Mouse Movement with 3D Chaotic Logistic Maps to Generate Random Numbers

Alaa Kadhim F. [1] and Hakeem Emad M. [2]

[1] Computer Science Department- University of Technology- Iraq
[2] Accounting Department- Kut Technical Institute- Middle Technical University- Iraq
[2]hakeem_emad@yahoo.com

Received: 26 November 2016 **Accepted: 9 April 2017**

Abstract

Since last three decades there are close relationships between chaotic theory and cryptographic theory. Chaotic system behaviors like; highly sensitive to initial states, mix up attribute, deterministic nature and also cannot predict the long term returns, these characteristics help the researchers to enhance security of a cryptography systems, therefore growing number of random numbers generators based on chaotic have been proposed. These proposed generators suffer from limited key space and those based on 1D chaotic map have limited entropy generation capability due to their finite number of Lyapunov exponent(s). In this paper, we propose a random binary sequences generator that produces sequence of bits. General structure of proposed model consists of two parts, first part is mouse device as the nondeterministic source and second part is 3D chaotic system with the coordinates of mouse cursor when movement as the initial seeds, and combines the produced values in algorithmic process. The coordinates of mouse cursor are treated as initial random number with post processing with 3D chaotic maps to increase the randomness and security of the keys. The proposed work has high key space and very long period. Also make obvious that the generated keys possess successful statistical characteristics which is expected of true random binary sequences that are suitable to use in critical cryptography systems, these made by evaluating the results by hardness of 16 tests of NIST(National Institute of Standards and Technology).

Keywords: Chaotic system, mouse device, cryptography, RNG, NIST. Lyapunov exponent.

حركة الماوس مع نظام الفوضى اللوجستي ثلاثي الابعاد لانتاج الارقام العشوائية

علاء كاظم فرحان[1] و حكيم عماد محيبس[2]

[1]قسم علوم الحاسوب - الجامعة التكنولوجية- العراق.
[2]قسم المحاسبة - المعهد التقني كوت - الجامعة التقنية الوسطى - العراق.

الخلاصة

منذ العقود الثلاثة الماضية، هناك علاقات وثيقة بين نظرية الفوضى ونظرية التشفير. سلوكيات نظام الفوضى مثل. الحساسية للحالة البدائية، خصائصها المختلطة، والطبيعة الحتمية وكذلك انه لا يمكن التنبؤ بالنتائج على المدى الطويل. هذه الخصائص تساعد الباحثين لتعزيز أمن أنظمة التشفير، وبالتالي هنالك تزايد في عدد المقترحات لمولدات الارقام العشوائية التي تعتمد على اساس نظرية الفوضى. هذه المولدات المقترحة تعاني من محدودية مساحة المفاتيح، والتي تعتمد على نظام الفوضى ذي البعد الواحد لها قدرة توليد محدودة نظرا للاعداد المحدودة في Lyapunov exponent. في هذه الورقة، تم اقتراح مولد ارقام ثنائية عشوائية والتي تنتج سلسلة من البتات. الهيكل العام للنموذج المقترح يتكون من جزئين، الجزء الأول هو جهاز الماوس كمصدر للارقام غير الحتمية والجزء الثاني هو نظام الفوضى ثلاثي الابعاد حيث يستقبل احداثيات الماوس عند حركته كارقام اولية، ويتم الجمع بين القيم التي تنتج في عملية معينة لتوليد سلسة واحدة . يتم التعامل مع إحداثيات مؤشر الماوس كرقم عشوائي اولي للنظام الفوضى الثلاثي الابعاد لزيادة العشوائية والامنية للمفاتيح المنتجة. الارقام العشوائية المنتجة لها دورة طويلة و وصعبة التكرار وتمتلك خصائص احصائية ناجحة مثل الخصائص المتوقعة لسلسة الارقام الثنائية العشوائية الحقيقية. هذه النتائج تكون مناسبة للاستخدام في أنظمة التشفير الحرجة. قيمت هذه النتائج باستخدام الاختبارات ال 16 لل،NIST (المعهد الوطني للمعايير والتكنولوجيا).

الكلمات المفتاحية: نظام الفوضى ,جهاز الماوس ,التشفير، اس Lyapunov، RNG، NIST.

Introduction

Random Numbers Generators (RNGs) are beneficial for a large number of systems including cryptography applications, stochastic modeling, simulation, and online gaming and lotteries [1]. To propose a successful RNG, we should keep in mind it possesses successful characteristics of randomness, it should be reliable, have minimum cost, unpredictable, complexity of the system, fast produced and so on. Usually, two techniques are available to generate random numbers: first technique depends on deterministic method processed using specific programs, the random numbers being generated from a specific singular input

number, such generator called PRNG, commonly they have higher speed and are better to produce large quantities of numbers. Second generator counts on high entropy source, with specific post-processing processes to these counted sources to generate random numbers, these can be from nondeterministic and stochastic physical events [2], or from deterministic but chaotic [3]. TRNG is commonly based on all types of physical events, these can be thermal noise [4], atmospheric noise [5], radioactive decay [6] and flip a coin. They are considered to generate random sequence with higher security. However, when running on PC platforms which are most popular with common consumers, these TRNGs demands specific expensive equipments which in return be unprofitable for computers users.Since last three decades, researchers have shown that an interesting relationship between cryptography and chaos exists [7]. The chaotic system behaviors like; highly sensitive to initial states, mix up attribute, deterministic nature and also cannot predict the long term returns, these characteristics help the researchers to enhance security of a cryptography systems. So diversities in the initial condition for chaotic system are important for performance of the system to produce highly randomize numbers that have very long period and key space.In this paper, we develop a web application to do an alternate method of seeding chaotic maps, generally consist of two components, the coordinates of mouse device and 3D chaotic maps to produce multiple pseudo-random sequences. The coordinates of mouse cursor are collected while the user moves his/her mouse over the screen for a while to produce a predefined number of values. Repetition of same values has the disadvantage that the same sequences will always be generated from the same seed [8]. To increase the security, manipulation is done for these coordinates, repeating same coordinate is not allowed and neglected. And when these numbers are enough to seed the 3D chaotic maps system then generator will be ready to generate random sequences of binary bits which can be used in the cryptography application. The proposed work has high key space and very max period. We make obvious that the generated keys possess efficient statistical characteristics which is expected of true random binary sequences that are suitable to use in critical cryptography systems. These made by evaluating the results by hardness of 16 tests of NIST. The results successfully passed these 16 tests.

Random Number Generator Using Chaos Theory

Chaos theory is an area of mathematical science that studies the dynamic systems behavior, as we state earlier it is a good choice to use in designing of random number generator [9]. In fact, chaotic system behaviors like; highly sensitive to initial states, mix up attribute, deterministic nature and also cannot predict the long term returns [10], so these properties are valuable in cryptography.Logistic equation of chaos which is explained below can be used to generate large pseudo random numbers. To increase the randomness of produced keys and make the keys more secure it is preferable to use several logistic equations during the generator design stage.

1. Chaos Logistic Equation

It is a dynamic system second order deference equation; the standard from of the logistic equation is given by:

$$F(Y_i) = R\, Y_i\, (1 - Y_i) \qquad (1)$$

Where Y_i is called the iteration of Y_0 (or population) and should by in subinterval [0, 1], and R is the growth rate of a population that takes any values between [1, 4].

Figure 2(a) shows the logistic equation F(Y) with four values of R, this diagram shows how the system is sensitive to initial conditions. The states of the equation are bound between (0, 1) and the equation is symmetric about the center value. With the iteration nature of the system, it enters the chaotic and complex behaviors at about R > 3.56994, and then the trajectory never repeat itself.Figure 2(b), shows that how the system is sensitive to initial conditions for R = 4.0. It illustrates how tow trajectories as the initial conditions to the system have began at almost same values and eventually take different paths with respect to the time. Obviously seems that they have no relation between them. Figure 2(c), we used the bifurcation diagram to show how the dynamic system behaves. It obvious that the states of the mapping is bounded in the region between (0, 1) at R=4.0.This plot illustrates the dynamic changes in the logistic equation with respect to growth rate R (from 2.6 to 4.0 only). From this plot, if R is almost in the range between (2, 3) the system will have one period of behavior (steady state) at different points. If R is almost in the region between (3, 3.5) the system will have two periods of behaviors at different points. If R is almost in the range of (3.5, 3.7) the

system will have 4 periods of behaviors, and so on. Increasing the periodicity of the system when the R is close to 4 and the system behavior will be unpredictable and looks random [11]. Figure 2(d), the quantative measurement of the dynamic system is computed using Lyapunov exponent, which tells if the system is chaos or not depending on the result of it. The positive value of Lyapunov exponent represents the chaotic system with respect to the growth rate of the system R, This figure shows that, the positive values of L when the growth rate R > 3.6. The negative value of Lyapunov exponent refers to non chaotic system(almost when R < 3.6).

$$L = \lim_{n\to\infty} \sum_{i=1}^{n} \ln |f(Y)| \qquad (2)$$

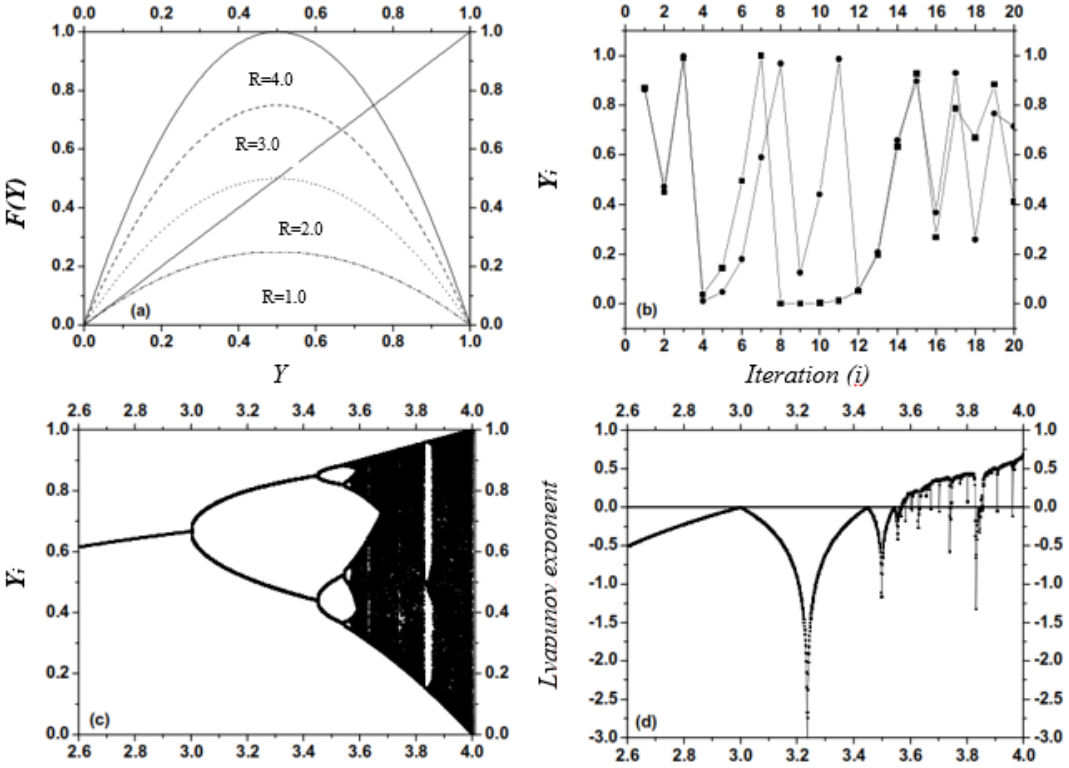


Figure 2: Logistic equation behavior: (a) mapping equation no.(1) with four values of R, (b) sensitivity behavior when R = 4.0, (c) the bifurcation diagram to show how the system behaves with respect to growth rate R. (d) the quantative measurement of the dynamic system is computed using Lyapunov exponent with respect to growth rate R.

2. 3D Logistic Map

The logistic map which is given by equation (1) is one dimension 1D, which depends only on one state variable (Y). The 2D extension of logistic map has been proposed by Hongjuan Liu. et al, which is given by the following two equations:

$$Y_{n+1} = R_1 Y_n (1 - Y_n) + Q_1 Z_n^2 \quad (2)$$

$$Z_{n+1} = R_2 Z_n (1 - Z_n) + Q_2 (Y_n^2 + Y_n Z_n) \quad (3)$$

These two equations depend on two states variables (Y) and (Z). The equations increase the complexity of the system and hence increase the security level. The parameters of these equations control the system behaviors. The two state variables (Y) and (Z) should lie between [0, 1]. The system enters in chaotic when the control parameters have the values such that; $2.75 < R_1 < 3.4$, $2.7 < R_2 < 3.45$, $0.15 < Q_1 < 0.21$, and $0.13 < Q_2 < 0.15$ [12]. The same idea is applying to extend the 2D logistic equations to 3D logistic equations with three basic variables (X), (Y) and (Z) and three control parameters R, Q and k.

$$X_{n+1} = R X_n (1-X_n) + Q Y_n^2 X_i + K Z_n^3 \quad (4)$$

$$Y_{n+1} = R Y_n (1-Y_n) + Q Z_n^2 Y_i + K X_n^3 \quad (5)$$

$$Z_{n+1} = R Z_n (1-Z_n) + Q X_n^2 Z_n + K Y_n^3 \quad (6)$$

The dynamic system enters the chaotic when the control parameters have the following values $3.53 < R < 3.81$, $0 < Q < 0.022$, $0 < K < 0.015$, the three basic state variables (X), (Y) and (Z) should lie between [0, 1][13].

Mouse Movement

As we stated earlier, a true random number can be generated by various methods of nondeterministic source, for example mechanical and electronic noise. In most cases, those methods require additional devices, which make their applications not universal. On the other hand a mouse device is used for pointing an object over the computer screen and controlled by user hand. Any motion of a mouse device on the screen has two dimensions, X-axis and Y-

axis corresponding to the coordinates of the mouse cursor on a surface at the specific point. The movement of the mouse cursor is interpreted into the digital movement. This is done by available application programming interface (APIs). Therefore at any position on the screen, mouse cursor has a unique X-axis and Y-axis values [14]. When the user move the mouse over the screen these values can be collected and processed for further works. Different numbers are generated each time when the user attempts to move the mouse cursor over the web page. It is difficult for the user to repeat the same pattern of moving. These generated numbers can be considered as a sequence of random numbers. Therefore, there is no need to buy additional device to generate random numbers. Also it is inexpensive, suitable and useful technique to produce numbers. It sounds good to use such a advice as a source of random numbers. Before seeding the 3D chaotic maps, these numbers are processing as we explain later.

Proposed model

In the proposed work, it has been developed a web application to do an alternate method that seeded chaotic maps. General structure of proposed method consists of two components, mouse movement and 3D chaotic system. See figure (3).

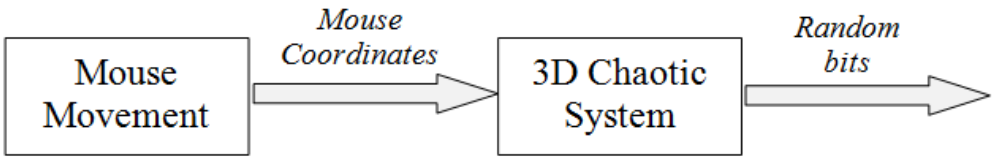


Figure (3): General structure of the proposed random bits generator

The first part of the proposed random bits generator which related to a mouse device requires the computer user moves his/her mouse cursor for a while to generate enough coordinates (X-axis and Y-axis) and then using these coordinates as the initial conditions to seed the chaotic generator.These coordinates are collected while cursor is moving over the screen of the web page to produce two sequences of different values with same length (m), first sequence for X-axis(called X_m) and second sequence for Y-axis(called Y_m). This done using JavaScript methods. To increase the security, repeating same coordinate is not allowed and neglected.

Since these sequences will be the initial conditions to the 3D chaotic maps, the slightest variation of these values will eventually diverge completely. These sequences are multiplied by some fraction to be numbers between [0, 1].

First sequence is for X coefficient for chaotic map as follow

$$X=X_0, X_1, X_2 \ldots\ldots X_n.$$

Second sequence is for Y coefficient for chaotic map as follow

$$Y=Y_0, Y_1, Y_2 \ldots\ldots\ldots Y_n.$$

3D logistic maps require 3 coefficients, say X, Y and Z. And mouse movement generates two sequences only. To get the third sequence Z_m, proposed next formula is used. By subtraction each number in each sequence and multiply the results by the current data and time (T_{curr}). Current date and time is employed in ANSI X9.17(American National Standards Institute)[15], which is updated on each number generation.

$$Z_m = (X_m - Y_m) * T_{curr}. \qquad (7)$$

The third sequence is multiplied by some fraction to be in interval [0,1]. Therefore See figure (4).

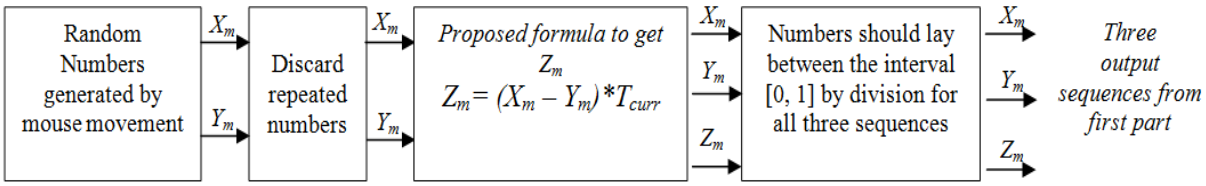


Figure (4): The operating principle of the first part

These three output sequences will be the initial conditions for the second part of the system (3D chaotic system). Such that the values of X_0,Y_0 and Z_0 will be the initial conditions to the 3D chaotic equations of the system to generate 256 binary bits (now m=0).

The next X_1,Y_1 and Z_1 will be the second initial conditions to the 3D Chaotic equations of the system to produce another 256 binary bits (now m=1) and so on for each m. To enter in chaotic behavior (increase the complexity and randomness of generated numbers) which are the desirable properties in cryptography systems, the proposed method ignores first 1000 iterations of each m. After 1000 iterations (proposed generator suggests more than thousand iteration for each m) select 256 numbers from each generated sequences (X_m,Y_m and Z_m) and comparing these three sequences in algorithmic way to get the two binary sequences(say X and Y). For more explanation let's take these steps of our generator processes;

Step1: Generate two sequences X_m and Y_m while moving the mouse devise by a user for a specific time.

Step2: Discard the repeated same numbers

```
Function ReadMouseMovement
{  Xm,new= new X-axis
 Ym,new= new Y-axis
 If (Xm,new not equal Xm,old) and  (Ym,new not equal Ym,old)
    then
       m++;
       Xm,new =   new X-axis;
       Ym,new =   new Y-axis;
       Xm,old  = Xm,new ;
      Ym,old  = Ym,new ;
   End if
}
```

Step3: A third sequence Z_m is obtained from equation (7).

Step4: Numbers in each three sequences are multiplied by some fraction to be values between [0, 1].

Step5: X_m, Y_m and Z_m sequences are the initial conditions to equation numbers (4), (5) and (6) respectively. First 1000 iterations are ignored, at least 1256 Iterations for each m.

Step6: To generate 256 binary bits, Select randomly only 256 values out of each of values from each three sequences that generated as the output from 3D chaotic system. Say new X_i, Y_i and Z_i. Where (i=1, 2, 3…256).

Step7: for each i, generate 256 binary bits (R1) from decimal values by comparing the values of new X_i and Y_i, using following threshold:

$$g(X_{i+1}, Y_{i+1}) = \begin{cases} 1 & \text{if } X_{i+1} > Y_{i+1} \\ 0 & \text{if } X_{i+1} \leq Y_{i+1} \end{cases}$$

And also generate second binary bits (R2) from decimal values by comparing the values of new Y_i and Z_i using this threshold:

$$g(Y_{i+1}, Z_{i+1}) = \begin{cases} 1 & \text{if } Y_{i+1} > Z_{i+1} \\ 0 & \text{if } Y_{i+1} \leq Z_{i+1} \end{cases}$$

Step8: for each i, the two new 256 bit sequences of binary R1 and R2 are XORing together in order to generate the final 256 bit as the output of our generator.

$$R_{out,\,i} = R1_i \oplus R2_i. \qquad (8)$$

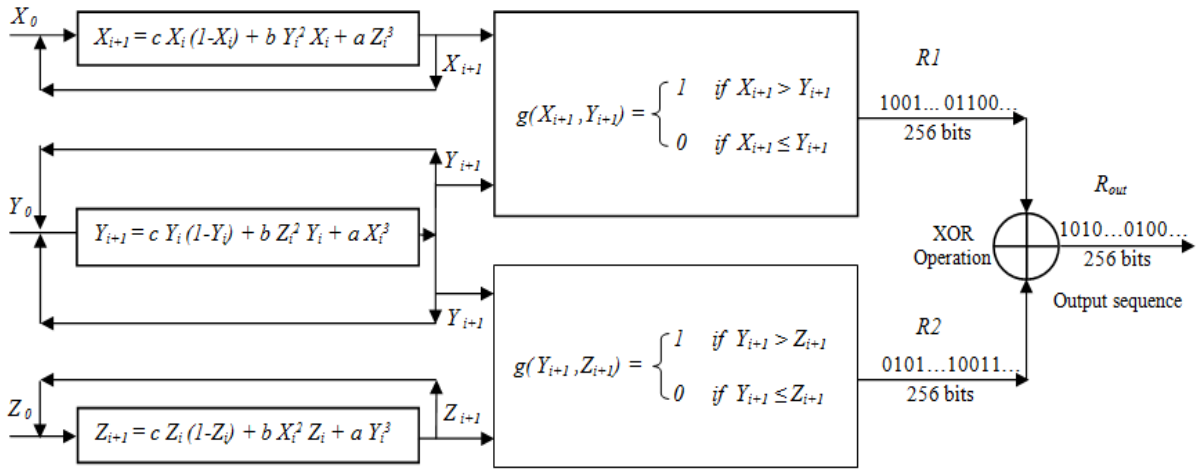
For example, if the values of X_{10}, Y_{10} and Z_{10} will be 0.271, 0.146 and 0.345 respectively, then these initial condition will enter the chaotic equations and produce 3 sequences of different numbers such that:

X_{10}= 0.6852283181271831, 0.6404263931576534… $X_{10,n}$

Y_{10}= 0.35499615224169578, 0.8363916781438735…$Y_{10,n}$

Z_{10}= 0.26059620634779834, 0.25497013155484266… $Z_{10,n}$

To increase the complexity and generate one output sequence of bits; these sequences are combine by comparing the outputs of the $X_{10,i}$ and $Y_{10,i}$ to produce first 256 binary bits sequence and comparing the $Y_{10,i}$ and $Z_{10,i}$ to produce second 256_{10} binary bits sequence and these two bits sequences are XORing together in order to produce 256 binary bits ($R_{out,10}$)output of the system. The general description of the proposed method is shown in the figure (5).



At the end we will have a huge numbers of binary keys that can be used in cryptography systems.

Statistical testing

To be confidence, that the proposed generator can produce random binary bits that should be cryptographically secure, they should be tested by specific tests that are dedicated to check whether the generated sequences of binary bits are expected to be like the truly random sequence or not For analyzing these sequences, two most common testing methods available

to do so; the NIST tests suite [16] and the DIEHARD test suite [17]. Each suite has different types of statistical test. Each type tests specific characteristic of random numbers. Proposed binary bits are tested using five basic tests of randomness and also using the NIST tests suite which is the most strength tests.

1. Five Basic test

Each following statistical test examines some characteristic of generated random binary bits for a specific pattern or set of patterns. These five tests examine 50 samples, each of length 10^6

1.1 Frequency test

Pass value 0.144 with freedom degree " 1 " must be <= 3.84

1.2 Run test

Pass value t0 = 6.454 with freedom degree " 5 " must be <= 10.788
Pass value t1 = 2.254 with freedom degree " 5 " must be <= 10.788

1.3 Poker test

Pass value 2.464 with freedom degree " 5 " must be <= 11.1

1.4 Serial test

Pass value 2.496 with freedom degree " 3 " must be <= 7.81

1.5 Auto_correlation test

Shift no. 1 >--> pass value 3.028

Shift no. 2 >--> pass value 0.786

Shift no. 3 >--> pass value 0.531

Shift no. 4 >--> pass value 2.313

Shift no. 5 >--> pass value 0.443

Shift no. 6 >--> pass value 0.487

Shift no. 7 >--> pass value 0.291

Shift no. 8 >--> pass value 0.065

Shift no. 9 >--> pass value 0.001

Shift no. 10 >--> pass value 2.525

with freedom degree " 1 " must be <= 3.84

2. National Institute of Standards and Technology (NIST) Suite

This suite consists of 16 tests designed to evaluate the criteria of binary sequences for randomness. These binary sequences are generated whether by soft programs or by dedicated special devices. As state before, each test concentrates on the non random binary bits that may find out in binary sequences. The 16 tests are classified into two classes; nonparameterized and parameterized tests [18].

Strategy of the Test

The NIST test suite is based on hypothesis testing. It tests whether the specific binary sequence is truly random binary sequence or not, such a hypothesis is named as null hypothesis and denoted by H_0. If the H_0 is accepted in specific tests then the selected binary sequences is passed that test of randomness, otherwise it rejected. In addition, probability values (P-value) is computed for each test under the H_0 and comparison to this (P-value) is done with a significance level ($\alpha = 0.01$) to check whether the sequence is passed or failed.

If the P-value $\geq \alpha$ then the sequence is successfully passed the statistical test and the H_0 is accepted. And if the P-value $< \alpha$ then the sequence is fail the statistical test and the H_0 is rejected. Usually, the value of (α) is selected between [0.001, 0.01]. If the value of $\alpha = 0.01$, that means only 1% of the sequences are fail [16].

Results interpretation

For testing purposes, in this case, generate 100 (m=100) sequences of binary using the proposed generator. Each sample length is 10^6 bits. We select the value of $\alpha = 0.01$ and probability values (P-value) related to each sequence is computed for each test. So the total number of P-value we calculate for all samples is 5000. The results of nonparameterized tests of NIST suite is shown in table (1), and the rest results corresponding to parameterized tests of NIST suite shown in table (2) . It is clear that our generated binary sequences are passing all the 16 tests of NIST.

Table 1: Results of Nonparameterized

#	Names of the Tests	Status
1.	Frequency	Success
2.	Longest runs of ones	Success
3.	Runs test	Success
4.	Lempel-Ziv compression	Success
5.	Discrete Fourier transform	Success
6.	Cumulative sums(reverse and forward)	Success
7.	Binary matrix rank	Success
8.	Random excursions	Success
9.	Random excursions variant	Success

Table 2: Results of Parameterized

#	Names of the Tests	Status
1.	Frequency within a block (Block length = 128)	Success
2.	Approximate entropy (Block length = 10)	Success
3.	Linear complexity (Block length =500)	Success
4.	Maurer's (Universal Statistical) (selected blocks = 7, length of the blocks = 1280)	Success
5.	Serial (length of the blocks = 16)	Success
6.	Overlapping template matching (Template =111111111)	Success
7.	Non-overlapping template matching (Template length=9)	Success

Conclusion

A new design of random number generator has been proposed. Repeated keys problems are solved in proposed generator. Mouse device is used as a non deterministic source of initial random numbers which is inexpensive, convenient and universal device. Also we mentioned that it is impossible to reproduce the same numbers with the same user. The second 3D chaotic system part has been used to increase the complexity and randomness of the generated keys. Tacking the advantages of chaotic behavior and simplicity of using mouse is the motivation of this proposed method. Eventually a huge numbers of random binary sequences that have maximum periods are generated. These binary sequences are tested. The results of these tests insured that the acceptable characteristics of generated binary sequences to be random binary sequences and hence can be used efficiently in designing of cryptography systems.

References

1. N. Ferguson, B. Schneier, and T. Kohno, "Cryptography Engineering: Design Principles and Practical Applications",Wiley, 2010.
2. C. R. S. Williams, J. C. Salevan, X. Li, R. Roy, and T. E. Murphy, "Fast Physical Random Number Generator Using Amplified Spontaneous Emission," Optics Express, vol. 18, pp. 23584–23597, November 2010.
3. K. Hirano, K. Amano, A. Uchida, S. Naito, M. Inoue, S. Yoshimori, K. Yoshimura, and P. Davis, "Characteristics Of Fast Physical Random Bit Generation Using Chaotic Semiconductor Lasers," IEEE J. Quant. Electron., vol. 45, no. 11, pp. 1367 – 1379, 2009.
4. M. Bucci, L. Germani, R. Luzzi, A. Trifiletti, and M. Varanonuovo, "A High Speed Random Number Source For Cryptographic Applications On A Smartcard", IEEE Transaction on Computer, 52(4), pp. 403–409, 2003.
5. W. T. Holman, J. A. Connelly, and A. B. Downlatabadi, "An Integrated Analog/Digital Random Noise Source", IEEE Transaction on Circuits and System I, 44(6), pp. 521–528, 1997.
6. J. Walker, HotBits: "Genuine Random Numbers Generated by Radioactive Decay", http://www.fourmilab.ch/hotbits, 2002.
7. G. M. Bernstein and M. A. Lieberman, "Secure Random Number Generation Using Chaotic Circuits," IEEE Trans. Circuits Syst., vol.37, no.9, pp.1157– 1164, 1990.
8. Wichmann B. and Hill I. "Generating Good Pseudorandom Numbers", Computational Statistics & Data Analysis, 51 (3): 1614 -1622, 2006.
9. M. François, T. Grosges, D. Barchiesi and R. Erra, "A New Pseudo-Random Number Generator Based On Two Chaotic Maps". Informatica, Vol. 24, No. 2, pp. 181–197, 2013.
10. J.M. Bahi, C. Guyeux and Q.Wang. "A Pseudo Random Numbers Generator Based On Chaotic Iterations. Application to watermarking", International Conference on Web Information Systems and Mining, Vol. 6318 of LNCS, pp. 202–211, Sanya, China, October 2010.
11. *Strogatz and Steven, "Nonlinear Dynamics and Chaos. Perseus Publishing", 2000.*

12. Hongjuan Liu, Zhiliang Zhu, Huiyan Jiang and Beilei Wang. "A Novel Image Encryption Algorithm Based on Improved 3D Chaotic CatMap", The 9th International Conference for Young Computer Scientists, 2008.

13. Pawan N. Khade and Prof. Manish Narnaware, "3D Chaotic Functions for Image Encryption" International Journal of Computer Science, Vol. 9, Issue 3, No 1, May 2012.

14. Charles Petzold, "Programming Microsoft Windows With C#", Microsoft Press, 2002

15. William Stallings "Cryptography and Network Security: Principles and Practice". Fifth Edition, Prentice Hall, 2011.

16. Runkin et al. "Statistical Test Suite For Random And Pseudo Random Number Generators For Cryptographic Applications" . NIST special publication 800-22, 2001.

17. Marsaglia G. "DIEHARD Statistical Tests", http://stst.fsu.edu/pub/diehard , 1995.