EAI Endorsed Transactions
on Energy Web and Information Technologies
Research Article EAI.EU

Non-Recurring Improved Random Number Generator- a new step to improve cryptographic algorithms

Antu Annam Thomas[1,*] and Varghese Paul[2]

[1]Assistant Professor, Dept. of Computer Application, Mar Thoma College, Thiruvalla
[2]Professor and Head, Dept. of Information Technology, CUSAT University

Abstract

The world is now behind technology transforming every terminology in daily life, like banking, cash, paper, shopping, communication and so on into, e-world terminologies. In this fast growing e-world efficiency of cryptographic algorithms that ensure security of data is of primary concern. Randomness plays an important role in improving the efficiency of cryptographic algorithms. Compared to pseudo Random number generators true random number generators produce random series that exhibit randomness more. But it is better to combine the advantages of both true random number generators and pseudo random number generators. In this work a non-recurring random number generator is being experimented and proved to better generator when compared to the existing systems. In this system it is ensured that a sequence of numbers never repeats within the generated series and also every number occurs with same frequency. Both these qualities of the generated series prove itself to be better than the existing systems.

Keywords: Data security, cryptography, True Random Number Generator, Pseudo Random Number Generator, Kolmogorov Smirnov Test, Runs Test, Prime Number, Scatter Diagram, Bar graph

Received on 12 May 2018, accepted on 01 June 2018, published on 12 June 2018

doi: 10.4108/eai.12-6-2018.154813

*Corresponding author. Email:antu@marthomacollege.org

1. Introduction

Random series refers to a series of numbers where the numbers are independent and cannot reliably be reproduced. Random number generation techniques are used in cryptographic systems where randomness plays an important and major role. Natural phenomena like radioactive decay, tossing of coin are all examples of ideal random number generators also called true random number generators. That is true random number generators are those generators that uses physical entropy sources for their processing. Another type of random number generators called pseudo random number generators uses computational algorithms for their generation. Since pseudo random number generators uses computational algorithms for generation, generated series though seems to be truly random are not truly random [1][2][3].

True random number generators also called hardware random number generator, since hardware is required to extract entropy values from true physical sources. But this type of generator produce limited number of random bits per second. In order to increase the bit rate we usually go for pseudo random number generator algorithms.

Linear Congruential Generator, Blum Blum Shub Generator, Linear Feedback Shift Register, XORShift Generators are some of the traditional pseudo random number generators [4].

Though there are limitations for true random number generators the random series generated by them are perfectly random. Thus we cannot neglect its perfect performance.

In order to get over the disadvantages of true random number generators it is better to combine true random

number generator and pseudo random number generator. True random number generators are used for generating seed values for pseudo random number generators [5].
In the proposed algorithm true random number generator is combined with pseudo random number generator. Concept of nesting is also used while generation so that a sequence of numbers never repeats in the generated series. Along with nesting recurrence tests are done so that it is ensured that every number occurs with equal frequency in the generated series.

2. Related works

A true random number generator generates series of numbers that are unpredictable, independent and uniform. Source of entropy is some real world events such as flipping coin and rolling of dice. Real world events have a high degree of entropy, because it is almost impossible to predict accurately what the final result will be. Thus it is the source of entropy that helps a true random number generator to generate a perfectly random series [6].

Random.org, Hotbits, lasers, and oscillators are popular true random number generators [7].

Random.org which is a website created in 1998 by Mads Haahr produces "true" random numbers based on atmospheric noise captured by several radios tuned between stations [8].

HotBits uses inherent uncertainty in the quantum mechanical laws of nature, for generation of series.

Broadband measurements of the vacuum field contained in the radio-frequency sidebands of a single-mode laser can also be exploited for random number generation [9].

Using physical chaos in semiconductor lasers or sampling phase jitter in oscillator rings good quality random bit sequence can be generated [10][11].

Random number generators that use computational algorithms for series generation is called pseudo random number generator. There are varieties of pseudo random number generators. Linear Congruential method, Middle-square method, Linear Feedback Shift Registers, Xorshift generators are some of the early Pseudo Random Number Generator [12][13][14][17].

3. Proposed System

The proposed system called Non-Recurring Improved Random Number Generator produces a random number series that promises unpredictability and uniformity. Fact that prime factorization is difficult is exploited in this algorithm. The true random source used during seed generation and series generation further increases system's efficiency. Since nesting is done generated series not only depends on its previous value but also a nested random series value. Nesting ensures that sequence is never repeated within the final series being generated The generated random series then goes through recurrence test. After this step occurrence of every numbers in the series are ensured to be uniformly distributed.

From the above discussion it is clear that the proposed system though it uses a computational algorithm for generation produces a series that will be independent, unpredictable and uniformly distributed.

3.1 Methodology

The proposed system consist of three steps

1. Getting the seed value
2. Generating the series
3. Refining the series

Getting the seed value

For generating the seed value first the system clock value is read. Based on the read clock value pixel value of the current picture captured by system camera is read. Thus here two true random values are used for seed generation system clock value and pixel value of the image.

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.

Now seed value is given by,

$$x_0=p1_0*p2_0 \bmod m \qquad (1)$$

Thus seed value x_0 is the product of two prime numbers.

The complexity of the seed is thus due to two factors

1. Difficulty in factorizing product of two prime numbers.
2. 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.

Generating the series

$$x_i=(x_{i-1}*b_i+a_i) \bmod m \qquad (2)$$

$$b_i=f_i*p1_{i-1}*p2_{i-1} \qquad (3)$$

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 $p2_i$.

$$a_i=p1_i*p2_i \qquad (4)$$

Equations (2), (3) and (4) together generate the random number series. The generated series goes through refining step later.

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.

Refining the series

The series while being generated goes through another refining step also. Here the generated value is checked and it is ensured that the values occur uniformly.

Nesting together with refining increases the efficiency of the system and makes it truly non-recurring.

3.2 Algorithm

Step 1. System clock time is read

Step 2. Based on the read system clock a pixel value is read from the current image captured using system camera.

Step 3. $p1_0$ and $p2_0$ are the first pair of prime numbers generated based on the pixel value read

Step 4. Seed value is generated using equation (1)

Step 5. Equations (2), (3) and (4) together generate the next value in the series.

Step 6. The generated value undergoes recurrence testing.

Step 7. Step 5 and Step 6 is repeated till required number of random numbers are generated.

Flowchart for the algorithm mentioned above is given below.

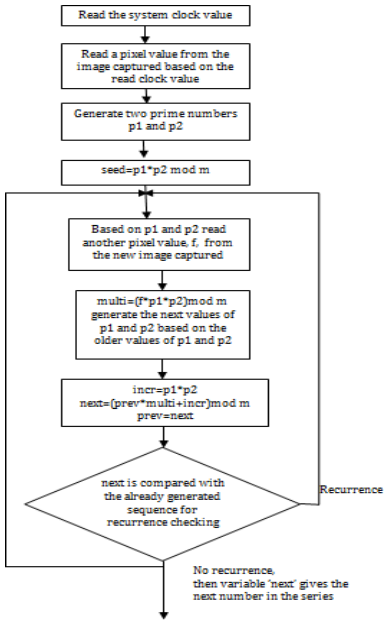

Figure 1. Flowchart of the proposed system

3.3 Advantages of the Proposed System

1. Seed value generated is based on true entropy sources, system clock value and value of a pixel selected based on system clock value read. Product of two prime numbers generated based on this pixel value read forms the seed value. The seed value generated in this manner ensures that it is truly random. The prime factorization complexity makes the seed value more random.
2. In the generated random number series a sequence is never repeated.
3. Nested concept used during series generation makes the series unpredictable and random.
4. The series is further refined which ensures uniformity for the sequence [6][15].

3.4 Complexities involved in this method

1. System clock time is used as the source of true entropy both during seed generation and series generation
2. The pixel value read from the current image being captured based on the system clock value adds to the random behavior of generated series.
3. Product of two prime numbers forms the seed value. The seed value is thus difficult to be factorized making cryptanalysis practically impossible.
4. Nesting of algorithm increases the randomness of the generated sequence.
5. Refining makes the series more uniform.

4 Implementation

The algorithm was implemented in Matlab and output was got as expected.
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 (1).
Now the series is generated using the equations (2), (3) and (4).
The output got after implementing the algorithm is given below.

'm' was chosen to be 197.

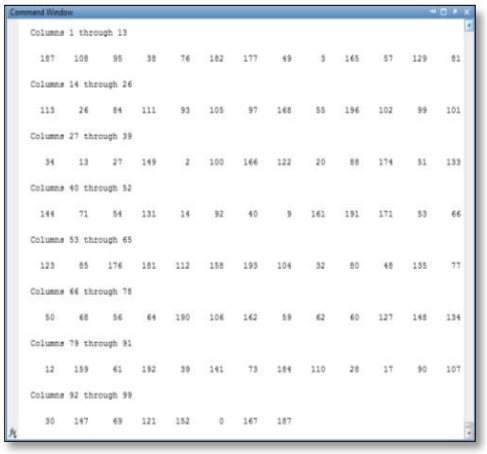

Figure 2 Output showing the first 99 elements in the random series

5 Result Analysis

5.1 Scatter Diagram Analysis

Scatter diagram of the random series generated are given below. From the diagram analysis it can be seen that the generated series posses good randomness and that there is no linear relationship or correlation between the generated random values.

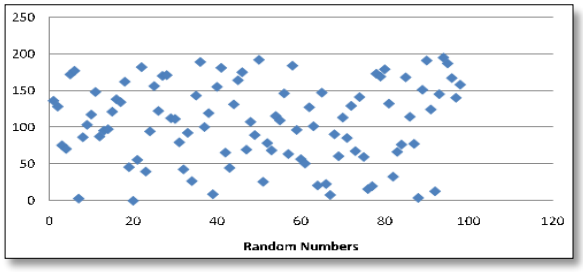

Figure 3 Scatter Diagram

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

As in the figure below since the value of Q is greater than the limit given in Trend Test Table it is proved that the pattern has occurred from random chance.

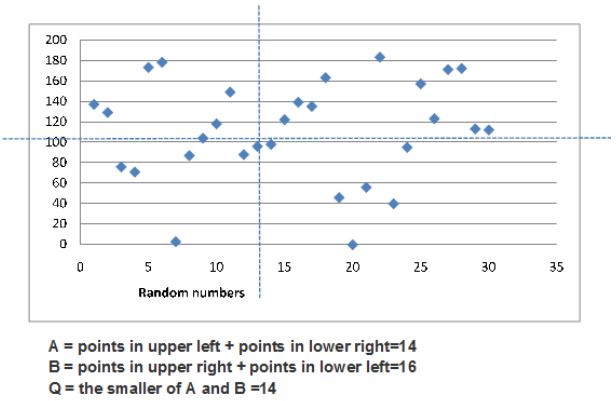

Figure 4 Bar Graph Analysis

5.2 Bar Graph Analysis

The bar graph plotted for the random values generated are given below. The graph shows that every value occur with same frequency or random series posses the property of uniformity.

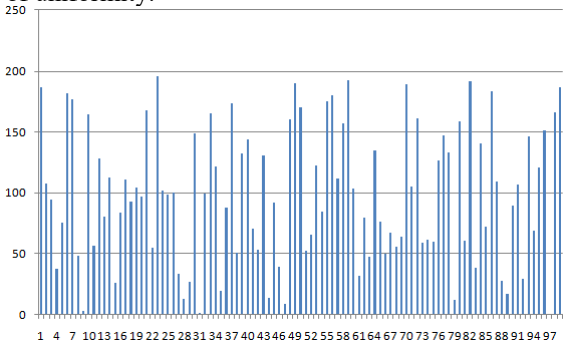

Figure 5 Bar Graph

5.3 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].

H_0 = Sequence being tested is random

H_a = Sequence being tested is not random

From the table given below

$K+ = 0.071066$

$K- = 0.208291$

From KS test table at n=30 and $1-\alpha=0.9$

$K=0.21756$

$K^+ < K$ and $K^- < K$ hence sequence is random and pass KS test.

Table 1. K S Test Analysis

j	random	normalised	(j/n-xj)	xj-(j-1)/n
1	0	0	0.033333333	0
2	3	0.015228426	0.05143824	-0.018104907
3	40	0.203045685	-0.103045685	0.136379019
4	46	0.233502538	-0.100169205	0.133502538
5	56	0.284263959	-0.117597293	0.150930626
6	71	0.360406091	-0.160406091	0.193739425
7	76	0.385786802	-0.152453469	0.185786802
8	87	0.441624365	-0.174957699	0.208291032
9	88	0.446700508	-0.146700508	0.180033841
10	95	0.482233503	-0.148900169	0.182233503
11	96	0.487309645	-0.120642978	0.153976311
12	98	0.497461929	-0.097461929	0.130795262
13	104	0.527918782	-0.094585448	0.127918782
14	112	0.568527919	-0.101861252	0.135194585
15	113	0.573604061	-0.073604061	0.106937394
16	118	0.598984772	-0.065651438	0.098984772
17	122	0.61928934	-0.052622673	0.085956007
18	123	0.624365482	-0.024365482	0.057698816
19	129	0.654822335	-0.021489002	0.054822335
20	135	0.685279188	-0.018612521	0.051945854
21	137	0.695431472	0.004568528	0.028764805
22	139	0.705583756	0.027749577	0.005583756
23	149	0.756345178	0.010321489	0.023011844
24	157	0.796954315	0.003045685	0.030287648
25	163	0.827411168	0.005922166	0.027411168
26	171	0.868020305	-0.001353638	0.034686971
27	172	0.873096447	0.026903553	0.00642978
28	173	0.878172589	0.055160745	-0.021827411
29	178	0.903553299	0.063113367	-0.029780034
30	183	0.92893401	0.07106599	-0.037732657

Random Series	Value>median +1 Median =115.5
137	1
129	1
76	-1
71	-1
173	1
178	1
3	-1
87	-1
104	-1
118	1
149	1
88	-1
96	-1
98	-1
122	1
139	1
135	1
163	1
46	-1
0	-1
56	-1
183	1
40	-1
95	-1
157	1
123	1
171	1
172	1
113	-1
112	-1

5.4 Runs Test

A series of increasing values or a series of decreasing values is known as a run.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.

H_o : Sequence is random

H_a : Sequence is not random

From the series generated sequence 30 samples are taken and median is calculated and found to be 115.5. Now all the values greater than 115.5 is denoted as +1 and values less than 115.5 as -1. Number of runs is got as 12. Now, n_1 , number of -1, is 15 and n_2 , 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 12 hence the generated series passes the test and is sufficiently random.

Table 2 Runs Test Analysis

6 Conclusion

Non-Recurring improved random number generator is a novel approach in random series generation. Concept of nesting and further tuning of the generated series makes cryptanalysis of the series difficult or rather impossible. The graphical and statistical tests proved the system to be efficient.

Complexity of Prime factorization also contributes to the efficiency of the system.

The true random sources used in the proposed system are system clock value and random pixel value read from image captured. The efficiency of the system can be further increased by using more complicated hardware for extracting true random numbers

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 10/16/2011from http://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] “Testing Random Number Generators”, Dan Biebighauser University of Minnesota - Twin Cities REU Summer 2000

[17] 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

[18] 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.767-77