International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 5 | ISSN : 2456-3307

Performance Enhancement of Cryptographic Algorithms by Increasing Randomness through Nesting In Random Number Generators

[1]Antu Annam Thomas, [2]Varghese Paul
[1]Department of Computer Application, Mar Thoma College, Thiruvalla, Kerala, India
[2]Department of Information Technology, Rajagiri School of Engineering and Technology, Rajagiri Valley, Kakkanad, Kerala, India

ABSTRACT

Randomness is a term that is commonly used in applications like cryptography. In cryptography randomness finds its role in seed value generation. There are various traditional random number generators. Linear Congruential Generator (LCG) is one among them. In order to improve the efficiency of the generated series concept of nesting is introduced into LCG to form Nested Linear Congruential Generator (NLCG). In this paper LCG is compared with NLCG. In the case of LCG the period of the generated sequence will be depending upon the value of increment and multiplicand chosen. But in the case of NLCG the period is always infinity. This is the greatest advantage of NLCG when compared to LCG. Both statistical and graphical analysis is done. The discussion proves that efficiency of the generated series has been improved when nesting is introduced, thus in turn improving the efficiency of the cryptographic algorithms.

Keywords: Cryptography; Data Security; True Random Number Generator; Pseudo random number generator; Linear Congruential Generator; Prime number; Kolmogorov Smirnov Test; Runs Test

I. INTRODUCTION

Randomness plays an important role in applications where unpredictability is inevitable. Cryptography is one such application where the random numbers are used as seed values. There are a large number of random number generators. The efficiency of the generator depends on how much unpredictable, independent and uniform is the generated series.[1][2][3]

The random number generators can be true random number generators or pseudo random number generators. Since physical entropy sources are used in true random number generators the generated series is more random when compared with pseudo random number generators.[4]Linear Congruential Generator, Blum Blum Shub Generator, Linear Feedback Shift Register, XORShift Generators are some of the traditional pseudo random number generators.[5]

Though there are many true random number generators such as Hotbits, Laser, Random.org and so on, pseudo random number generators are used in most of the applications where randomness comes into play. [6][7][19]True random number generators produce random series that are truly random since it uses true random source as their source of entropy. But the bit rate of True Random number generator is too low thus in most of the cases Pseudo Random number generators and true random number generators are used in combination.[14][15]

In this paper two random number generators Linear Congruential Generator is compared with Nested Linear Random Number Generator. The analysis concludes that nested concept when introduced the traditional LCG has improved its performance. In nested random series the period is always infinity that is a subsequence never repeats.

II. Random Number Generators

Random number generators are to generate series of numbers that are independent, uniform and unpredictable. Random number generators can be divided into two pseudo random number generators and true random number generators. The classification is based on its method of generation. [11]

True random number generators use real world phenomena as source of entropy. The series thus generated is truly random. But true random number generators requires a lot of hardware and hence it is expensive, bit rate is very low, can be influenced by changes in physical environment and entropy sources could be attacked by intruders. [8][9][10]

Pseudo random number generators use computational algorithms for its generation. Though the series may not be ideally random it will mimic the true random series.

Thus what is done usually is to combine true and pseudo random number generators. [12][13]

III. Linear Congruential Generator (LCG)

LCG is one of the oldest and most popular random number generators. LCG is simple and easy to implement.
Random series is generated based on a piecewise linear equation given below.

$$X_{n+1}=(aX_n+c)\bmod m \qquad (1)$$

Here,

X is the sequence of random numbers

m, $0<m$ -	modulus
$a, 0<a<m$-	multiplier
$c, 0\leq c<m$-	increment
$X_0, 0\leq X_0<m$-	seed value

LCG is fast and requires very less memory. Period of the generated series depends upon the value of ‘m’. LCG is not suitable for applications like cryptography where high security is demanded.[16][17]

IV. Nested Linear Congruential Generator(NLCG)

In Nested LCG concept of nesting is introduced into traditional LCG. The series is generated based on the equation given below. The equation is the same linear piecewise equation as the traditional LCG. But here multiplier and increment is not a constant value as ‘a’ and ‘c’ in equation (1). ‘multi’ and ‘incr’ are the random numbers generated by two other random number series.

NLCG consist of three steps:

i. Getting the seed value
ii. Generating the series

A. Getting the seed value:

Based on the current system clock value read a pixel value of the current picture captured by system camera is read.

Based on the pixel value read two prime numbers $p1_0$ and $p2_0$ are generated. Here $p1_0$ is the greatest prime number less than the read pixel value and $p2_0$ is the smallest prime number greater than the read pixel value. Seed value is given by,

$$X_0=p1_0*p2_0 \bmod m \qquad (2)$$

m is relatively prime to $p1_0*p2_0$

Thus seed value X_0 is the product of two prime numbers. Generated seed value is cryptographically secure due to two factors, difficulty in factorizing product of two prime numbers and true randomness introduced, clock value and pixel value.

The above mentioned complexity increases the efficiency of the system and makes the job of cryptanalyst difficult or rather impossible.[20]

B. Generating the series

$$X_i=(X_{i-1}*b_i+a_i)\bmod m \qquad (3)$$

$$b_i=f_i*p1_{i-1}*p2_{i-1} \qquad (4)$$

f_i is the pixel value read from the image based on $p1_{i-1}$ and $p2_{i-1}$

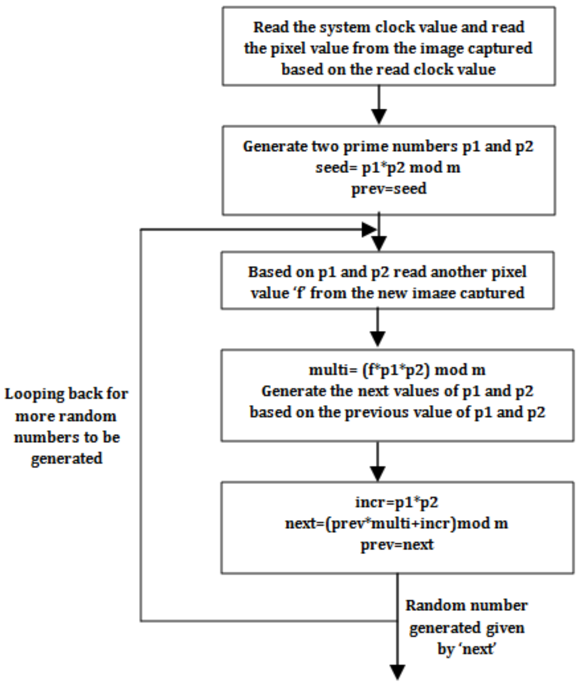
Now based on $p1_{i-1}$ and $p2_{i-1}$ next pair of prime numbers is generated $p1_i$ and $p2_i$.

$$a_i=p1_i*p2_i \qquad (5)$$

Equations (3), (4) and (5) together generate the random number series. The next element in the generated random sequence ‘X_i’ not only depends on previous value X_{i-1} but also on a_i and b_i which are next elements of two other random series generated by the equations (3) and (4).

Thus two random series values contribute for final random series generation. That is two random series is nested within another series.

Nesting ensures that sequence is never repeated within the final series being generated and period is infinity. NLCG algorithm is pictorially represented in the flowchart given below.



Read the system clock value and read the pixel value from the image captured based on the read clock value
Generate two prime numbers p1 and p2 seed= p1*p2 mod m prev=seed
Based on p1 and p2 read another pixel value 'f' from the new image captured
multi= (f*p1*p2) mod m Generate the next values of p1 and p2 based on the previous value of p1 and p2
incr=p1*p2 next=(prev*multi+incr)mod m prev=next
Looping back for more random numbers to be generated
Random number generated given by 'next'

Figure 1. Flowchart of NLCG

NLCG is cost effective since no expensive hardware is required. Seed value generated is sufficiently complex since true randomness and prime factorization problem is used in this step. Nested concept used during series generation makes the series unpredictable and random and subsequence never repeats in the generated sequence. [6][15]

Complexities involved in this method include true entropy sources introduced in the generation, prime factorization problem, concept of nesting used in the algorithm.

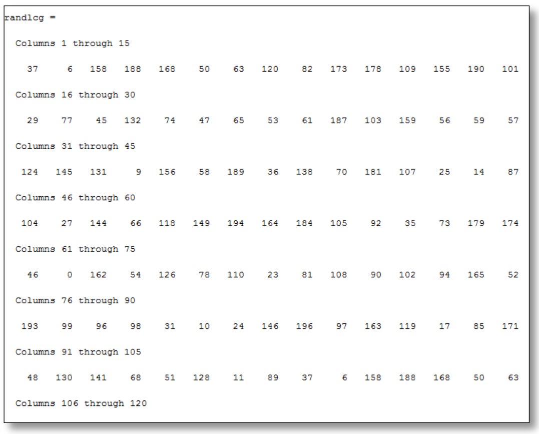
V. Implementation

Both LCG and NLCG were implemented in Matlab and output was analyzed.
System clock value was read and a pixel value was read from the current image captured based on the pixel value read. Then two prime numbers $p1_0$ and $p2_0$ was generated. Seed value is evaluated based on equation (2).

The same seed value was used for LCG also.

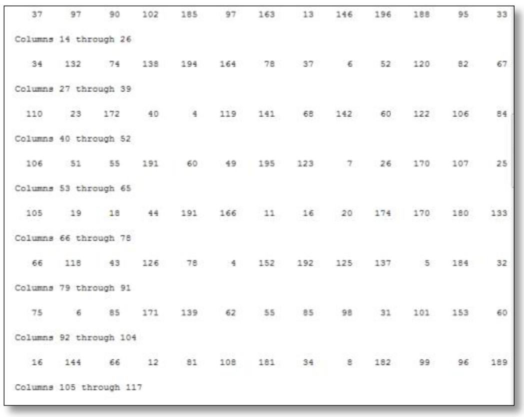
Now the series is generated for NLCG using the equations (3),(4) and (5) and for LCG using the equation (1).

The output got for LCG and NLCG after implementing the algorithm is given below. 'm' was chosen to be 197.



```
randlcg =

  Columns 1 through 15

    37     6   158   188   168    50    63   120    82   173   178   109   155   190   101

  Columns 16 through 30

    29    77    45   132    74    47    65    53    61   187   103   159    56    59    57

  Columns 31 through 45

   124   145   131     9   156    58   189    36   138    70   181   107    25    14    87

  Columns 46 through 60

   104    27   144    66   118   149   194   164   184   105    92    35    73   179   174

  Columns 61 through 75

    46     0   162    54   126    78   110    23    81   108    90   102    94   165    52

  Columns 76 through 90

   193    99    96    98    31    10    24   146   196    97   163   119    17    85   171

  Columns 91 through 105

    48   130   141    68    51   128    11    89    37     6   158   188   168    50    63

  Columns 106 through 120
```

Figure 2. Output showing the first 105 elements in the random series generated by LCG



```
    37    97    90   102   185    97   163    13   146   196   188    95    33

  Columns 14 through 26

    34   132    74   138   194   164    78    37     6    52   120    82    67

  Columns 27 through 39

   110    23   172    40     4   119   141    68   142    60   122   106    84

  Columns 40 through 52

   106    51    55   191    60    49   195   123     7    26   170   107    25

  Columns 53 through 65

   105    19    18    44   191   166    11    16    20   174   170   180   133

  Columns 66 through 78

    66   118    43   126    78     4   152   192   125   137     5   184    32

  Columns 79 through 91

    75     6    85   171   139    62    55    85    98    31   101   153    60

  Columns 92 through 104

    16   144    66    12    81   108   181    34     8   182    99    96   189

  Columns 105 through 117
```

Figure 3. Output showing the first 104 elements in the random series generated by NLCG

VI. Result Analysis

A. Scatter Diagram Analysis

Scatter diagram analysis proves that the series posses good randomness and that there is no linear relationship or correlation between the generated random values.
For analyzing first 30 elements are chosen. Points on the graph are divided into four quadrants. If there are X

points on the graph, Count X/2 points from top to bottom and draw a horizontal line. Count X/2 points from left to right and draw a vertical line. Here 30 points are considered so lines are drawn after 15 points and graph divided into four quadrants.

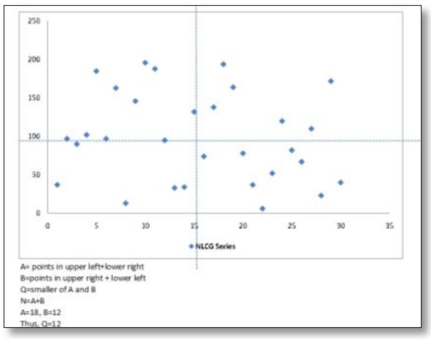


Figure 4. NLCG Scatter Diagram

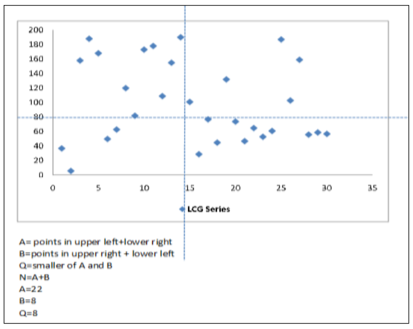


Figure 5. LCG Scatter Diagram

Limit given in the Trend Test Table for sample of 30 is 9. Here for NLCG value of Q is 12 and is greater than the limit 9 hence the numbers in the series is drawn by random chance. But for LCG value of Q is 8 which is less than the limit value 9 hence numbers in the series are somehow related. Hence scatter diagram analysis shows that NLCG is a better random number generator than LCG.

B. Bar Graph Analysis

The bar graph is plotted for first 200 random numbers generated both for LCG and NLCG.

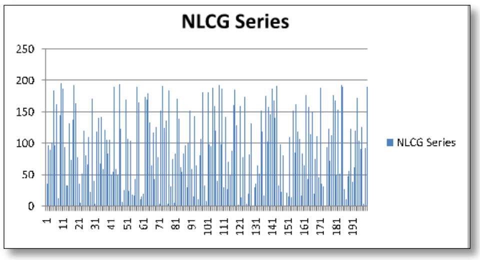


Figure 6. Bar Graph of NLCG output

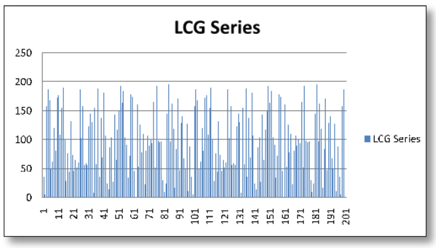


Figure 7. Bar Graph of LCG ouput

The plotted bar graph shows that the generated sequence is random and a subsequence never repeats for NLCG. But the case is different with LCG here the period is 98 that is after 98 numbers being generated the generated sequence repeats again and this process continues. But in the case of NLCG the subsequence never repeats. This is because the increment and the multiplicand value is never constant it forms the random number in another nested random series. Hence bar graph analysis also shows NLCG is better than LCG as a random number generator.

C. Kolmogorov Smirnov Test

KS test can be used to check the randomness of the numbers generated by a RNG that is allowed to take on any value within a certain interval, leading to a continuous cdf [16]. KS Test is conducted on both NLCG and LCG results and the analysis table is given below.[18]

H0 = Sequence being tested is random
Ha = Sequence being tested is not random

Table1. K S Test Analysis of NLCG

i	random	normalised	(i/n-xi)	xi-(i-1)/n
1	6	0.030456853	0.002876481	0.030456853
2	13	0.065989848	0.000676819	0.032656514
3	23	0.116751269	-0.016751269	0.050084602
4	33	0.16751269	-0.034179357	0.06751269
5	34	0.172588832	-0.005922166	0.039255499
6	37	0.187817259	0.012182741	0.021150592
7	37	0.187817259	0.045516074	-0.012182741
8	40	0.203045685	0.063620981	-0.030287648
9	52	0.263959391	0.036040609	-0.002707276
10	67	0.340101523	-0.00676819	0.040101523
11	74	0.375634518	-0.008967851	0.042301184
12	78	0.395939086	0.004060914	0.02927242
13	82	0.416243655	0.017089679	0.016243655
14	90	0.456852792	0.009813875	0.023519459
15	95	0.482233503	0.017766497	0.015566836
16	97	0.492385787	0.040947547	-0.007614213
17	97	0.492385787	0.07428088	-0.040947547
18	102	0.517766497	0.082233503	-0.048900169
19	110	0.558375635	0.074957699	-0.041624365
20	120	0.609137056	0.057529611	-0.024196277
21	132	0.670050761	0.029949239	0.003384095
22	138	0.700507614	0.032825719	0.000507614
23	146	0.741116751	0.025549915	0.007783418
24	163	0.827411168	-0.027411168	0.060744501
25	164	0.83248731	0.000846024	0.03248731
26	172	0.873096447	-0.00642978	0.039763113
27	185	0.939086294	-0.039086294	0.072419628
28	188	0.954314721	-0.020981387	0.054314721
29	194	0.984771574	-0.018104907	0.05143824
30	196	0.994923858	0.005076142	0.028257191

From the table generated for NLCG output

K+ = 0.0822335

K- = 0.0724196

From KS test table at n=30 and 1-α=0.9

K=0.21756

K+ < K and K- < K hence sequence generated by NLCG is random and pass KS test

Table 2. K S Test Analysis of LCG

i	random	normalised	(i/n-xi)	xi-(i-1)/n
1	6	0.030456853	0.002876481	0.030456853
2	29	0.147208122	-0.080541455	0.113874788
3	37	0.187817259	-0.087817259	0.121150592
4	45	0.228426396	-0.095093063	0.128426396
5	47	0.23857868	-0.071912014	0.105245347
6	50	0.253807107	-0.053807107	0.08714044
7	53	0.269035533	-0.0357022	0.069035533
8	56	0.284263959	-0.017597293	0.050930626
9	57	0.289340102	0.010659898	0.022673435
10	59	0.299492386	0.033840948	-0.000507614
11	61	0.30964467	0.057021997	-0.023688663
12	63	0.319796954	0.080203046	-0.046869712
13	65	0.329949239	0.103384095	-0.070050761
14	74	0.375634518	0.091032149	-0.057698816
15	77	0.390862944	0.109137056	-0.075803723
16	82	0.416243655	0.117089679	-0.083756345
17	101	0.512690355	0.053976311	-0.020642978
18	103	0.52284264	0.07715736	-0.043824027
19	109	0.553299492	0.080033841	-0.046700508
20	120	0.609137056	0.057529611	-0.024196277
21	132	0.670050761	0.029949239	0.003384095
22	155	0.78680203	-0.053468697	0.08680203
23	158	0.802030457	-0.03536379	0.068697124
24	159	0.807106599	-0.007106599	0.040439932
25	168	0.852791878	-0.019458545	0.052791878
26	173	0.878172589	-0.011505922	0.044839255
27	178	0.903553299	-0.003553299	0.036886633
28	187	0.949238579	-0.015905245	0.049238579
29	188	0.954314721	0.012351946	0.020981387
30	190	0.964467005	0.035532995	-0.002199662

From the table generated for LCG output

K+ = 0.1170897

K- = 0.1284264

From KS test table at n=30 and 1-α=0.9

K=0.21756

K+ < K and K- < K hence sequence generated by LCG is random and pass KS test

The output analysis shows that the sequence generated by both LCG and NLCG are random.

D. Runs Test

Run can be defined as a series of increasing values or a series of decreasing values. Length of the run is the number of increasing, or decreasing, values. For starting the runs test median of first thirty elements are found out. If a value in the series is less than median then it is denoted by -1 otherwise +1. Now runs are counted for this series of +1 and -1 and hypothesis testing is done.

Ho : Sequence is random

Ha : Sequence is not random

Table 3. Runs Test Analysis for LCG

Random Series	Value>median +1 else -1
	Median =79.5
37	-1
6	-1
158	1
188	1
168	1
50	-1
63	-1
120	1
82	1
173	1
178	1
109	1
155	1
190	1
101	1
29	-1
77	-1
45	-1
132	1
74	-1
47	-1
65	-1
53	-1
61	-1
187	1
103	1
159	1
56	-1
59	-1
57	-1

From the sequence generated by LCG 30 samples are taken and median is calculated. Median is got as 79.5. Now all the values greater than 79.5 is denoted as +1 and values less than 79.5 as -1. Number of runs is got as 15. Now, n1, number of -1, is 15 and n2 , number of +1 is 15. From runs table the test is passed if the number of runs is between 10 and 22. Here number of runs is 9 and hence the generated sequence is not random.

Table 4. Runs Test Analysis for NLCG

Random Series	Value>median +1 else -1
	Median=96
37	-1
97	1
90	-1
102	1
185	1
97	1
163	1
13	-1
146	1
196	1
188	1
95	-1
33	-1
34	-1
132	1
74	-1
138	1
194	1
164	1
78	-1
37	-1
6	-1
52	-1
120	1
82	-1
67	-1
110	1
23	-1
172	1
40	-1

Here the median is 96, n1 is 15 and n2 is 15. Number of runs is 17 which is between 10 and 22, hence the generated sequence is random. Thus the runs test also proves that the NLCG is better random number generator than LCG.

VII. Conclusion

In this paper Nested Linear Congruential Generator (NLCG) is compared with Linear Congruential Generator (LCG). True random source, prime factorization problem and nesting all together contributes to the enhanced behavior of NLCG. For NLCG the period of the generated sequence is always infinity but for LCG the period depends upon the value of increment and multiplicand chosen. Period is infinity for NLCG since the value of increment and multiplicand is never constant. Statistical and Graphical analysis conducted on the generated sequence proved NLCG to be a better generator when compared to LCG. Nesting hence proves to increase the efficiency of the generator whereby increases the efficiency of the cryptographic algorithm and makes the job of cryptanalyst tough.

VIII. REFERENCES

[1] B. Schneier, “Applied cryptography: protocols, algorithms, and source code in C,” Second Edition, John Wiley & Sons, 1996.

[2] D. Dilli, Madhu S., “Design of a New Cryptography Algorithm using Reseeding - Mixing Pseudo Random Number Generator,” IJITEE, vol.52, No. 5, 2013

[3] K. Marton, A. Suciu, C. Sacarea, and Octavian Cret, “Generation and Testing of Random Numbers for Cryptographic Applications,” Proceedings of the Ramanian Academy, Series A, Vol. 13, No. 4, 2012, PP 368–377.

[4] Wikipedia, “Pseudorandom number generator”, Last visited December 2014.

[5] D. Dilli, and S. Madhu, “Design of a New Cryptography Algorithm using Reseeding - Mixing Pseudo Random Number Generator,” IJITEE, vol. 52, no. 5, 2013.

[6] “True Random Number Generators Secure in a Changing Environment”, Boaz Barak, Ronen Shaltiel, and Eran Tromer, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science , Rehovot, ISRAEL

[7] David DiCarlo, “Random Number Generation: Types and Techniques,” A Senior Thesis submitted in partial fulfillment of the requirements for graduation in the Honors Program Liberty University Spring 2012.

[8] McNichol, Tom (2003-08-11). "Totally Random". Conde Nast Publications. p. 2. Retrieved 2009-10-23. Mads Haahr, a lecturer in computer science at Trinity College in Dublin, designed the system

[9] T. Simul, S.M. Assad, P.K. Lam “Real time demonstration of high bitrate quantum random number generation with coherent laser light”, Appl Phys Lett 98:231103-1-3

[10] Atsushi Uchida, Kazuya Amano, Masaki Inoue, Kunihito Hirano, Sunao Naito, Hiroyuki Someya, Isao Oowada, Takayuki Kurashige, Masaru Shiki, Shigeru Yoshimori, Kazuyuki Yoshimura & Peter Davis, “Fast physical random bit generation with chaotic semiconductor lasers”, Nature Photonics 2, 728 - 732 (2008)

[11] Sunar, B., Martin, W.J., Stinson, D.R. “A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks”, Computers, IEEE Transactions on (Volume:56 , Issue: 1), Jan. 2007, pp. 109 – 119

[12] Hamed Rahimov, Majid Babaie, Hassan Hassanabadi, “Improving Middle Square Method RNG Using Chaotic Map”, Applied Mathematics, 2011, 2, 482-486

[13] Chan, H. “Random number generation”. Retrieved 10/16/2011fromhttp://fuchun00.dyndns.org/~mcmintro/random.pdf, 2009.

[14] Nishimura, T, “Tables of 64-bit mersenne twisters” , ACM Transactions on Modeling and Computer Simulation, 10(4), 348-357, 2000.

[15] Adi A. Maaita, Hamza A. A. Al_Sewadi, “Deterministic Random Number Generator Algorithm for Cryptosystem Keys”, International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol:9, No:4, pp 972-977, 2015

[16] "Linear Congruential Generators" by Joe Bolte, Wolfram Demonstrations Project.

[17] Donald E. Knuth (6 May 2014). Art of Computer Programming, Volume 2: Seminumerical

Algorithms. Addison-Wesley Professional. pp. 4. ISBN 978-0-321-63576-1.

[18] “Testing Random Number Generators”, Dan Biebighauser University of Minnesota - Twin Cities REU Summer 2000

[19] Antu Annam Thomas and Varghese Paul, “Random Number Geneeration Methods a Survey”, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 6, Issue 1, January 2016, pp.556-559

[20] Antu Annam Thomas and Varghese Paul, “Nested Random Number Generator”, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 7, Issue 5, May 2017, pp.767-773