Modern Applied Science; Vol. 10, No. 4; 2016
ISSN 1913-1844 E-ISSN 1913-1852
Published by Canadian Center of Science and Education

Implementation of the Binary Random Number Generator Using the Knight Tour Problem

Ali Shakir Mahmood[1, 2, 3], Mohd Shafry Mohd Rahim[1, 3, 4] & Nur Zuraifah Syazrah Othman[1]

[1] Faculty of Computing, University Technology Malaysia, Johor Bahru, Malaysia

[2] Department of Computer Science, College of Education, AL-Mustansiriyah University, Baghdad, Iraq

[3] UTM-ViCube, Faculty of Computing, University Technology Malaysia, Johor Bahru, Malaysia

[4] UTM-IRDA, Digital Media Centre, University Technology Malaysia, Johor Bahru, Malaysia

Correspondence: Ali Shakir Mahmood, Faculty of Computing, University Technology Malaysia, Johor Bahru, Malaysia. Tel: 6011-2374-2310. E-mail: asmjhm2006@yahoo.com

Received: August 30, 2015 Accepted: November 25, 2015 Online Published: January 13, 2016

dio: 10.5539/mas.v10n4p35 URL: http://dx.doi.org/10.5539/mas.v10n4p35

Abstract

A random number can be defined as a set of numbers produced by a numerical function, in which the next number is unpredictable and a relationship between successive occurrences is lacking. Moreover, these sequences cannot be reproduced unless the same generator function with an exact initial value is used. The design of a random number generator must overcome the previous problems of a low periodic and the capacity to reproduce the same sequence. This paper proposes the knight tour as a tool for generating pseudo random numbers. These random numbers can be use in the encryption process or in a password generator for network administrators. The randomness test suite is used to ensure the randomness of outcome sequences. Roughly, 75% of the test results obtained is better than the results from other works. The statistical properties and security analysis indicate that the knight tour application is highly successful in generating a pseudo random number with good statistical results, high linear complexity and strong capacity to withstand attacks.

Keywords: knight tour, random generator, random number, randomness tests

1. Introduction

Random numbers are widely used for many applications, such as keys for encryption and decryption, numerical analysis, simulating and modelling, as well as for selecting random samples from larger data sets (Li, 2012; Mahmood & Rahim, 2014; Tong, Liu, Zhang, Xu, & Wang, 2015). A random number can be generated by measuring random physical phenomena, such as temperature, wind speed and sunlight level. This type of generator is called a true random number generator (TRNG). TRNG requires additional equipment to produce random numbers and lacks the capacity to regenerate the same sequence unless the same initial key is used. It hardly ever regenerates the same random sequence because the sequence produced comes from the natural particular physical phenomenon. Another type of generator, called pseudorandom number generator (PRNG), uses mathematical algorithms to generate the random numbers. PRNG is more suitable for generating crypto keys because it is capable of regenerating the same random sequence and additional equipment to obtain the initial seed is unnecessary. PRNGs are also periodic generators, that is, they regenerate the same sequence after a certain round.

Generators for random numbers are characterized by several general properties. Firstly, they cannot predict the next number in the sequence. Secondly, the appearance probability of any element in the sequence is equal to other elements in the same sequence. Thirdly, the random sequence cannot be reproduced unless the same initial value is used (Rahman et al., 2014). Several methods for generating random numbers can be employed, such as genetic algorithms, neural networks and shift registers (Dubrova, Naslund, & Selander, 2014; Hrdy, Prazan, & Holoubek, 2014; Shenoy, Srikanth, & Srinivas, 2013; Toso & Resende, 2014).

The proposed method in this paper overcomes the problems in most previous works; these problems include a large initial seed and incapacity to regenerate the same sequence even with the same initial seed (Anceaume, Brasileiro, Ludinard, Sericola, & Tronel, 2011; Chen, 2013; Xingyuan, Xue, & Lin, 2012; Zhao, Ran, Yuan, Chi, & Ma, 2015).

The proposed method generates a pseudo binary random number sequence by implementing the knight tour problem. The knight tour problem is a mathematical problem involving the identification of a sequence of moves of a knight (on a chessboard) based on the knight movement rule in a chess game. The produced sequence is tested with a statistical test suite and security analysis is performed to verify that such sequence meets the specification of random numbers. The obtained results indicate that the proposed method could successfully generate pseudo random numbers with good statistical and security properties and high linear complexity.

This paper is organized into several sections. Section 2 briefly examines previous works on random number generators. Section 3 provides a detailed explanation of knight tour. Section 4 provides the general definition of random numbers. Section 5 describes the statistical tests. Section 6 discusses the proposed system. Section 7 details the experimental results. Finally, Section 8 presents the conclusions.

2. Related Works

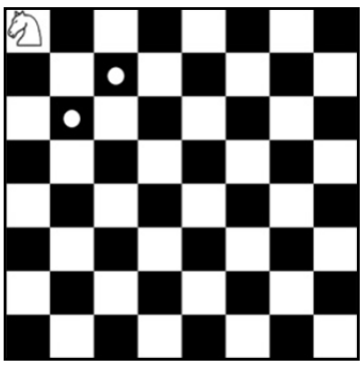
Several methods for generating random number have been proposed indecent years. Most of the methods were implemented in hardware rather than software (De Schryver et al., 2012; VishnuRaj & Yuvaraj, 2014). These methods provide a maximum periodic of random sequence and high throughput rate while adhering to established statistical standards by applying a seeding mechanism. To generate random numbers, previous generator shave used prime number theory (Kumar & Dhiman, 2015), audio and video sources (Chen, 2013), mouse motion and one-dimensional chaotic map (Hu, Liu, & Ding, 2013; Liu, Yang, Zhang, & Du, 2014), biometric feature of human iris (Taherdoost, Chaeikar, Jafari, & Shojae Chaei Kar, 2013) and even the contents of input/output buffers (Pardo, 2012).

However, most of these methods require large initial values to start (Chen, 2013; Deng, Hu, Xiong, Xiong, & Liu, 2015). This requirement is undesirable if the generator is used to produce cryptography keys. In the management of cryptographic keys, in which these keys need to be exchanged between sender and recipient, a large initial key will require additional efforts or resources to be sent; this requirement slows down delivery and spurs security problems. Furthermore, not all of the previous generators are ‘re-generate able’ (Hu et al., 2013; Xingyuan et al., 2012).

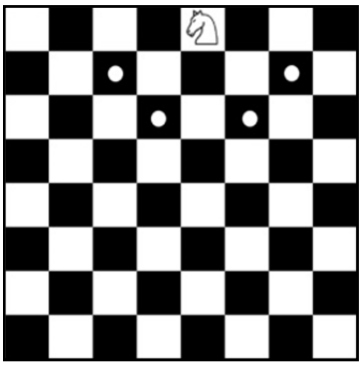
When a key sequence must be reproduced, the same initial value has to be used. Hence, generators that depend on a physical initial seed (i.e. wind speed and temperature) would be incapable of regenerating the same key sequence because the same initial value is impossible to obtain. Moreover, some of the previous generators fail in randomness statistical tests because the generated sequences are insufficiently random (Pashley, 2014). To overcome these issues and obtain a good random sequence, this paper proposes the use of the knight tour problem, which is described in detail in the next sections.

3. Knight Tour

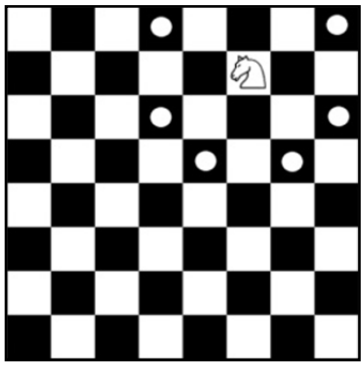
Knight tour is an arithmetical problem relating to a knight’s move on a chessboard. The knight is positioned on an empty board and peddling according to the rules of the knight moves in a chess game, where each square must be visited exactly once (Elkies, Stanley, Kleber, & Vakil, 2003). Five sets of moves are possible for the knight according to its position on the chessboard, as illustrated in Figure1. If the knight is at the corner of the chessboard, then it can be moved to two positions, as shown in Figure1 (a). The nearer the knight’s position in the centre, the more choices of movements it has, as depicted in Figures 1(b), (c) and (d). All of the possible movements can be expressed in a single representation, as demonstrated in Figure1 (e).



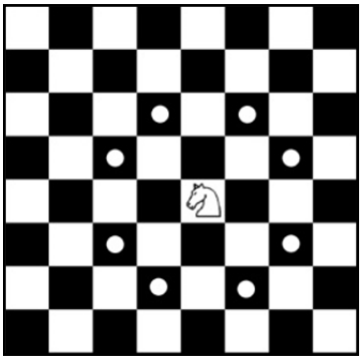
(a)



(b)



(c)



(d)

2	3	4	4	4	4	3	2
3	4	6	6	6	6	4	3
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
3	4	6	6	6	6	4	3
2	3	4	4	4	4	3	2

(e)

Figure 1. Different Knight Moves in a Chess Game, (a) Corner Position (b) Top Position (c) Semi-Top Position (d) Centre Position

A knight's tour is called a closed tour if the knight ends in the same starting position, such that it may tour the board again immediately with the same path. If it does not, the tour is called an open tour (Golenia, Golenia, & Erde, 2012). Both instances are illustrated in Figure 2.

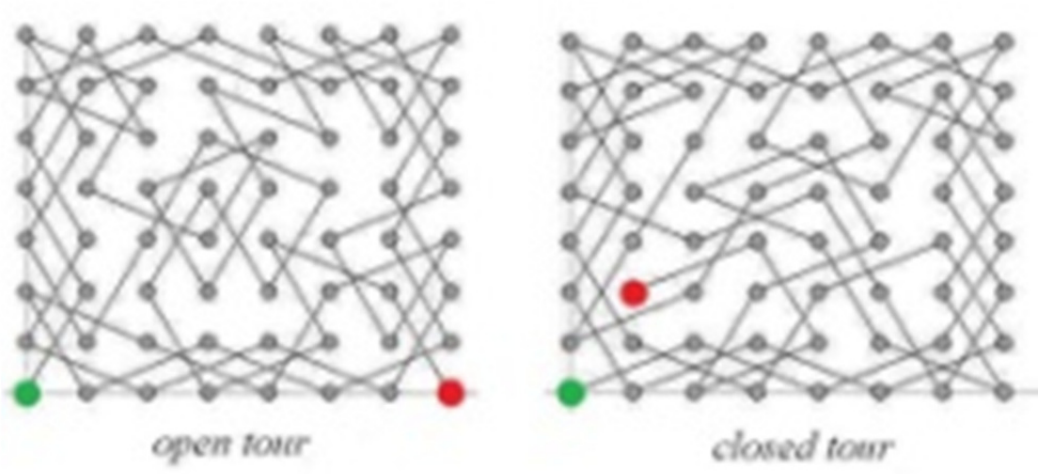


Figure 2. Two Strategies of Knight Moves in a Chess Game

The precise number of moves in an open tour remains unidentified, whereas the number of moves in a closed tour can be approximately determined by the number of moves in an equal chessboard size (n × n) (Gusfield, 1997; Löbbing & Wegener, 1996). Furthermore, researchers have suggested several methods for identifying the knight moves in an unequal chessboard (n × m) (Pampara, 2012). This type of generation can be classified into a

deterministic random number generator (Blum, Blum, & Shub, 1986).

Numerous methods have been previously used instating a programming solution to the knight tour problem; these methods include intelligence searching–backtracking algorithm (Bai & Cao, 2001), heuristic approach with minimal outlet (Zhang & Zhang, 2012), depth-first search algorithm (Gordon & Slocum, 2004) and divide–conquer method (Sen & Xiao, 2000). Moreover, optimization has been used to solve this problem; the algorithms for optimization include neural network algorithm (Takefuji & Lee, 1992), genetic algorithms (Al Gharaibeh, Qawagneh, & Zahawi, 2007) and ant colony optimization algorithm (Jiang, Bai, & Dong, 2009).

4. Random Numbers

The sequence of numbers can be called a random number if the numbers lack a relationship with one other, and the possibility of appearance is equal to all of the sequence numbers(Barker & Kelsey, 2007).

Two fundamental methods are used in generating random numbers. The first method, called non-deterministic, depends on physical processes that are changeable; these processes include radioactive decay, thermal noise, transistor noise, computer clock, keyboard and mouse movements. The second method computes random bits using an algorithm; this class of random bit generator is known as deterministic random bit generator (Armknecht, Maes, Sadeghi, Standaert, & Wachsmann, 2011).

Based on the observations performed with different random number generators, a good and reliable generator should exhibit several properties; the properties are summarized in Table1 (Akhshani, Behnia, Akhavan, Lim, & Hassan, 2010; Ferrenberg, Landau, & Wong, 1992; Hellekalek, 1998).

Table 1. General Properties of Random Number Generator

Property	Description
Impervious	An attacker who is aware of a portion of the input to the generator should be unable to use this information to recover the generator's state.
Opposing	An attacker who is able to feed a chosen input to the generator should be unable to influence its state in any predictable manner.
Resistant to analysis	An attacker who recovers a portion of the generators output; should be unable to recover any other generator state information from this.
Protect internal state	A generator should take steps to protect its internal state to ensure that it cannot recover through techniques.
Unequivocal for any activities	Explicit any actions such as extracting data in order to allow the conformance of the code to the generator design for easily checked.
Information leakage	Any leakage of internal states that would allow an attacker to predict further generator output should be regard as a catastrophic failure of the generator.
Tests	Perform any viable tests on the generated sequence to ensure that it is not producing bad output or is stuck in a cycle and repeatedly producing the same output.

5. Randomness Statistical Tests

Many tests can be applied to measure the randomness strength in the generated sequence (Fernández, Quintas, Sánchez, & Arias, 2015; Zhou, Liao, Wong, Hu, & Xiao, 2009). A binary string can be considered a random stream if no observable relationship exists between the individual bits of the sequence. The sequence generated by any algorithm should not be periodic (i.e. recurring at intervals of time). Sequences that are periodic cannot be regarded as true random sequences, but pseudorandom sequences (Abdulbari Ali, 2005; Pareschi, Setti, & Rovatti, 2010).

Many types of tests can be used for determining whether the generated sequence is random or not, such as Beker and Piper test suite (Beker & Piper, 1982), Diehard test suite (Marsaglia, 1996), TestU01 test suite (L'Ecuyer & Simard, 2007) and NIST-SP-800-22 test suite(Rukhin, Soto, Nechvatal, Smid, & Barker, 2013).

The NIST-SP-800-22 test suite is used in this work because it is one of the most extensively employed inspection standards thus far; moreover, this test suite can be applied on a binary sequence. This test contains 15 methods. The value of each test is named as P-value, which represents the degree of randomness of the tested binary sequence. A P-value larger than 0.01 indicates that the sequence passes the test, and is thus considered a random sequence (Rukhin et al., 2013).

6. Proposed System

From a functional viewpoint, the proposed system will apply the knight's tour problem as a tool for generating a pseudo random number that can be used as an encryption key or password generator. Several randomness tests are also used to verify the randomness strength of the numbers produced. The proposed work is presented in Figure 3.

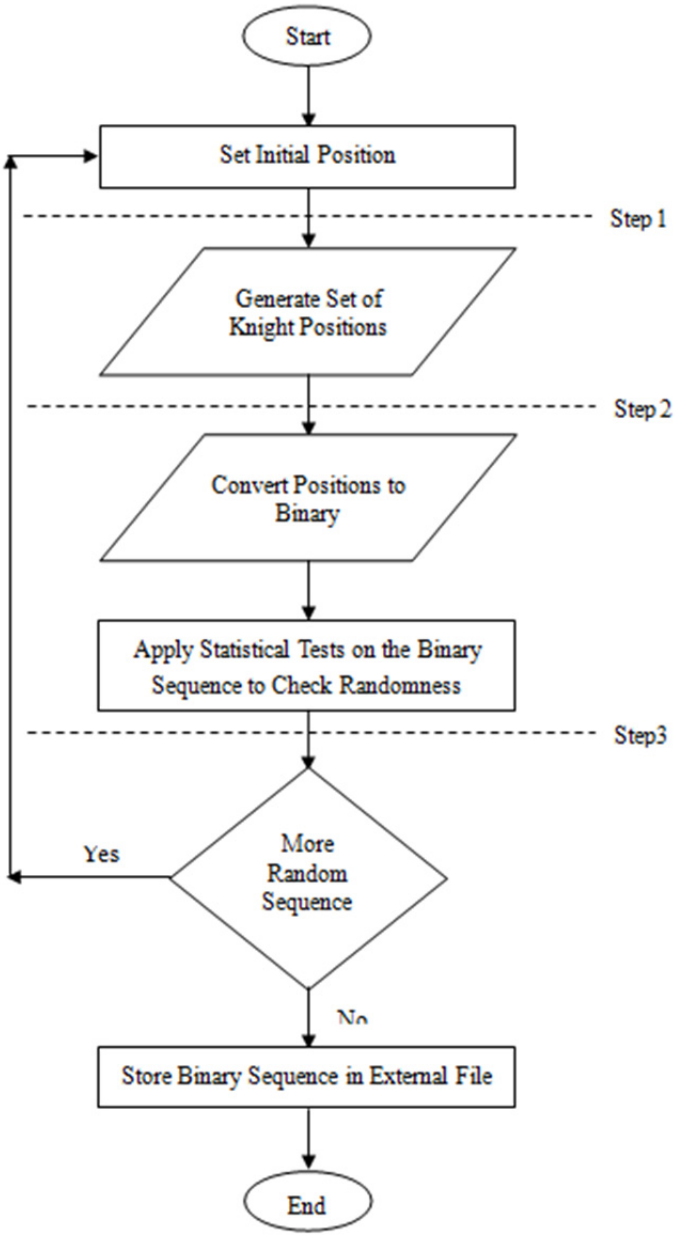


Figure 3. Flowchart of the Proposed System

The above figure represents the main steps for generating a random number. Three steps are applied for the generating process; these steps can be described as the following, choosing initial position on the chessboard and consider it as a seed for the generator. Apply the knight tour to fill the chessboard with the different moves. Then convert the sets of moves to binary number and tests them during statistical tests.

The following steps are under taken in the proposed work and describe the detail descriptions of the implementation of the proposed method, all which explained in figure 4, which represents the pseudo code of the proposed method:

```
Input: Initial Position (i,j)
Output: Set of Random Binary Random Number
Variable: i,j,x,y: integer
Begin
    x,y ← Chose an initial position
    For i=1 → x        Increment by 1 do
        For j=1 → y        Increment by 1 do

            x + 2 < xSize   and   y - 1 >= 0     x + 2, y – 1   right and upward
            x + 2 < xSize   and   y + 1 <ySize   x + 2, y + 1   right and downward
            x - 2 >= 0      and   y - 1 >= 0     x - 2, y – 1   left and upward
            x - 2 >= 0      and   y + 1 <ySize   x - 2, y + 1   left and downward
            x + 1 < xSize   and   y - 2 >= 0     x + 1, y – 2   upward and right
            x - 1 >= 0      and   y - 2 >= 0     x - 1, y – 2   upward and left
            x + 1 < xSize   and   y + 2 <ySize   x + 1, y + 2   downward and right
            x - 1 >= 0      and   y + 2 <ySize   x - 1, y + 2   downward and left
            Moves (i,j) ← (x,y)
        End For;
    End For;

Converting Sets of Knight Moves to Binary Sequence        Binary(i) ← Moves(i,j)

    For i=1 → x        Increment by 1 do
        For j=1 → y        Increment by 1 do
            While (Moves (i,j) Not Equal 0)
                Binary (i) ← ( Moves(i,j) Mod 2) * f;
                f ← f * 10;
                NewMoves (i,j)← Moves(i,j) / 2;
            End While;
        End For;
    End For;

    Apply Randomness Tests on Binary (i)
End.
```

Figure 4. The pseudo code of the Proposed method

Figure 4 explains the detail descriptions of the implementation, where the pseudo code can reflect the actual picture to the proposed method.

7. Experiment Results and Discussion

The process of generating random numbers begins with the specification of the start cell of the knight in an 8 × 8 chessboard. The generated tour is an open tour because the end cell is not in the same position as that of the start cell. Each run can generate 64 random numbers, and these numbers can subsequently become a 512-bit number when converted to a binary format. Figure 5 illustrates the sequence of generating the random binary bits.

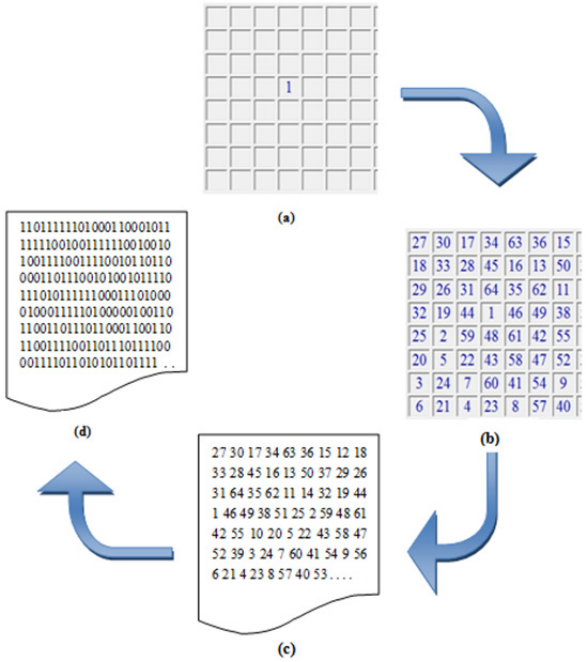


Figure 5. Generating Sequence: (a) Select an initial position (b) Knight moves to fill the chessboard (c) Save the moves in an external file (d) Convert to a binary format

The implementation of the proposed method explained in previous figure, where this figure describes the details descriptions of the generating steps. While choosing the initial position and then generating the knight moves to fill the chess board then convert these moves to binary sequence.

7.1 Statistical Analysis

For the statistical analysis, 10 set of pseudo random numbers are tested. The sets of numbers consist of 512 random bits for each set generated by the proposed method. The generated binary sequence by the proposed method passes the NIST-SP-800-22 (Rukhin et al., 2013) test suite. Table 2 and Figure 5 demonstrate the results obtained from the tests.

Table 2. NIST Tests Results

	1	2	3	4	5	6	7	8	9	10
Approximate Entropy Test	0.431	0.519	0.483	0.547	0.672	0.578	0.689	0.543	0.463	0.593
Block Frequency Test	0.311	0.542	0.209	0.214	0.281	0.353	0.401	0.394	0.295	0.389
Cumulative Sums Test	0.542	0.653	0.424	0.372	0.623	0.670	0.739	0.619	0.524	0.344
Discrete Fourier Transform Test	0.813	0.951	0.747	0.781	0.753	0.848	0.832	0.842	0.796	0.837
Frequency Test	0. 011	0. 019	0.014	0.012	0.013	0.010	0.019	0.014	0.013	0.019
Linear Complexity Test	0.841	0.992	0.713	0.860	0.718	0.817	0.880	0.866	0.745	0.885
Longest Run Test	0.615	0.802	0.573	0.656	0.651	0.769	0.887	0.758	0.589	0.738
Non Overlapping Template	0.341	0.407	0.310	0.210	0.365	0.252	0.351	0.209	0.391	0.289
Overlapping Template	0.316	0.385	0.373	0.231	0.348	0.459	0.421	0.417	0.320	0.469
Random Excursions Test	0.341	0.371	0.109	0.141	0.198	0.242	0.372	0.285	0.265	0.206
Rank Test	0.501	0.762	0.459	0.346	0.574	0.652	0.668	0.597	0.489	0.565
Runs Test	0.754	0.940	0.710	0.361	0.710	0.816	0.819	0.857	0.869	0.824
Serial Test	0.318	0.468	0.284	0.196	0.189	0.257	0.360	0.283	0.213	0.253
Universal Statistical Test	0.651	0.737	0.408	0.463	0.358	0.539	0.576	0.548	0.407	0.580

The result value obtained from each test represents the degree of randomness of the tested sequence. A value larger than 0.01 indicates that this sequence passes the test, and the sequence is considered a random number sequence (Rukhin et al., 2013). All of the results presented in Table 2 are larger than 0.01; thus, all 10 sequences can be regarded as random. However, the frequency test results shown in Table 2 are the same in all 10 sets of the generated sequences. This result occurs because the binary bits are generated from the same set of integer numbers, but with different positions in each set. Hence, the number of ones and zeroes will be equal in all sets, but a different consequence of zeroes and ones will emerge. In addition, the low values of the frequency test and nearness to the failure condition may be attributed to the process of converting the decimal numbers to a binary system. Each decimal number is equal to 8 bits in a binary system; by contrast, decimal numbers smaller than 64 indicate that most bits in any converted number are equal to zero. Consequently, the number of zeroes is higher

than the number of ones in the generated binary sequence.

After proving that the proposed generator is a random generator with good statistical properties, the advantage of this generator over the other generators should be demonstrated through a comparison of the results with those of previous works. As illustrated in Table 3, the comparison between the average of the 10 generated sequences using the proposed generator and four different generators indicates that they all use NIST-SP-800-22 as a random test method.

Table 3. Comparison of the Proposed NIST Test Results and the Results of Others Works

Tests	Proposed Generator	Hyper chaotic Map, (Tong et al., 2015)	Chen Chaotic Map, (Hu et al., 2013)	Discrete Devaney's Chaotic Map, (Deng et al., 2015)	Mouse Movement and Chaotic Hash Function, (Zhou et al., 2009)
Approximate Entropy Test	0.9769	0.8184	0.1554	0.5039	0.9737
Block Frequency Test	0.7938	0.2095	0.4968	0.7222	0.2236
Cumulative Sums Test	0.9012	0.5922	0.8894	0.5850	0.1875
Discrete Fourier Transform Test	0.6204	0.7120	0.1957	0.5946	0.2622
Frequency Test	0.0107	0.1455	0.8556	0.5283	0.1223
Linear Complexity Test	0.8694	0.9018	0.8687	0.4973	0.3267
Longest Run Test	0.9091	0.3943	0.6153	0.9087	0.0616
Non Overlapping Template	0.5802	0.4064	-	0.4774	-
Overlapping Template	0.7904	0.0459	0.6224	0.7742	-
Random Excursions Test	0.6503	0.3757	0.2117	0.4585	0.6325
Rank Test	0.5718	0.8510	0.3430	0.4083	0.0391
Runs Test	0.9038	0.4811	0.8965	0.1390	-
Serial Test	0.6836	0.2110	0.0672	-	0.6434
Universal Statistical Test	0.6792	-	0.6508	0.1558	-

Table 3 shows a comparison between the average results of the 10 random sets generated by the proposed method and four binary random generation methods (Tong et al., 2015), (Hu et al., 2013),(Deng et al., 2015)and (Zhou et al., 2009), respectively. All of these methods use NIST-SP-800-22 as a random testing suite. A comparison of the items in Table 3 suggests that several items of the proposed generator exhibit better results than those of the other four generators. This result is attributed to the dependence of the works of the pass tests on blocks, in which the probability of the occurrence of the redundancy of the block is extremely low against other tests whose works depend on the probability of bit occurrence. The proposed method clearly achieves the goal of generating random numbers with good statistical results compared with the other examined generators. However, the frequency test values do not meet the perfect value as explained in the preceding discussion of results.

7.2 Security Analysis

An excellent generator method must be robust against several types of cryptanalytic, statistical and brute force attacks. Different types of analysis are applied to measure the strength of the proposed generator method against different decryption trials, as described in the subsequent section.

7.2.1 Key Space Analysis

The size of the key space represents the overall number of keys used for encryption. A direct relationship exists between key space size and security. A proficient encryption method must have a large key space (i.e. approximately larger than 2^{100}) to render brute force attacks infeasible (Alvarez & Li, 2006). In the proposed generator, each round can generate 512 random bits from different 64 decimal numbers; the total number of

different keys is equal to $2^{512} \approx 1.34 \times 10^{154}$; thus, this key space is sufficiently large for resisting all forms of brute force attack.

7.2.2 Key Sensitivity Analysis

Key sensitivity is examined by selecting a set of keys and performing a small alteration on it to produce new keys; the differences between these two sets are subsequently computed. For a good encryption algorithm, key sensitivity should be more than 50% (Saranya, Mohan, & Anusudha, 2014). The testing conducted in this work shows a variance ratio of approximately 62%, even if the change is small. The good results from this analysis indicate that the selected plain text attacks and linear cryptanalysis do not influence the generated keys; consequently, the encryption algorithm is unaffected.

7.2.3 Linear Complexity Analysis

The linear complexity of algorithm concerned on how fast or slow for particular performs. In the proposed generator can be defined as a slight recurrence in linear feedback shift register (LFSR) to generate the random sequences. A proficient Berlekamp–Massey algorithm of linear complexity is proposed in (Eidelman & Gohberg, 1997).

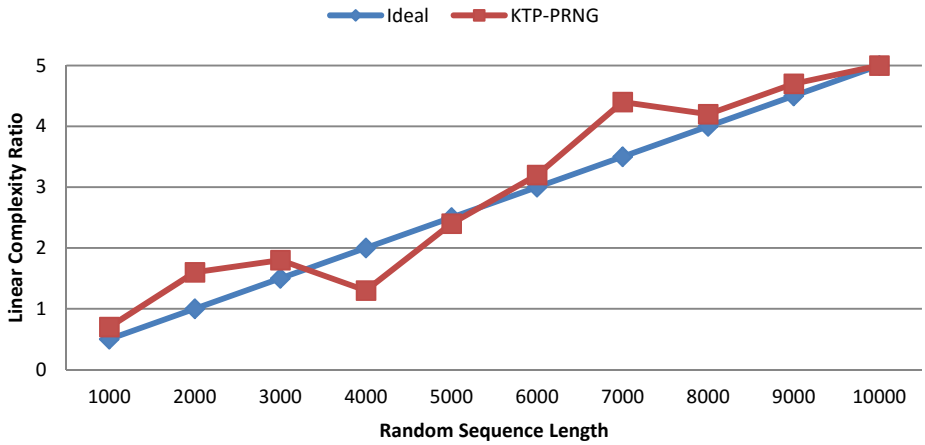


Figure 6. Linear Complexity Analyses

This algorithm computes the smallest LFSR. The results obtained from this analysis are presented in Figure 6, where the linear complexity is near the ideal values, thereby indicating that the generated sequences exhibit a linear complexity against different types of attack.

8. Conclusions

This paper has designed and tested the use of the knight tour algorithm as a pseudo random number generator; this process was achieved in two steps. Firstly, a knight move sequence was generated to fill an 8 × 8 chessboard, and these numbers were considered pseudo random numbers. Secondly, the NIST test suite was used in checking randomness and in considering the criteria of accepted randomness. The original goal was clearly achieved by solving most of the difficulties encountered by the generator designers of small initial value and not depending on the physical seed. The statistical tests and security analysis demonstrate that the proposed system is highly successful in generating pseudo random numbers with good statistical properties, large key space and resistance to various types of attacks. The proposed system can be used for generating cryptographic keys, as password generator, game key generator and probability in mathematics.

Acknowledgments

The authors are grateful to the Ministry of Higher Education and Scientific Research, Iraq, for providing a Ph.D. sponsorship. Furthermore, I would like also to thank University Technology Malaysia (UTM), Johor for their grant, Vot. NO.: Q.J130000.2428.02G28 (Flagship) and Q.J130000.2528.09H69 (GUP).

References

Abdulbari, A. S. (2005). *Improving the Randomness of Output Sequence for the Advanced Encryption Standard Cryptographic Algorithm.* Universiti Putra Malaysia.

Akhshani, A., Behnia, S., Akhavan, A., Lim, S. C., & Hassan, Z. (2010). Pseudo random number generator based on synchronized chaotic maps. *International Journal of Modern Physics C, 21*(02), 275-290. http://dx.doi.org/10.1142/S0129183110015117

Al Gharaibeh, J., Qawagneh, Z., & Al Zahawi, H. (2007). *Genetic Algorithms with Heuristic-Knight's Tour Problem.* Paper presented at the GEM.

Alvarez, G., & Li, S. (2006). Some basic cryptographic requirements for chaos-based cryptosystems. *International Journal of Bifurcation and Chaos, 16*(08), 2129-2151. http://dx.doi.org/10.1142/S0218127406015970

Anceaume, E., Brasileiro, F., Ludinard, R., Sericola, B., & Tronel, F. (2011). Dependability evaluation of cluster-based distributed systems. *International Journal of Foundations of Computer Science, 22*(05), 1123-1142. http://dx.doi.org/10.1142/S0129054111008593

Armknecht, F., Maes, R., Sadeghi, A., Standaert, O. X., & Wachsmann, C. (2011). *A formalization of the security features of physical functions.* Paper presented at the Security and Privacy (SP), 2011 IEEE Symposium on.

Bai, S., & Cao, C. (2001). A Novel Algorithm of Digital Image Scrambling and Hiding. *Computer Engineering, 11*, 7.

Barker, E. B., & Kelsey, J. M. (2007). *Recommendation for random number generation using deterministic random bit generators (revised)*: US Department of Commerce, Technology Administration, National Institute of Standards and Technology, Computer Security Division, Information Technology Laboratory.

Beker, H., & Piper, F. (1982). *Cipher systems: The protection of communications*: Northwood Books.

Blum, L., Blum, M., & Shub, M. (1986). A simple unpredictable pseudo-random number generator. *SIAM Journal on computing, 15*(2), 364-383. http://dx.doi.org/10.1137/0215025

Chen, I. T. (2013). Random numbers generated from audio and video sources. *Mathematical Problems in Engineering*. http://dx.doi.org/10.1155/2013/285373

De Schryver, C., Schmidt, D., Wehn, N., Korn, E., Marxen, H., Kostiuk, A., & Korn, R. (2012). A hardware efficient random number generator for nonuniform distributions with arbitrary precision. *International Journal of Reconfigurable Computing*, 12. http://dx.doi.org/10.1155/2012/675130

Deng, Y., Hu, H., Xiong, W., Xiong, N. N., & Liu, L. (2015). Analysis and Design of Digital Chaotic Systems With Desirable Performance via Feedback Control. *Systems, Man, and Cybernetics Society*, 99. http://dx.doi.org/10.1109/TSMC.2015.2398836

Dubrova, E., Naslund, M., & Selander, G. (2014). *Secure and efficient LBIST for feedback shift register-based cryptographic systems.* Paper presented at the Test Symposium (ETS), 2014 19th IEEE European, Paderborn.

Eidelman, Y., & Gohberg, I. (1997). Inversion formulas and linear complexity algorithm for diagonal plus semiseparable matrices. *Computers & Mathematics with Applications, 33*(4), 69-79. http://dx.doi.org/10.1016/S0898-1221(97)00008-4

Elkies, N. D., Stanley, R. P., Kleber, M., & Vakil, R. (2003). The mathematical knight. *The Mathematical Intelligencer, 25*(1), 22-34. http://dx.doi.org/10.1007/BF02985635

Fernández, N., Quintas, F., Sánchez, L., & Arias, J. (2015). Social Noise: Generating Random Numbers from Twitter Streams. *Fluctuation and Noise Letters, 14*(01), 1550012. http://dx.doi.org/10.1142/S0219477515500121

Ferrenberg, A. M., Landau, D., & Wong, Y. J. (1992). Monte Carlo simulations: Hidden errors from" good" random number generators. *Physical Review Letters, 69*(23), 3382-3384.

Golenia, B., Golenia, S., & Erde, J. (2012). The closed knight tour problem in higher dimensions. *arXiv preprint arXiv:1202.5291*.

Gordon, V. S., & Slocum, T. J. (2004). *The knight's tour-evolutionary vs. depth-first search.* Paper presented at the Evolutionary Computation, 2004. CEC2004. Congress on.

Gusfield, D. (1997). *Algorithms on strings, trees and sequences: computer science and computational biology*. Cambridge university press.

Hellekalek, P. (1998). Good random number generators are (not so) easy to find. *Mathematics and Computers in Simulation, 46*(5), 485-505. http://dx.doi.org/10.1016/S0378-4754(98)00078-0

Hrdy, T., Prazan, M., & Holoubek, P. (2014). RANDOM NUMBER GENERATOR: US Patent 20,140,324,934.

Hu, H., Liu, L., & Ding, N. (2013). Pseudorandom sequence generator based on the Chen chaotic system. *Computer Physics Communications, 184*(3), 765-768. http://dx.doi.org/10.1016/j.cpc.2012.11.017

Jiang, D., Bai, S., & Dong, W. (2009). *An ant colony optimization algorithm for knight's tour problem on the chessboard with holes.* Paper presented at the 1st International Workshop on Education Technology and Computer Science.

Kumar, R., & Dhiman, M. (2015). Secured Image Transmission Using a Novel Neural Network Approach and Secret Image Sharing Technique. *International Journal of Signal Processing, Image Processing and Pattern Recognition, 8*(1), 161-192.

L'Ecuyer, P., & Simard, R. (2007). TestU01: A C Library for Empirical Testing of Random Number Generators. *Transactions on Mathematical Software ACM, 33*(4), 22-40.

Li, K. (2012). Performance analysis and evaluation of random walk algorithms on wireless networks. *International Journal of Foundations of Computer Science, 23*(04), 779-802.

Liu, G., Yang, F., Zhang, Y., & Du, C. (2014). A New True Random Number Generator Based on Feedback System. *Journal of Computational Information Systems, 10*(19), 8469-8476.

Löbbing, M., & Wegener, I. (1996). The number of knight's tours equals 33,439,123,484,294—counting with binary decision diagrams. *The Electronic Journal of Combinatorics, 3*(1), R5.

Mahmood, A. S., & Rahim, M. S. M. (2014). State of the Art of Image Ciphering: A Review. *International Journal of Computer Science Issues (IJCSI), 11*(2).

Marsaglia, G. (1996). DIEHARD A battery of tests of randomness. Retrieved from http://stat.fsu.edu/pub/diehard/

Pampara, G. (2012). *Angle modulated population based algorithms to solve binary problems.* University of Pretoria, Pretoria.

Pardo, J. L. G. (2012). *Introduction to Cryptography with Maple*: Springer Science & Business Media.

Pareschi, F., Setti, G., & Rovatti, R. (2010). Statistical Testing of a Chaos Based CMOS True-Random Number Generator. *Journal of Circuits, Systems, and Computers, 19*(04), 897-910. http://dx.doi.org/10.1142/S0218126610006517

Pashley, P. J. (2014). On generating random sequences. *A Handbook for Data Analysis in the Behaviorial Sciences: Volume 1: Methodological Issues Volume 2: Statistical Issues*, 395.

Rahman, M. T., Xiao, K., Forte, D., Zhang, X., Shi, J., & Tehranipoor, M. (2014). *TI-TRNG: Technology Independent True Random Number Generator.* Paper presented at the Proceedings of the The 51st Annual Design Automation Conference on Design Automation Conference.

Rukhin, A., Soto, J., Nechvatal, J., Smid, M., & Barker, E. (2013). NIST Special Publication 800-22 Revision 1a: A statistical test suite for random and pseudorandom number generators for cryptographic applications. *Date of Access, 1*(03).

Saranya, M., Mohan, A. K., & Anusudha, K. (2014). *A composite image cipher using DNA sequence and genetic algorithm.* Paper presented at the Contemporary Computing and Informatics (IC3I), 2014 International Conference on.

Sen, B., & Xiao, F. Y. (2000). An Optimal Algorithm for Searching a Hamilton Cycle of the Knight Tour Problem. *Computer Engineering & Science, 2*(2).

Shenoy, H. A., Srikanth, R., & Srinivas, T. (2013). Efficient quantum random number generation using quantum indistinguishability. *Fluctuation and Noise Letters, 12*(04), 1350020. http://dx.doi.org/10.1142/S021947751350020X

Taherdoost, H., Chaeikar, S. S., Jafari, M., & Shojae Chaei Kar, N. (2013). Definitions and Criteria of CIA Security Triangle in Electronic Voting System. *International Journal of Advanced Computer Science and Information Technology (IJACSIT), 1*, 14-24.

Takefuji, Y., & Lee, K. C. (1992). Neural network computing for knight's tour problems. *Neurocomputing, 4*(5), 249-254. http://dx.doi.org/10.1016/0925-2312(92)90030-S

Tong, X., Liu, Y., Zhang, M., Xu, H., & Wang, Z. (2015). An Image Encryption Scheme Based on Hyperchaotic Rabinovich and Exponential Chaos Maps. *Entropy, 17*(1), 181-196. http://dx.doi.org/10.3390/e17010181

Toso, R. F., & Resende, M. G. (2014). A C++ application programming interface for biased random-key genetic algorithms. *Optimization Methods and Software*, 1-13.

VishnuRaj, C., & Yuvaraj, M. S. (2014). RANDOM NUMBER GENERATION BY USING RESEEDING MIXING PSUEDO RANDOM NUMBER GENERATOR. *International Journal of Power Control and Computation(IJPCSC), 6*(1), 36-42.

Xingyuan, W., Xue, Q., & Lin, T. (2012). A novel true random number generator based on mouse movement and a one-dimensional chaotic map. *Mathematical Problems in Engineering*. http://dx.doi.org/10.1155/2012/931802

Zhang, J., & Zhang, Z. (2012). Ant Colony Algorithms and Logistics Distribution Solutions. *Communications and Information Processing*, 734-740.

Zhao, T., Ran, Q., Yuan, L., Chi, Y., & Ma, J. (2015). Image encryption using fingerprint as key based on phase retrieval algorithm and public key cryptography. *Optics and Lasers in Engineering, 72*, 12-17. http://dx.doi.org/10.1016/j.optlaseng.2015.03.024

Zhou, Q., Liao, X., Wong, K. W., Hu, Y., & Xiao, D. (2009). True random number generator based on mouse movement and chaotic hash function. *Information Sciences, 179*(19), 3442-3450. http://dx.doi.org/10.1016/j.ins.2009.06.005