S., Li Z., and Lü Z., "A Novel Image Encryption Scheme Based on Spatial Chaos Map," *Chaos, Solitons and Fractals*, vol. 38, no. 3, pp. 631-640, 2008.
[12] Hu Y., Liao X., Wong K., and Zhou Q., "A True Random Number Generator Based on Mouse Movement and Chaotic Cryptography," *Chaos, Solitons and Fractals*, 2007.
[13] Asim M. and Jeoti V., "An Efficient Hybrid Chaotic Image Encryption Scheme," *New Technologies, Mobility and Security*, pp. 471-480, Springer, Netherlands, 2007.
[14] Chong F., Zhang Z., Chen Y., and Wang W., "An Improved Chaos-Based Image Encryption Scheme," *Lecture Notes in Computer Science*, vol. 4487, pp. 575-582, Springer-Verlag, Berlin, Heidelberg, 2007.
[15] Forré R., "The Hénon Attractor as A Keystream Generator," *In Advances in Cryptology-EuroCrypt'91*, vol. 0547, pp. 76-81, Berlin, Springer-Verlag, 1991.
[16] Pareek N., Patidar V., and Sud K., "Image Encryption Using Chaotic Logistic Map," *Image and Vision Computing*, vol. 24, no. 9, pp. 926-934, 2006.
[17] Shujun L., Xuanqin M., and Yuanlong C., "Pseudo-Random Bit Generator Based on Couple Chaotic Systems and its Applications in Stream-Cipher Cryptography," *in Proceedings of the Second International Conference on Cryptology in India: Progress in Cryptology*, vol. 2247, pp. 316-329, 2001.
[18] Kotulski Z. and Szczepanski J., "On Constructive Approach to Chaotic Pseudorandom Number Generator," *in Proceedings of Regional Conference on Military Communication and Information Systems*, Zegrze, pp. 191-203, 2000.
[19] Kelber K., Götz M., and Schwarz W., "Generation of Chaotic Signals with N-Dimensional Uniform Probability Distribution by Digital Filter Structure," *in Proceedings of the 7th IEEE Digital Signal Processing Workshop (DSPWS'96)*, Norway, Loen, pp. 486-489, 1996.
[20] Marcel A. and Michel D., *The Logistic Map and the Route to Chaos: From the Beginnings to Modern Applications (Understanding Complex Systems)*, Springer, 2006.
[21] Papoulis A., *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, New York 1965.
[22] Williams D., *Weighing the Odds: A Course in Probability and Statistics*, Cambridge University Press, pp. 548, 2001.
[23] Kolmogorov-Smirnov Test Table, http://www. eridlc.com/onlinetextbook/appendix/table7.htm
[24] Santis A., Ferrara A., and Masucci B., Discrete Applied Mathematics , vol. 154, pp. 234, 2006.
[25] Shannon C., "Communication Theory of Secrecy Systems," *Bell System Technical Journal*, vol. 28, pp. 656-715, 1949.
[26] Lian S., JinshengSun., and Wang Z., "Security Analysis of a Chaos-Based Image Encryption Algorithm," Physica A 351, pp. 645-661, 2005.
[27] Chen G., Mao Y., and Charles K., "A Symmetric Image Encryption Scheme Based on 3D Chaotic Cat Maps," *Chaos Solitons and Fractals*, pp. 749-761, 2004.
[28] Vanstone S., Menezes A., and Oorschot P., Handbook of applied cryptography. London: CRC Press, 1996.
[29] Crypto++ Library, http://www.cryptopp.com.

Kamel Faraoun received his Master's degree in computer science from the Computer Science Department of Djilali Liabbes University, Algeria in 2002. His current research areas include computer safety systems, genetic algorithms, fractal images compression evolutionary programming and grammatical inferences, and physical materials structures modeling. He is preparing his PhD thesis in the field of computer security using artificial intelligence systems.