IJCCCE Vol.15, No.3, 2015

New Pseudo-Random Number Generator System Based on Jacobian Elliptic maps and Standard Map

Asst. Prof. Dr. Luma Fayeq Jalil[1] Prof.Dr. Hilal Hadi Saleh[2] Ekhlas Abass Albhrany[3]

[1,2] Department of Computer Science, University of Technology, Baghdad
[3] Department of Computer Science, Mustansiriyah University, Baghdad, Iraq,

e-mail: dr_lumafaik79@yahoo.com, hhsrq888@yahoo.com akhlas_abas@yahoo.com,

Received: 25/2/2015

Accepted: 11/11/2015

Abstract- Chaotic systems have numerous properties, for example: mixing property, sensitivity to initial conditions parameters, structural complexity and deterministic dynamics. These properties were investment in the last decade for cryptographic applications and developments of pseudorandom number/bit generator. The paper propose new pseudorandom number (bits) generator (PRNG) based on the Jacobian elliptic chaotic maps and standard map. The principle of the method consists in generating binary sequence from elliptic chaotic maps of cn and sn types. These sequence positions is permuted using standard map. The performance of the generator is studied through conventional statistical methods and also using the NIST test suite. The results show that the produced sequences possess high randomness statistical properties and good security level which make it suitable for cryptographic applications.

Keywords: chaotic function, pseudorandom sequences, Jacobian elliptic maps, Standard map, NIST test suite.

1. Introduction

The generation of PRNG plays an essential role in a large number of applications such as simulations of numerical, statistical mechanics, gaming industry, communication or cryptography. The main advantages of such generators are the rapidity and the repeatability of the sequences and require less memory for algorithm storage. First way to design such a pseudo-random number generator is connected to the chaos theory [1]-[3]. That theory focuses primarily on the description of these systems that are often very simple to define, but whose dynamics appears to be very confused. Indeed, chaotic system very attractive for pseudo-random number generators because a slightly change in the input can cause a large change in the output (i.e. the extreme sensitivity to the initial conditions). Moreover, during this last decade several pseudo-random number generators have been successfully developed.

Patidar, 2009 designed a new PRBG (pseudo random bit generator) based on running pair of chaotic logistic maps side-by-side and beginning from random independent initial conditions [3]. Pareek , Patidar , Sud, 2010 proposed a new binary sequence generator, called Cross- Coupled Chaotic Random Bit Generator (CCCBG), which achieve the interesting properties of a skew tent map. They used two chaotic maps which are the piecewise linear skew tent maps and cross-coupled. The CCCBG generates the binary sequences based on the comparison between the outputs of the skew tent and cross coupled chaotic maps [5]. Francois, Grosges, 2011 proposed an algorithm for generation of multiple pseudo-random sequences using a chaotic function. The algorithm uses permutations. The permutation positions are computed and indexed using a chaotic function based on linear congruencies. These chaotic permutations are obtained iteratively on this initial vector to produce two chaotic maps [6]. Azeem, Adriana, Adrian, 2013 suggest using tent map to generate pseudorandom binary sequences. The binary sequences under investigation are obtained either by considering all the successive iterations of the tent map and choosing a threshold equal to the tent map parameter or by applying a periodical sampling on the tent map values and by choosing a threshold equal to 0.5 [7]. Recently, Michael François, David Defour and Christophe Negre 2014 propose pseudo-random bit generator based on combined three logistic maps and generate a block of 32 random bits at each iteration. The proposed generator based on the binary64 double that produced based on the IEEE 754-2008 standard for floating-point arithmetic [8].In this paper, we propose a novel pseudo-random number (bit) generator (PRNG) based on the Jacobian elliptic chaotic maps and standard map. It combines the output of the elliptic chaotic maps of cn and sn types. The output from each type is converted to binary sequence. Theses sequences are combined to produce one binary sequence. This sequence is permuted by using standard map. The choice of using Jacobian elliptic chaotic maps and standard map enlarges the complexity of the system and increases the difficulty for an attacker to extract sensitive information from the outputs. Experimental results and security analysis indicate that, the elliptic chaotic map is advantageous

from the point of view of large key space and high level of security.

The produced pseudo-random sequences have successfully passed the various statistical tests. The assets of the generator are: high sensitivity to initial seed values, high level of randomness and good throughput.

The paper is structured as follows, the description of the method as well as the chaotic functions analysis are given in Section 2 and 3 and 4. Section 5 presents the statistical analysis applied on a set of generated pseudo-random sequences. The security analysis of the generator is achieved in Section 6, before conclusions.

2. Jacobian Elliptic Chaotic Map

The core of the generator is the of the cn and sn type of Jacobian elliptic chaotic map. Ergodicity and fixed interval of chaotic orbits are two major properties for the performance of chaos based cryptosystems. The families of one-parameter elliptic chaotic maps of cn and sn at the interval [0, 1] are defined as the ratio of Jacobian elliptic functions of cn and sn types [9] through the following equations:

$$\Phi_N^{(1)}(x,\alpha) = \frac{\alpha^2 (cn(Ncn^{-1}(\sqrt{x})))^2}{1 + (\alpha^2 -))\ \ (cn\left(Ncn^{-1}(\sqrt{x})\right)\Big)^2} \quad (1)$$

$$\Phi_N^{(2)}(x,\alpha) = \frac{\alpha^2 (sn(Nsn^{-1}(\sqrt{x})))^2}{1 + (\alpha^2 - 1)(sn\left(nsn^{-1}(\sqrt{x})\right)\Big)^2} \quad (2)$$

Obviously, these equations map the unit interval [0, 1] into itself. The maps $\Phi_N^{(w)}(\alpha\ ,\ x)$, w = 1, 2, are (N−1)-nodal maps, that is, they have (N−1) critical points in unit interval [0, 1] and they have only a single period one stable fixed point or they are ergodic.

As an example of the Jacobian elliptic maps (1), the following maps can be presented:-

$$\phi_2^{(cn)}(x,\alpha) = \frac{4\alpha^2 x(1-k^2x)(1-x)}{(1-k^2x^{2)^2} + 4(\alpha^2-1)x(1-k^2x)(1-x)} \quad (3)$$

$$\phi_{\ 2}^{(sn)}(x,\alpha) = \frac{\alpha^2((1-k^2)(2x-1)+k^2x^2)^{\ 2}}{((1-k^2+2k^2x-k^2x^2)^{\ 2} + (\alpha^2-1)\big((1-k^2)(2x-1)+k^2x^2\big)^{\ 2})} \quad (4)$$

Where $x_0 \in [0, 1]$, $\alpha \in [0, 4]$ and $k \in [0, 1]$. The parameter k (modulus) represents the parameter of the elliptic functions. Elliptic chaotic maps are ergodic for certain values of their parameters.

3. Standard Map

The so-called standard map was introduced in [10]-[11], and is described by:

$$\begin{cases} a_{i+1} = (a_i + b_i) mod\ 2\pi\ , \\ b_{i+1} = (b_i + K\sin(a_i + b_i)) mod\ 2\pi, \end{cases} \quad (5)$$

where k is the control parameter satisfying k > 0, and the i^{th} states ai and bi both take real values in $[0,2\pi)$ for all i. Thestandard map was discretized in astraightforward manner [12] by substituting x = aN/2 π, y = bN/2 π, K = kN/2 π intoEq. (6), which maps from $[0,2\ \pi) \times [0,2\ \pi)$ to N × N. After discretization, the map becomes

$$\begin{cases} x_{i+1} = (x_i + y_i) mod\ N\ , \\ y_{i+1} = (y_i + K\sin\frac{x_{i+1}N}{2\pi}) mod\ N\ , \end{cases} \quad (6)$$

where K is a positive integer. The properties of this discretized map may not be as good as the original one, but it can be implemented in the integer domain, which reduces the computational complexity and is more suitable for real-time data encryption. The standard map is used to realize data permutation [12].

In the standard map, the pixels at the corners of a square image have some

special properties. For example, the pixel at position (0, 0) remains unchanged after any number of iterations. This is actually a weakness of the permutation process. And it can do some help to the attackers although the permutation process is further strengthened by a diffusion process. In order to avoid it, [13] proposed to change the positions of the pixels at the corners ((0, 0), (0,N-1), (N- 1, 0) and (N- 1,N - 1)). That is, the normal scan order is changed into a random one. A random-couple (rx, ry) is generated (after the iteration of chaotic map), which represents the position of a randomly selected pixel in the square image. Then, the whole image shifts in horizontal and vertical directions by rx and ry, respectively as shown in Figure 1.The normal scan mode by using the random shift process is changed into a random one, so it is named a random-scan mode.

The two parameters rx and ry both vary from 0 to N-1. Thus, the random-scan process can be combined with the chaotic permutation process, and the modified chaotic map becomes

$$\begin{cases} x_{i+1} = (x_i + r_x + y_i + r_y) mod\ N\ , \\ y_{i+1} = (y_i + r_y + K \sin\frac{x_{i+1}N}{2\pi}) mod\ N. \end{cases} \quad (8)$$

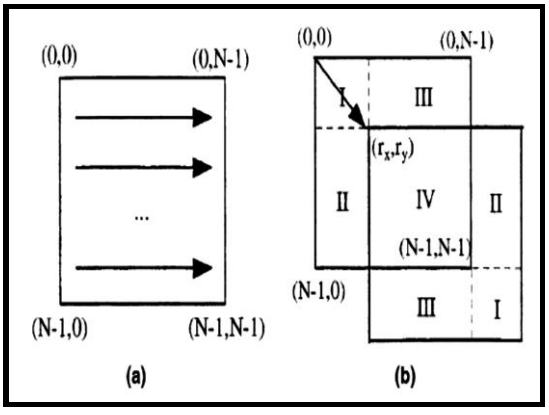

Figure 1 Scan order in a square image:

(a) Normal scan mode, (b) Random scan mode

The modified map is still invertible, so the inverse-permutation process can be easily realized. The modified chaotic confusion process has two advantages [13]:

First the random-couple can be generated under the control of keys, which enlarges the cryptosystems key space. This mean that the key space for the random-couple is 2 × N (N is the width or height of the matrix).

Second the random-scan process makes it difficult to break the diffusion key under known-plaintext attacks. This mean that the random-scan process confuses the position of the first pixel, which makes attackers difficult to get the first pixels cipher-pixel, and thus increases the difficulty of breaking the diffusion key.

4. The Proposed PRNG

The main idea of the proposed PRNG consists of the following major steps:-

- **Step 1**: the initial condition (x) and control parameters (α and k) are input to the Jacobian elliptic chaotic map type cn equation (3) and sn equation (4). These numbers are floating point numbers where the precision is 10^{-16} for each of x_0, α and k, considered as the keys of the generator.
- **Step 2**: Iterate the Jacobian elliptic maps 100 times and ignore the results, in order to eliminate the transient effect of chaotic map.
- **Step 3**: modify the values of x, k and α using simple XOR operation to increase the complexity of algorithm.
- **Step 4**: Iterate the Jacobian elliptic maps one time and convert the floating point output of each map to binary sequence of random length. The two sequences are

combined into one random length binary sequence.

- **Step 5**: The parameters of standard map equation (7) are constructed from the half 24-bit of the resulted sequence. These bits are divided into three 8-bit integer numbers to produce the parameters of standard map (r_1, r_2, k). These 24-bit then is eliminated from the sequence. The resulted binary sequence is translate to matrix of 8 x N where N is an integer number.
- **Step 6**: Diffuse the resulted matrix by using standard map equation (7).
- **Step 7**: the resulted matrix from the standard map is transferred column by column to new binary sequence.
- **Step 8**: repeat from step 4 until the desired number of bits (numbers) is reached. When number of bits becomes a chosen number (in proposed algorithm this number is 500), the parameters x, k and α are modified based on last values of cn and sn maps in order to increase the complexity of detect the keys.
- **Step 9**: The output of the algorithm can be either a binary sequence of random length or a sequence of a random number of integer numbers.

The flowchart of the proposed algorithm is presented in Figure 2.

5. Statistical Analysis

A statistical analysis should be carefully conducted to prove the quality of the pseudo-random sequences. The quality of the output sequences produced by any PRBG must have a high level of randomness and be completely decorrelated from each other.

5.1. Randomness evaluation.

This testing is implemented using statistical tests NIST (National Institute of Standards and Technology of the U.S. Government) [14]. The testing is achieved on sequences produced from nearby or successive seed values.

Because for very distant seed values, the chaotic trajectories are very different, this usually allows obtaining good pseudo-random sequences.

The testing was realized by generating a number of m = 1000 different binary sequences of length 1500. Each sequence is generated by using different seed values. The seed values are distributed in the range [0..1] floating point number. All sequences generated by the proposed PRNG are analyzed using the NIST statistical package. These tests are divided into two groups based on the sequence length.

The first group consists of tests that can be evaluated on the sequences that have length >=100 bits, while the second group consists of tests that are evaluated on the sequences have length >=1000, 000. Therefore we analyzed individual sequences and concatenate sequences by using the first group and only the concatenate sequences using second group. The results of NIST tests obtained on the two groups are presented in Table 1 and Table 2 respectively. The acceptable proportion should lie above the proportion previously defined. For individual sequences, the proportion is:

$$0.99 \pm 3\sqrt{\frac{0.99 \times 0.01}{1000}} = 0.99 \pm 0.0094392.$$

This mean that the proportion should be in the confidence interval [0.9994392.. 0.9805608]. All the tested sequences (individual and concatenate) pass

successfully the NIST tests. These results show clearly the quality of the produced sequences from successive seed values.

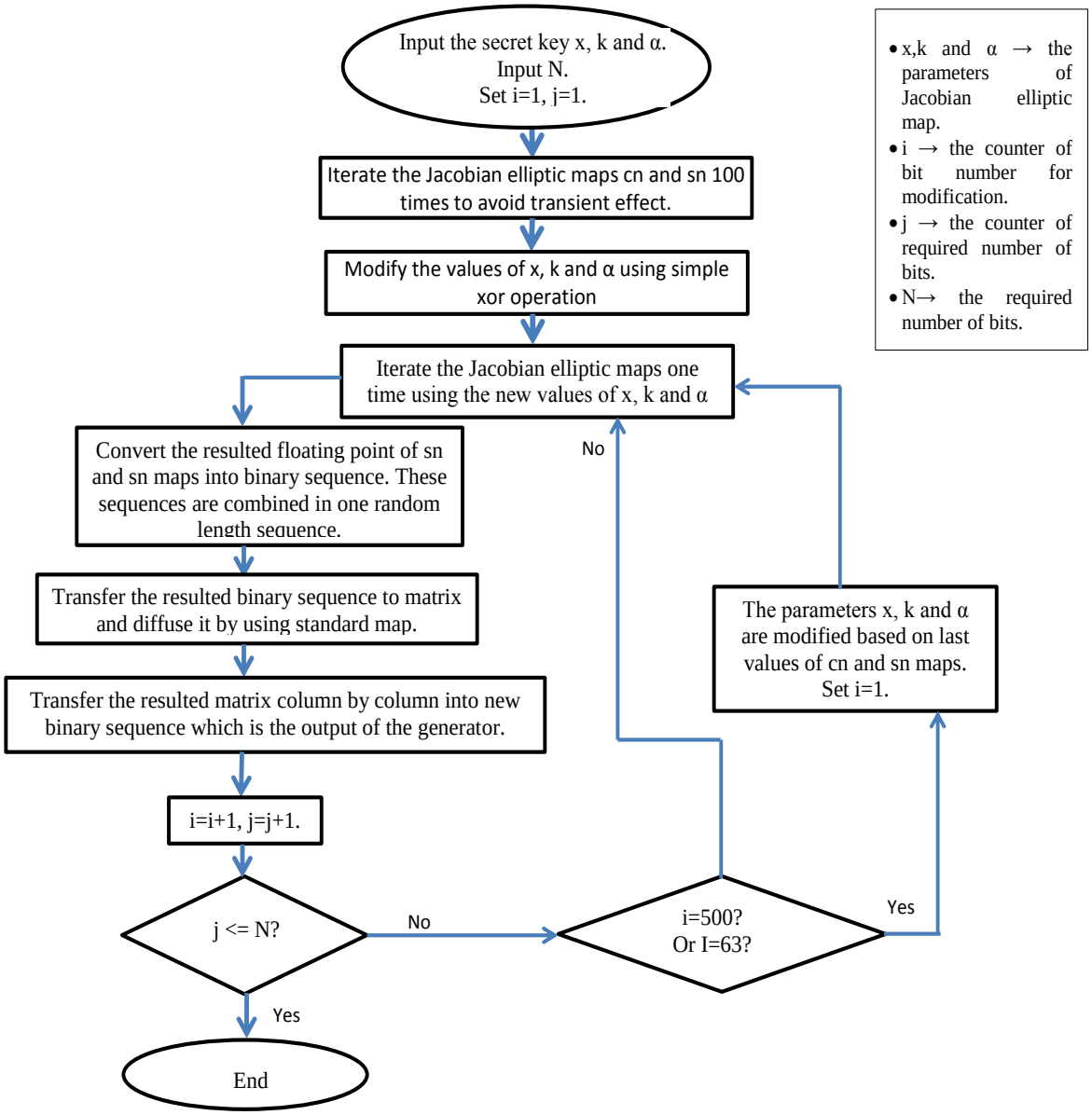

Figure 2 The Flowchart of Proposed PRNG.

5.2. Correlation evaluation

Correlation evaluation is to check the correlation between the produced pseudo-random number sequences. This can be done in two different ways.

Firstly, the correlation between generated sequences is analyzed globally by computing the Pearson's correlation coefficient of each pair of sequences [15].

Secondly correlation based directly on the bits of sequences is analyzed. The Hamming distance between two binary sequences (of the same length M) is the number of places where they differ, i.e., the number of positions where one has a 0 and the other a 1 [16].

5.2.1. Pearson's correlation coefficient

Consider a pair of sequences given by: S1 = [x_1, . . . ,x_N] and S2 = [y_1, . . . , y_N]. Therefore, the corresponding correlation coefficient is[15]:

$$C_{s1,s2} = \frac{\sum_{i=0}^{N-1}(x_i - \bar{x}).(y_i - \bar{y})}{[\sum_{i=0}^{N-1}(x_i - \bar{x})^2]^{\frac{1}{2}}.[\sum_{i=0}^{N-1}(y_i - \bar{y})^2]^{\frac{1}{2}}} \quad (8)$$

Where the mean values of S_1 and S_2 are:

$$\bar{x} = \sum_{i=0}^{N-1} x_i / N \quad \text{and} \quad \bar{y} = \sum_{i=0}^{N-1} y_i / N$$

Two uncorrelated sequences are characterized by $C_{S1,S2}$ =0. The closer the value of $C_{S1,S2}$ is to ±1, the stronger the correlation between the two sequences. In the case of two independent sequences, the value of $C_{S1,S2}$ is equal to 0. Correlation coefficients are computed for each pair of sequences and the distribution of their values is presented by a histogram.

The correlation between each pair of the 1000 produced sequences is computed using Pearson's correlation coefficient. The results of the coefficient are represented in the histograms shown in Figure 3. The histogram shows that the computed coefficients are very close to 0. This means that around 99.9% of the coefficients belong to [-0.08, 0.08] and the correlation between the produced sequences is very small.

5.2.2. Hamming distance

Given two binary sequences S = [s_0,……, s_{M-1}] and $S' = [s'_0$,…… s'_{M-1}] of same length (M), the Hamming distance is the number of positions where they differ. The distance is given as [16]:

$$d(S, \acute{S}) = \sum_{j=0}^{M-1} (s_j \oplus \acute{s}_j). \quad (9)$$

In the case of truly random binary sequences, such distance is typically around M/2, which gives a proportion (i.e. d(S, S′) / M) of about 0.50. For each pair of produced sequences, this proportion is determined and all values are represented through a histogram. The interest of both approaches is to check the correlation for generated sequences mainly from nearby or successive seed values.

The bits of 1000 produced pseudo-random sequences are analyzed using Hamming distance. All resulted values are represented through a histogram shown in Figure 4. The distributions show that all the proportions are around 50% and 99.3% of the coefficients are belonging to [0.465, 0.535]. This testing provides another

indication about the decorrelation between the generated sequences.

Table 1Results of first group of the NIST tests on the 1000 generated sequences for individual and concatenate sequences. The ratio η of p-value concerns individual sequences while the p-value concerns the concatenate sequences.

Test name	**η of Individual**	**Final Result**	**p-value of concatenate**	**Final Result**
Frequency	0.994	Success	0.040903443	Success
Block Frequency	0.988	Success	0.553430376	Success
Runs	0.991	Success	0.880462635	Success
Longest Run	0.993	Success	0.231078877	Success
Rank	0.993	Success	0.823833254	Success
FFT	0.989	Success	0.514489087	Success
Non-Overlapping	0.984	Success	0.398411794	Success
Serial (1)	0.989	Success	0.684240449	Success
Serial (2)	0.993	Success	0.999648182	Success
Cumulative Sums	0.995	Success	0.063395343	Success

Table 2Results of second group of the NIST tests on the 1000 generated sequences for concatenate sequences. The p-value concerns the concatenate sequences.

Test name	**p-value of Concatenate**	**Final Result**
Overlapping	0.957408045388764	Success
Universal	0.745350098931327	Success
Linear Complexity	0.402768848028256	Success
Approximate Entropy	0.110486457868039	Success
Random Excursions (8 p-values)	0.807101754304452	Success
	0.485159643724902	Success
	0.94699932242908	Success
	0.719851271910543	Success
	0.150518027509838	Success
	0.286566773029429	Success
	0.539296753686001	Success
	0.850627304185789	Success
Random E-Variant (18 values)	0.349048077620093	Success
	0.490537427022143	Success
	0.351985540956662	Success
	0.253294182514715	Success
	0.268472090963728	Success
	0.1927496064346 1	Success
	0.467725747869967	Success
	0.843110263994956	Success
	0.759058537369923	Success
	0.080102266851582	Success
	0.128301667190791	Success
	0.759139341678901	Success
	0.832576263792955	Success
	0.704772436102199	Success
	0.866081601939654	Success

	0.726128320880061 0.592148885856078 0.843894535051766	Success Success Success

6. Security analyses

When a new PRNG is proposed, it should always be accompanied by some security analyses. All the critical points of the cryptosystem and cryptographic requirements should be taken into account when the analysis is done [1]. The analyzed points are: the size of the key space, the key sensitivity, randomness quality of the outputs and two basic attacks are evaluated: brute-force attack, differential attack

.

6.1. Key space

Crucial part of each cryptosystem is the key. Keeping in mind the end goal to make brute-force attacks infeasible, PRNG ought to have a large key space. The size of key space that is smaller than 2^{128} is not secure sufficiently. Here, the key space is constructed form the parameters of Jacobian elliptic maps cn and sn (initial value x_0 and control parameters k and α) types which are floating point numbers, where $\mathbf{x_0 \in [0, 1], \alpha \in [0, 4] \text{ and } k \in [0, 1]}$.If the precision is $\mathbf{10^{-16}}$ for each of x_0, α and k, the size of key space for initial conditions and control parameters is $\mathbf{2^{160}((10^{16})^3)}$.

In addition the parameters of the standard map (permutation parameter k and the random scan keys $[r_x,r_y]$) which are integer number, where k, r_x and $r_y \in$ [0..255]. If each parameter has **256** possible keys (2^8), the total number of keys is $\mathbf{(2^8)^3=2^{24}}$. So the total space of keys is $\mathbf{2^{160}+2^{24}}$.

6.2. Key sensitivity.

Sensitivity analysis is the investigation of how the instability in the yield of a model can be distributed to various sources of instability in the model input [17]. An essential factor for the pseudo-random generation is the sensitivity on the key. In other words, a small changing in the starting seeds should cause a large change in the pseudo-random sequences.

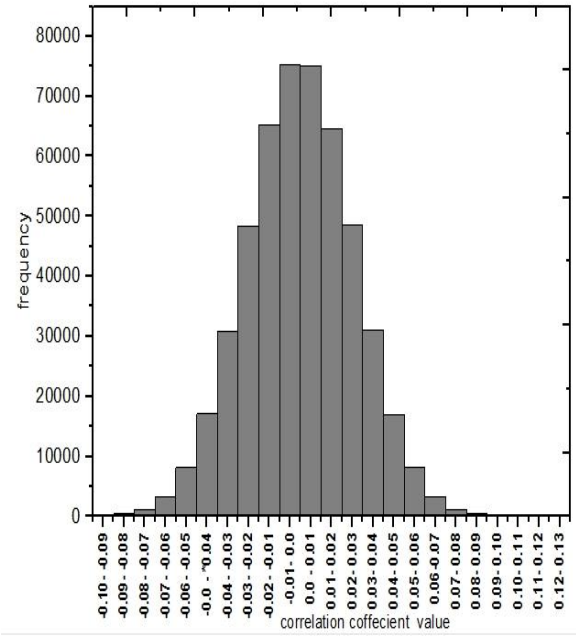

Figure 3 Histogram of Pearson's correlation coefficient values on interval [-0.10, 0.10] for the 1000 sequences

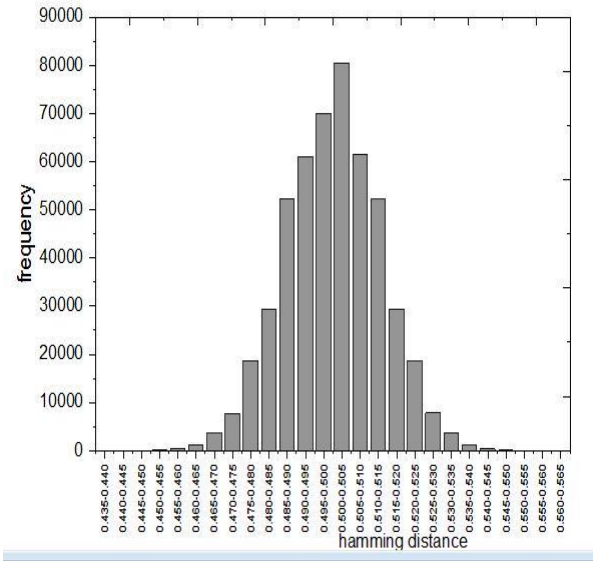

Figure 4 Histogram of Hamming distance on interval [0:435; 0:565] for the 1000 sequences

This means that a small difference on seed values, the output sequences should be completely uncorrelated.

Actually in the test of correlation, the key sensitivity was already tested due to the successive seed values. To ensure the sensitivity of the key, additional analyses have been done using Pearson's correlation coefficients and Hamming distance. Four large pseudo-random sequences of size N =500,000 bits S1, S2, S3, S4 produced from slightly different initial seeds are considered.

A sequence S1 is produced by using the seed values X_0=0.2344587645985498, K_0=0.5123678943210314, α_0=2.8231406754308769.

A sequence S2 is produced by using the seed values $X_1=X_0+8\times10^{-16}$, $K_1=K_0+8\times 10^{-16}$, $\alpha_1=\alpha_0+8\times10^{-16}$.

A sequence S3 is produced by using the seed values $X_2=X_1+8\times10^{-16}$, $K_2=K_1+8\times 10^{-16}$, $\alpha_2=\alpha_1+8\times10^{-16}$.

A sequence S4 is produced by using the seed values $X_3=X_2+8\times10^{-16}$, $K_3=K_2+8\times 10^{-16}$, $\alpha_3=\alpha_2+8\times10^{-16}$.

The analysis is done using the linear correlation coefficient of Pearson and the Hamming distance between the four pseudo-random sequences produced with slightly different seed. The results show that, the sequences are highly correlated from each other as shown in Table 3.

Table 3 Correlation Coefficients between four pseudorandom sequences produced with slightly different seeds

Tests	S1/S2	S1/S3	S1/S4	S2/S3	S2/S4	S3/S4
Pearson Corr. Coef.	-0.00089	-0.00104	-0.00163	0.00166	0.001881	0.001594
Hamming Distance	0.500012	0.50062	0.500818	0.499150	0.499050	0.499200

6.3 Attacks of proposed PRNG

Any new PRNG must be analyzed against attacks to check if the generator cannot be

broken. Here, the resistance of the generator against two basic attacks as the brute-force attack and differential attack is analyzed.

a) Brute-force attack.

The size of the key space must be large enough to prevent a brute-force attack [18]. This attack consists in checking systematically all possible keys until the correct key is found. In the worst case, all the combinations are tested, that necessitates trying all the key space.

When it is not possible to detect any weakness in the algorithm, such an attack might be utilized that would make the task easier. To resist this kind of attack, the size of the key space must be large. It is generally accepted that a key space of size larger than 2^{128} is computationally secure against such attack. In the proposed PRNG, the size of the key space

is around 2^{184} [1], which clearly allows resisting the brute force-attack.

b) Differential Attacks **7**

This attack is similar to the chosen-**8** plaintext attack; its principle is studying how differences in an input can affect the resultant difference at the output in an attempt to derive the key [18].Trying to make a slight change on the input pair, attacker observes the change of the produced sequences. Such technique of cryptanalysis was introduced by Biham and Shamir [19]. Given two inputs In_1 and In_2 to the generator and the corresponding outputs Out_1 and Out_2, there are two methods to find the differences between the two outputs.

Firstly, the difference can be computed between the two pseudo-random sequences relatively to the bits or blocks of bits by subtraction method. This can be done by

$\Delta in = | In1-In2 |$ and $\Delta out = | Out1 - Out2 |$, respectively.

Secondly, the difference between the two output sequences can be computed by

$$\Delta in = In1 \oplus In2$$

and

$$\Delta out = Out1 \oplus Out2$$

Differential probability is then used to measure the diffusion aspect on the initial conditions. In proposed PRNG, we iterate the jacobain maps 100 times before the beginning of the generation. In addition, the results of the analyses showed that even with a slight difference on the seeds, the produced outputs are almost uncorrelated from each other. So, the proposed PRNG is designed to avoid this kind of cryptanalysis.

7. Comparative Results of proposed PRNG

The proposed CRNG is compared with the PRNG proposed by et al [8]. The deference between the proposed PRNG and the M. François PRNG is made on numbers of factors which include key space, number of bits in each iteration, the result of NIST test and the result of correlation coefficients. Table 4 shows the result of comparison.

The key space of proposed PRNG is larger than that of M. François PRNG. It is by and large acknowledged that a key space

of size bigger than 2^{128} is computationally secure against brute-force attack [8].

The number of bits in each iteration of proposed PRNG is randomly in the rang [64..80] bits in each iteration. This means that the proposed PRNG is faster than M. François PRNG.

8. Conclusions

In this paper, a novel pseudo-random number generator based on the Jacobian elliptic chaotic map types sn,cn and standard map. The initial condition x_0 and the control parameters k and α is the input to the Jacobian elliptic chaotic map to produce binary sequence. Chaotic standard map is used to diffuse the binary sequence. Such a generator has shown its ability to produce a very large number of pseudo-random sequences which can be useful in several cryptographic applications because it has many properties which are the adaptive size of the key space, the sensitivity to the initial inputs (keys), the quality of pseudo-random sequences, the security level against several attacks.

It can be used as pseudo random bit generator or as pseudo random number generator.

The quality of the output sequences randomness is evaluated through NIST tests which are 15 tests. Table 4 shows that the output sequence of proposed PRNG is random with large values for the tests than that for M. François PRNG.

Table 4 Comparison between the proposed PRNG and M. Francois PRNG

	key space	number of bits at each iteration	Freq.	Block Freq.	Runs	Longest Run	Rank	FFT	Non-Over-lapping	Serial (1)	Serial (2)	Cumu-lative Sums
proposed PRNG	$2^{160}+2^{24}$	Randoml y in the range [64..80] bits.	0.994	0.988	0.991	0.993	0.993	0.989	0.984	0.989	0.993	0.957
M. François PRNG	2^{147}	32 bits.	0.991	0.991	0.989	0.989	0.988	0.987	0.993	0.989	0.990	0.991

References

[1] G. Alvarez, S. Li, "Some Basic Cryptographic Requirements for Chaos-Based Cryptosystems", International Journal of Bifurcation and Chaos, World Scientific, Vol. 16, 2006, pp 2129-2151.

[2] F. Zheng, X. Tian, J. Song and X. Li, "Pseudo-random sequence generator based on the generalized Henonmap", The Journal of China Universities of Posts and Telecommunications, vol. 15, no. 3, 2008, pp. 64–68.

[3] V. Patidar and K.K. Sud, "A pseudo random bit generator based on chaotic logistic map and its statistical testing", Informatics, 33, 2009, pp. 441–452.

[4] N. Pareek, V. Patidar, and K. Sud, "A Random Bit Generator Using Chaotic Maps", International Journal of Network Security, vol.10, No.1, 2010, pp. 32-38.

[5] C. E. Shannon. "Communication theory of secrecy systems". Bell System Technical Journal, July, 1948, p.623.

[6] M. Francois, T. Grosges, D. Barchiesi and R. Erra, "A New Pseudo-Random Number Generator Based on Two Chaotic Maps", INFORMATICA, vol. 24, no. 2, 2013, pp. 181–197 Vilnius University.

[7] Ilyas, A. Vlad and A. Luca, "Statistical Analysis of Pseudo Random Binary SequencesGenerated By Using TentMap", U.P.B. Sci. Bull., Series A, vol. 75, no. 3, 2013.

[8] M. François, D. Defour and C. Negre, "A Fast Chaos-Based Pseudo-RandomBit Generator Using Binary64 Floating-Point Arithmetic", Informatica vol. 38, 2014, pp. 115–124.

[9] M. A. Jafarizadeh and S.Behnia, "Hierarchy of one- and many-parameter families of elliptic chaotic maps of cn and sn types", Physics Letters, vol. 310, April 2003, pp. 168–176.

[10] E. A. Jackson, *Perspectives in nonlinear dynamics*, Cambridge University Press, vol. 1, Reprint Edition, 1991.

[11] F. Rannou, "Numerical study of discrete plane area-preserving map", Astron & Astrophys: vol. 31, 1974, pp. 289–301.

[12] J. Fridrich, "Symmetric ciphers based on two-dimensional chaotic maps", International Journal Bifurcation Chaos, vol. 8, no. 6, June 1998, pp.1259-1284.

[13] J. Soto and L. Bassham, "Randomness Testing of the Advanced Encryption Standard Finalist Candidates", Computer Security Division National Institute of Standards and Technology, March 2000.

[14] Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray and S. Vo, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications", NIST Special Publication Revision 1a,vol. 800, April 2010, pp. 131.

[15] V. Patidar, N. K. Pareek, G. Purohit and K. K. Sud, "A robust and secure chaotic standard map based pseudorandom permutation-substitution scheme for image encryption", Optics Communications, vol. 284, no. 19, September 2011, pp. 4331- 4339.

[16] W. Janke, "Pseudo random numbers: generation and quality checks", Quantum Simulations of Complex Many-Body Systems, vol. 10, 2002, pp. 447–458.

[17] J.C. Ascough, , T.R. Green, L. Ma and L.R. Ahjua, "Key Criteria and Selection of Sensitivity Analysis Methods Applied to Natural Resource Models", Research Get, pp. 2463 - 2469.

[18] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd Edition, John Wiley & Sons; 1996.

[19] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," Journal of Cryptology, vol. 4, no. 1,January 1991, pp. 3–72.