Generation of Pseudorandom Sequence Using Regula-Falsi Method

Proceedings of the Sixth International Conference on Mathematics and Computing pp 393-405 | Cite as

- Aakash Paul (1) Email author (aakash18iiest@gmail.com)
- Shyamalendu Kandar (1)
- Bibhas Chandra Dhara (2)

1. Department of Information Technology, Indian Institute of Engineering Science And Technology, , Howrah, India
2. Department of Information Technology, Jadavpur University, , Kolkata, India

Conference paper
First Online: 11 December 2020

Part of the Advances in Intelligent Systems and Computing book series (AISC, volume 1262)

Abstract

Pseudorandom number generator (PRNG) generates a sequence of numbers whose properties approximate the properties of sequences of random numbers. The sequence is not truly random as it can be regenerated by some initial values called seed. Pseudorandom sequence has a wide range of applications in science and engineering like modeling and simulation, encryption, gambling, gaming, etc. Chaos theory has established itself a good choice for pseudorandom sequence generation due its intrinsic properties like ergodicity, sensitivity to initial condition, etc. Several non-chaotic methods have also established with dignity for pseudorandom number generation. In this paper, we have proposed a non-chaotic method for pseudorandom sequence generation. Regula-Falsi method is used as the backbone for the said. NIST randomness test and several other tests have proved the randomness of the generated sequence and have established it as a suitable alternative for pseudorandom sequence generation.

Keywords

Pseudorandom sequence Regula-Falsi method Non-chaotic NIST randomness test

1 Introduction

Random sequence is preferred in the areas where unpredictable result is necessary. It has a wide region of applications like modeling and simulation, gaming, randomized design, statistical sampling, etc. These types of sequences cannot be reasonably predicted. But in the number of fields like cryptography, randomized algorithm, etc., a type of random sequence is preferred which can be regenerated by multiple agents. These types of sequences are known as pseudorandom sequence. These sequences are generated through a particular mathematical formula with some initial conditions known as seed, and given the same initial conditions, the same sequence is generated every time. The difference between truly random number sequence (TRNS) and pseudorandom number sequence (PRNS) is that TRN generator uses an unpredictable physical means to generate numbers (like atmospheric noise), and PRNS generator uses mathematical algorithms to generate the sequence. One major difference between TRNS and PRNS is that TRNS has infinite length of periodicity, whereas PRNS has a sufficiently large length of periodicity.

Earlier the output of linear feedback shift register (LFSR) was widely used for pseudorandom sequence generation [1]. Wolfram [2] has used connected cellular automata (CA) to generate pseudorandom sequence. In [3], a VLSI implementation of parallel cellular automata is used which generate better and faster random sequence than LFSR-based system due to its nearest neighbor wiring. Martin [4] in his research article has shown that a system with multiple LFSR with different seed values produces more random sequence than single LFSR. Hardware-based pseudorandom sequence generator is complex and time taking for designing. Chaos theory soon takes the place of pseudorandom sequence generation due to its properties like ergodicity, large periodicity, sensitivity to initial condition [5].

In an earlier research article [6], chaotic logistic map is used to generate pseudorandom number generation. In [7], two chaotic logistic maps with different random seed values are used to generate pseudorandom bit sequence. Some different types of logistic map like digitized modified logistic map [8], piecewise logistic map [9], enhanced logistic map [10], etc. are used to generate pseudorandom sequence.

Based on the properties of tent map with the cooperation of fair coin tossing model, two pseudorandom bit generators are addressed in [11]. A compounded tent map consisting of a couple of tent maps with different parameters are applied in [12] to produce pseudorandom bit sequence. Some recent proposals [13, 14, 15] of pseudorandom number generation using tent map have also attracted researchers' attention.

Several other chaotic maps like Chen chaotic system [16], spatiotemporal chaotic map [17, 18], Tinkerbell maps [19], sine logistic map [20, 21], Arnold cat map [22, 23] etc. are well accepted for pseudorandom sequence generation.

Chaotic map is familiar to the research communities due to its intrinsic properties mentioned earlier. Researchers have proposed some non-chaotic techniques to define pseudorandom sequence and test results have established good even better results in comparison with chaotic map-based techniques. A balanced gray code based pseudorandom sequence generator is available in [24]. The generators are received from the iteration of Boolean maps, computed from balanced gray code. A number of PRNG [25, 26, 27] have been derived from Fibonacci series. In [28], the motion path of a billiards ball on a circular billiards board has been used to generate pseudorandom sequences. Other non-chaotic methods such as pendulum simulations [29], bouncing balls simulations [30], cellular neural network simulations [31], etc. are available in literature.

In this article, we have used Regula-Falsi method to generate pseudorandom sequence. Regula-Falsi method is used for finding roots of a polynomial *f*. In this technique, we have taken two points *a* and *b* as seed values such that $f(a) * f(b) < 0$. A straight line is drawn between $(a, f(a))$ and $(b, f(b))$. This line intersects the X-axis at a point let us say x_{new}. The fraction part of $f(x_{\text{new}})$ is used to generate the entry in the sequence and also used for deriving new iteration by modulo operation. NIST randomness test [32] and several other tests like entropy analysis, histogram analysis, key sensitivity, etc. provide a strong base for pseudorandom sequence generation.

The rest of the paper is organized as follows. The conventional Regula-Falsi method is discussed in Sect. 2. Section 3 describes the proposed technique of pseudorandom sequence generation. Experimental results are displayed in Sect. 4. The randomness test and security analysis are performed in Sect. 5. Finally, conclusion is drawn in Sect. 6.

2 Regula-Falsi Method

The Regula-Falsi or False position method is a well-known technique for finding the root of a polynomial *f*(*x*). In this method, two points *a* and *b* on *X*-axis, are taken in such a way that $f(a) * f(b) < 0$. A straight line is drawn between $(a, f(a))$ and $(b, f(b))$, which intersects the X-axis at a point $(x_{\text{new}}, 0)$. This gives an improved estimation of the root and is called the false position. The newly received point x_{new} replaces one of the initial guesses which has the same sign as $f(x_{\text{new}})$. By this way, the process iterates and narrows down the possible range for the value of the root. The process terminates when the value $f(x_{\text{new}})$ is less than some acceptable margin of error τ close to 0.

The Regula-Falsi method is presented in Algorithm 1.

Algorithm 1: *Regulafalsi*$(f(x), a, b, \tau)$

Input: The polynomial function $f(x)$, Two initial points a and b s.t. $f(a) * f(b) < 0$, Predefined threshold value τ

Output: Estimated root x_{new}

while *true* **do**

  Draw a straight line S between $(a, f(a))$ and $(b, f(b))$

  $x_{\text{new}} = a - \frac{f(a)(b-a)}{f(b)-f(a)}$ `// intersection pt. of S and X-axis`

  if $|f(x_{\text{new}})| \leq \tau$ **then**

    break

  if $f(x_{\text{new}}) * f(a) < 0$ **then**

    $b = x_{\text{new}}$

  else

    $a = x_{\text{new}}$

return x_{new}

The Regula-Falsi method is described using the following example. A polynomial $f(x) = 4((x-8)^2 - 4)$ is taken with two points $a = 5$ and $b = 9$ ($f(5) * f(9) = -240$). The estimated roots approach the original roots in the further iterations. This is depicted in Fig 1.

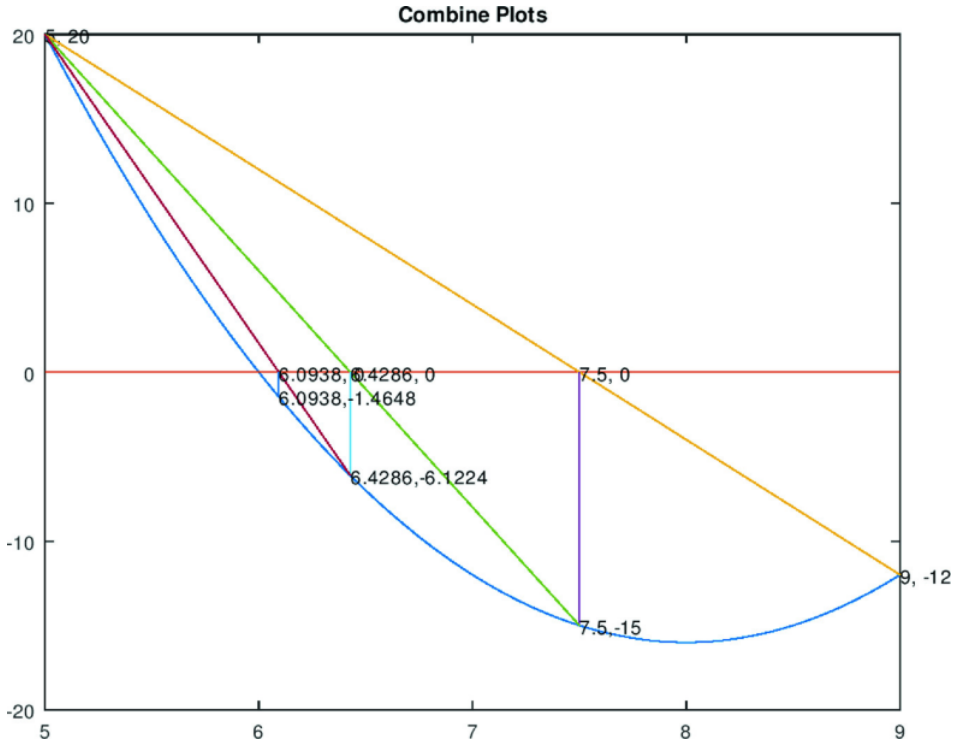


Fig. 1

A plot showing convergence toward root in Regula-Falsi method

3 Proposed Technique of Pseudorandom Sequence Generation

Figure 1 clearly signifies that Regula-Falsi method converges rapidly toward the root. In this paper, we have proposed a new technique for pseudorandom sequence generation using modified Regula-Falsi method.

Like the standard Regula-Falsi method, the modified algorithm takes as input a polynomial $f(x)$ of degree two or above and two seed points a and b such that $f(a) \times f(b) < 0$. Other requisite inputs are the length of the pseudorandom sequence L and the number of bits in each term of the sequence n.

In the first iteration, a straight line is drawn between the points $(a, f(a))$ and $(b, f(b))$, which intersects the X-axis at the point $(x_{\text{new}}, 0)$. The integer value $(\lfloor f(|x_{\text{new}}|) \times 10^5 \rfloor) \bmod 2^n$ corresponds to the first item in the sequence, whose value lies in the range $(0, 2^n - 1)$. Alternatively, it can be said that the 1st n bits of the pseudorandom sequence are generated.

The algorithm needs to generate a pair of new pair of points in order to draw a straight line in the subsequent iteration. The integer value $m = \lfloor (|f(x_{\text{new}})| * 10^3) \rfloor \bmod 4$ is calculated. A pair of real values m_1 and m_2 are calculated, while being conditionally dependant on the value of m. This pair of real values is calculated using the value $|f(x_{\text{new}})| - \lfloor |f(x_{\text{new}})| \rfloor$, i.e., the fractional part of real number $|f(x_{\text{new}})|$. The next straight line is drawn between $(a, m_1 * f(a))$ and $(b, m_2 * f(b))$. The process iterates L times to generate L length pseudorandom sequence each of n bit. This is illustrated in Algorithm 2.

Algorithm 2: $prsg_{rf}(f(x), a, b, n, L)$

Input: The polynomial $f(x)$, Two initial points a and b, Bit length of each term n, Length of the sequence L

Output: L length pseudo random sequence Seq, with each term of n bits

$Seq = \mathbf{0}$

Draw a straight line S between $(a, f(a))$ and $(b, f(b))$

$x_{\text{new}} = a - \frac{f(a)(b-a)}{f(b)-f(a)}$ // intersection pt. of S and X-axis

for $k \leftarrow 1$ *to* L **do**

$Seq[i] = (floor(|f(x_{\text{new}}| * 10^5)\ mod\ 2^n)$

$m = (floor(|f(x_{\text{new}})| * 10^3)\ mod\ 4)$

if $m = 0\ OR\ m = 3$ **then**

$m_1 = (1 - (|f(x_{\text{new}})| - \lfloor |f(x_{\text{new}})| \rfloor))^{-1}$

$m_2 = (1 + (|f(x_{\text{new}})| - \lfloor |f(x_{\text{new}})| \rfloor))^{-1}$

else

$m_1 = (1 + (|f(x_{\text{new}})| - \lfloor |f(x_{\text{new}})| \rfloor))^{-1}$

$m_2 = (1 - (|f(x_{\text{new}})| - \lfloor |f(x_{\text{new}})| \rfloor))^{-1}$

Draw line segment S between $(a, m1 * f(a))$ and $(b, m2 * f(b))$

$x_{\text{new}} = a - \frac{m_1 * f(a)(b-a)}{m_2 * f(b) - m_1 * f(a)}$ // (intersection pt. of S and X-axis)

return *Seq*

In Algorithm 2 the value of m_1 and m_2 changes $f(a)$ and $f(b)$. Drawing straight line between $(a, m_1 * f(a))$ and $(b, m_2 * f(b))$ will intersect the X-axis between $(a, 0)$ and $(b, 0)$. A diagrammatically representation of the sequence generation process is presented in Fig. 2 with the polynomial $f(x) = 5x^3 - 30x$, with $a = -1.30001$ and $b = 1.60001$. The corresponding values of m_1, m_2, x_{new} and the terms in the sequence upto third iteration are listed in Table 1 (for n = 8).

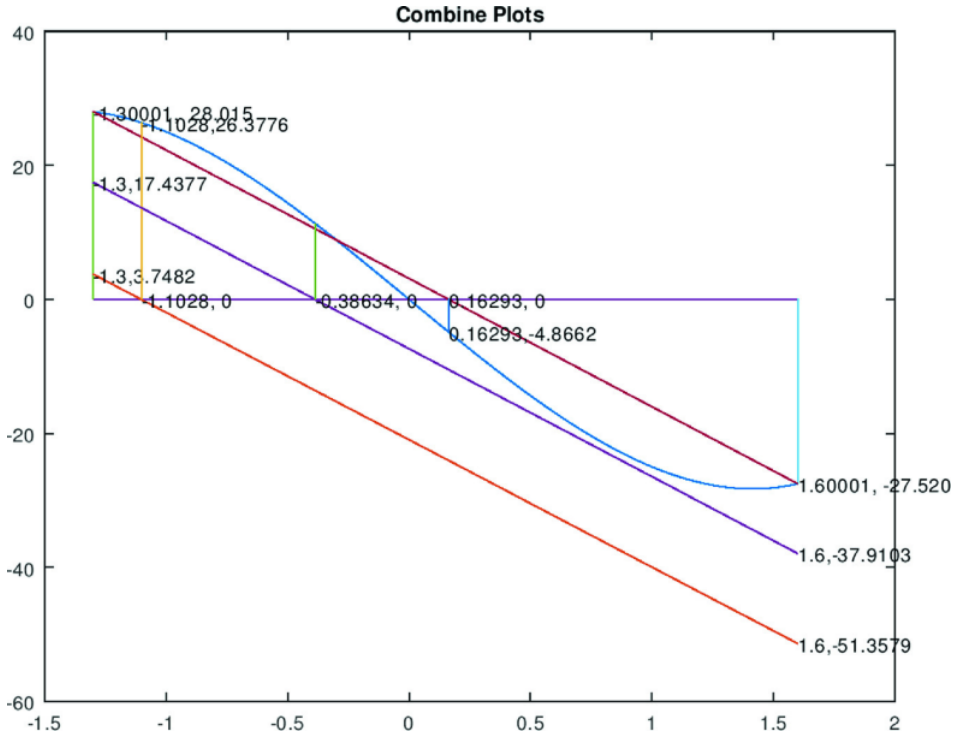


Fig. 2

A plot showing successive steps in proposed method

Table 1

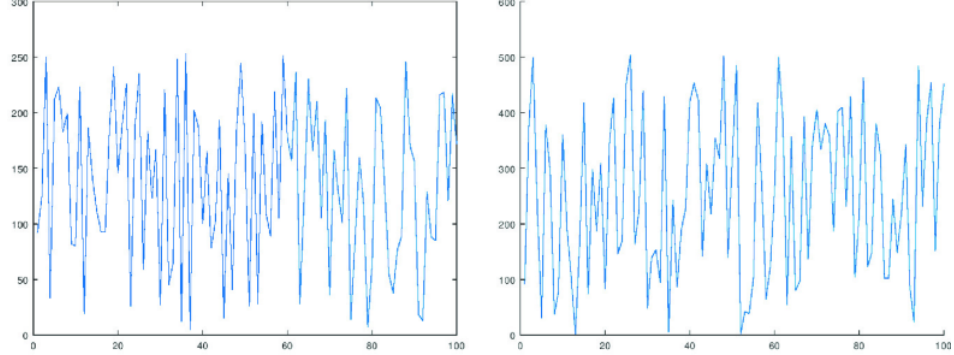
The values generated by the first three iterations of the algorithm, based on Key K1 in Table 3

Iteration no. (*i*)	m_1	m_2	**Intersection pt.** $x_{\text{new}} f(x_{\text{new}})$	***i*th term of sequence**
1	1	1	0.16293 −4.8662	92
2	0.13379	1.8662	−1.1028 26.37756	123
3	0.62244	1.3776	−0.38634 11.30195	243

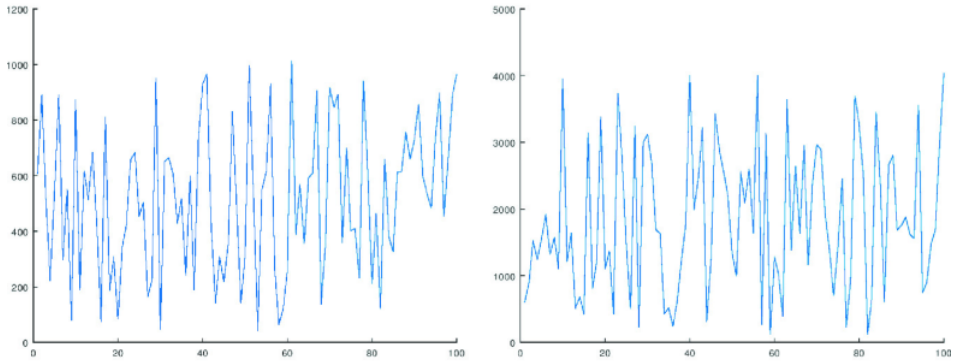
4 Experimental Results

For experimental purpose, we have taken a polynomial $5x^3 - 30x$ with initial seed points $a = -1.30001$, $b = 1.60001$. Algorithm 2 is executed 2×10^5 times (to generate sequence of length 2×10^5) with different bit length n as 8, 9, 10, and 12. First 100 values of each cases are plotted against their respective iteration-number and are presented in Fig 3a–d. The random occurrence of picks and valleys signifies the randomness of the generated sequence for each case.

To have a better understanding, histograms are drawn with the full sequence (2×10^5) for each case. This is presented in Fig 4. From the histograms, it is clear that there are not so much variations in the frequency of occurrence of each integer value.



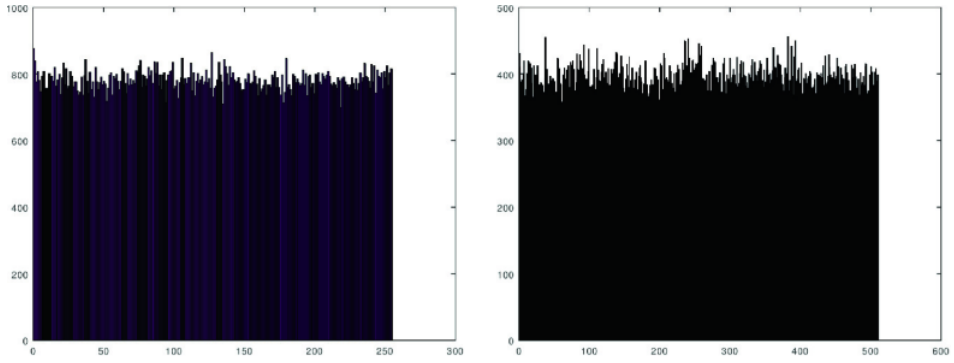
(a) Plot of 8-bit values (b) Plot of 9-bit values



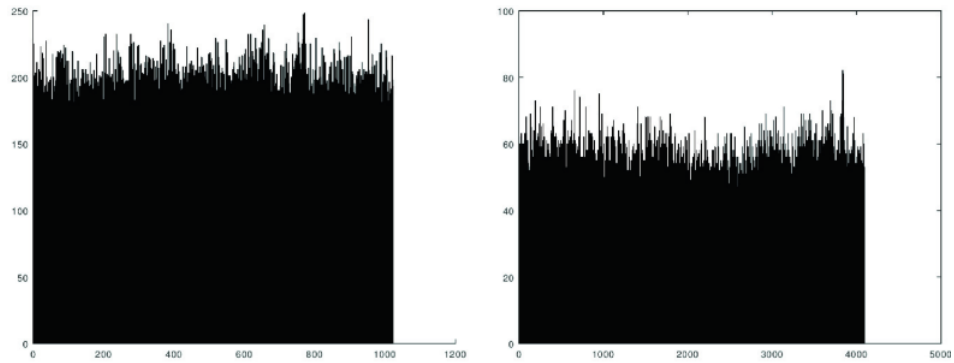
(c) Plot of 10-bit values (d) Plot of 12-bit values

Fig. 3

A plot of the first 100 terms generated by modified Regula-Falsi method with **a** n = 8, **b** n = 9 **c** n = 10 and **d** n = 12



(a) Histogram of 8-bit values (b) Histogram of 9-bit values



(c) Histogram of 10-bit values (d) Histogram of 12-bit values

Fig. 4

Histograms of 2×10^5 length sequence generated using modified Regula-Falsi method with **a** n = 8, **b** n = 9 **c** n = 10 and **d** n = 12

A plot of the first 100 8-bit, 9-bit, 10-bit, and 12-bit integers generated by modified Regula-Falsi method using Key K1 from Table 3 is shown in Fig. 3.

5 Randomness Test and Security Analysis

In this section, the randomness test of the generated sequence using Algorithm 2 is performed using NIST randomness test. The security analysis covers key space analysis, information entropy analysis, and key sensitivity analysis.

5.1 NIST Randomness Test

NIST randomness test is a preferable choice for testing the randomness of some sequence. This is a package of 15 different tests developed by National Institute of Standard and Technology. These tests are applied on binary sequences and the output is received in the form of p-value. 'p-value' denotes the probability that a perfect random number generator would produce less random sequences than the sequence being tested. Detail about NIST randomness test can be found in [32]. For testing the randomness using NIST test a binary sequence of length 5×10^7 is generated using Algorithm 2 with $f(x) = 5x^3 - 30x$, $a = -1.30001$, $b = 1.60001$. The results of NIST Randomness test are tabulated in Table 2. The generated sequence has passed 13 out of 15 tests as shown in Table 2. From this result, it can be said that the proposed method generates pseudorandom sequences.

Table 2

Results of NIST Randomness Test of the chaotic sequence generated by proposed method using key K1 from Table 3

Test name	*p*-value	Success rate	Result
Frequency	0.171867	99/100	Pass
Block frequency	0.897763	99/100	Pass
Cumulative sums	0.122325	98/100	Pass
Runs	0.946308	99/100	Pass
Longest runs	0.657933	99/100	Pass
Rank	0.574903	98/100	Pass
FFT	0.366918	99/100	Pass
Non overlapping template	0.935716	100/100	Pass
Overlapping template	0.023545	96/100	Pass
Universal	0.162606	99/100	Pass
Approximate entropy	0.000000	13/100	Fail
Random excursions	0.474986	51/51	Pass
Random excursions variant	0.514124	51/51	Pass
Serial	0.000000	80/100	Fail
Linear complexity	0.191687	100/100	Pass

5.2 Security Analysis

Cryptography is one of the major application field of pseudo random sequence. A good pseudo random sequence must carry some security features which are analyzed by some standard techniques like key space analysis, information entropy analysis, key sensitivity analysis etc. These are described in the following.

Key Space Analysis Keys are the trivial part for generating the pseudorandom sequence. If the keys are managed, the sequence can easily be generated. Attacker uses brute force technique to guess the keys and limited key space makes the job easy. For a good pseudorandom sequence, the key space should be large enough to resist brute force attack. In this technique the key space of the method is defined by the polynomial functions and the initial seed values *a* and *b*. If the polynomial is of degree *d* with *k* term system, it can be represented by $(d+1)$ coefficients or by *k* pairs of coefficient and exponent [i.e., $(\text{cof}_i; \exp_i)$; $1 \le i \le k$]. If 64-bit floating point is considered to represent the terms, the key space will be $(d+1+2) \times 64$ bits or $(2k+2) \times 64$ bits ('+2' is for '*a*' and '*b*'). Having such a huge key space it becomes difficult to guess about *f*(*x*) and seed points.

Key Sensitivity Analysis A good pseudorandom sequence generation algorithm should produce a completely different sequence with a small change of the key-a little deviation from the original. Key sensitivity analysis is measured by the percentage of change of terms of the two different sequences having a little deviation of the key.

Table 3

Details of secret keys

Serial no.	Key id.	Polynomial *f*(*x*)	a	b
1	K1	$5x^3 - 30x$	−1.30001	1.60001
2	K2	$5x^3 - 30x$	−1.30001001	1.60001001

For testing purpose, we have generated two sequences seq_1 and seq_2 each of length $L = 10^4$ using the same polynomial $5x^2 - 30x$ but two different keys (with a little change) as given in Table 3. This is measured using number of term change rate (NTCR) as given in Eq. 1. For ideal case, the value of NTCR is 100 (i.e., all the terms are changed).

The result is presented in Table 4 for different values of *n*. For all of the cases, the NTCR value is close to the ideal value from which it can be said that the proposed method generates different sequences with a slide change of the key.

$$NTCR(\text{seq}_1, \text{seq}_2) = \frac{1}{L} * \sum_{i=1}^{L} E(i) * 100$$
$$E(i) = \begin{cases} 1, & \text{if } \text{seq}_1(i) = \text{seq}_2(i) \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

Table 4

NTCR of a pair of 10^4 term sequences for different bit length(n)

Serial No.	Bits per iteration n	NTCR
1	8	99.74
2	9	99.85
3	10	99.93
4	11	99.98

5.3 Information Entropy Analysis

Information entropy of a sequence seq of length L, with each term of n is given by Eq. 2

$$H(\text{seq}) = \frac{1}{2^n} \sum_{i=0}^{2^n-1} p(i) \log_2 \frac{1}{p(i)} \tag{2}$$

Here, $p(i)$ is the probability that a term in the sequence has value i.

The ideal value for information entropy is n for a sequence with each term of bit length n.

Four distinct sequences of length 10^5 terms, each using polynomial $f(x) = 30x^3 - 5x$, with $a = -1.30001$, $b = 1.60001$, and having bit length n as 8, 9, 10, and 12, respectively, are used for the analysis.

The information entropy of the four sequences is presented in Table 5. For each of the cases, the value of information entropy is closer to bit length. Thus, the result is acceptable.

Table 5

The information entropy of 8-bit, 9-bit, 10-bit, and 12-bit values generated by the proposed method

Serial no.	Bits per iteration n	Information entropy
1	8	7.9980
2	9	8.9961
3	10	9.9920
3	12	11.969

6 Conclusion

In this paper, we have proposed a pseudorandom sequence generation using modified Regula-Falsi method. Its performance has been analyzed through NIST Randomness test and other metrics. Still, further work is possible which could lead to even better results on these metrics.

References

1. Knuth D (1981) The art of computer programming, vol 2 (Seminumerical Algorithms).” Addison-Wesley
Google Scholar (https://scholar.google.com/scholar?q=Knuth%20D%20%281981%29%20The%20art%20of%20computer%20programming%2C%20vol%202%20%28Seminumerical%20Algorithms%29.%E2%80%9D%20Addison-Wesley)
2. Wolfram S (1986) Random sequence generation by cellular automata. Adv Appl Math 7:123–169
MathSciNet (http://www.ams.org/mathscinet-getitem?mr=845373)
CrossRef (https://doi.org/10.1016/0196-8858(86)90028-X)
Google Scholar (http://scholar.google.com/scholar_lookup?title=Random%20sequence%20generation%20by%20cellular%20automata&author=S.%20Wolfram&journal=Adv%20Appl%20Math&volume=7&pages=123-169&publication_year=1986)
3. Hortensius P, McLeod R, Card H (1989) Parallel random number generation for VLSI systems using cellular automata. IEEE Trans Comput 38(10):1466–1473 Oct
CrossRef (https://doi.org/10.1109/12.35843)
Google Scholar (http://scholar.google.com/scholar_lookup?title=Parallel%20random%20number%20generation%20for%20VLSI%20systems%20using%20cellular%20automata&author=P.%20Hortensius&author=R.%20McLeod&author=H.%20Card&journal=IEEE%20Trans%20Comput&volume=38&issue=10&pages=1466-1473&publication_year=1989)
4. Martin P (2002) An analysis of random number generators for a hardware implementation of genetic programming using FPGAs and Handel C. In: GECCO 2002: proceedings of the genetic and evolutionary computation conference, pp 837–844
Google Scholar (https://scholar.google.com/scholar?q=Martin%20P%20%282002%29%20An%20analysis%20of%20random%20number%20generators%20for%20a%20hardware%20implementation%20of%20genetic%20programming%20using%20FPGAs%20and%20Handel%20C.%20In%3A%20GECCO%202002%3A%20proceedings%20of%20the%20genetic%20and%20evolutionary%20computation%20conference%2C%20pp%20837%E2%80%93844)
5. Kandar S, Dhaibat C, Bhattacharjee A, Dhara BC (2019) Image encryption using sequence generated by cyclic group. J Inf Secur Appl 44 117–129 (2019)
Google Scholar (https://scholar.google.com/scholar?q=Kandar%20S%2C%20Dhaibat%20C%2C%20Bhattacharjee%20A%2C%20Dhara%20BC%20%282019%29%20Image%20encryption%20using%20sequence%20generated%20by%20cyclic%20group.%20J%20Inf%20Secur%20Appl%2044%20117%E2%80%93129%20%282019%29)
6. Phatak S, Rao S (1995) Logistic map: a possible random number generator. Phys Rev E 51(4):3670–3678
CrossRef (https://doi.org/10.1103/PhysRevE.51.3670)
Google Scholar (http://scholar.google.com/scholar_lookup?title=Logistic%20map%3A%20a%20possible%20random%20number%20generator&author=S.%20Phatak&author=S.%20Rao&journal=Phys%20Rev%20E&volume=51&issue=4&pages=3670-3678&publication_year=1995)
7. Patidar V, Sud KK, Pareek NK (2009) A pseudo random bit generator based on chaotic logistic map and its statistical testing. Informatica 33.4
Google Scholar (https://scholar.google.com/scholar?q=Patidar%20V%2C%20Sud%20KK%2C%20Pareek%20NK%20%282009%29%20A%20pseudo%20random%20bit%20generator%20based%20on%20chaotic%20logistic%20map%20and%20its%20statistical%20testing.%20Informatica%2033.4)

8. Chen Shih-Liang, Hwang TingTing, Lin Wen-Wei (2010) Randomness enhancement using digitalized modified logistic map. IEEE Trans Circuits Syst II: Exp Br 57(12):996–1000
CrossRef (https://doi.org/10.1109/TCSII.2010.2087951)
Google Scholar (http://scholar.google.com/scholar_lookup?title=Randomness%20enhancement%20using%20digitalized%20modified%20logistic%20map&author=Shih-Liang.%20Chen&author=TingTing.%20Hwang&author=Wen-Wei.%20Lin&journal=IEEE%20Trans%20Circuits%20Syst%20II%3A%20Exp%20Br&volume=57&issue=12&pages=996-1000&publication_year=2010)

9. Wang Yong, Liu Zhaolong, Ma Jianbin, He Haiyuan (2016) A pseudorandom number generator based on piecewise logistic map. Nonlinear Dyn 83(4):2373–2391
MathSciNet (http://www.ams.org/mathscinet-getitem?mr=3457749)
CrossRef (https://doi.org/10.1007/s11071-015-2488-0)
Google Scholar (http://scholar.google.com/scholar_lookup?title=A%20pseudorandom%20number%20generator%20based%20on%20piecewise%20logistic%20map&author=Yong.%20Wang&author=Zhaolong.%20Liu&author=Jianbin.%20Ma&author=Haiyuan.%20He&journal=Nonlinear%20Dyn&volume=83&issue=4&pages=2373-2391&publication_year=2016)

10. Murillo-Escobar MA, Cruz-Hernández C, Cardoza-Avendaño L, Méndez-Ramírez R (2017) A novel pseudorandom number generator based on pseudorandomly enhanced logistic map. Nonlinear Dyn 87(1):407–425
MathSciNet (http://www.ams.org/mathscinet-getitem?mr=3590771)
CrossRef (https://doi.org/10.1007/s11071-016-3051-3)
Google Scholar (http://scholar.google.com/scholar_lookup?title=A%20novel%20pseudorandom%20number%20generator%20based%20on%20pseudorandomly%20enhanced%20logistic%20map&author=MA.%20Murillo-Escobar&author=C.%20Cruz-Hern%C3%A1ndez&author=L.%20Cardoza-Avenda%C3%B1o&author=R.%20M%C3%A9ndez-Ram%C3%ADrez&journal=Nonlinear%20Dyn&volume=87&issue=1&pages=407-425&publication_year=2017)

11. Luca A, Ilyas A, Vlad A (2011) Generating random binary sequences using tent map. In: ISSCS 2011-international symposium on signals, circuits and systems. IEEE, pp 1–4
Google Scholar (https://scholar.google.com/scholar?q=Luca%20A%2C%20Ilyas%20A%2C%20Vlad%20A%20%282011%29%20Generating%20random%20binary%20sequences%20using%20tent%20map.%20In%3A%20ISSCS%202011-international%20symposium%20on%20signals%2C%20circuits%20and%20systems.%20IEEE%2C%20pp%201%E2%80%934)

12. Cristina DA, Radu B, Ciprian R (2012) A new pseudorandom bit generator using compounded chaotic tent maps. In: 2012 9th International Conference on Communications (COMM). IEEE, pp 339–342
Google Scholar (https://scholar.google.com/scholar?q=Cristina%20DA%2C%20Radu%20B%2C%20Ciprian%20R%20%282012%29%20A%20new%20pseudorandom%20bit%20generator%20using%20compounded%20chaotic%20tent%20maps.%20In%3A%202012%209th%20International%20Conference%20on%20Communications%20%28COMM%29.%20IEEE%2C%20pp%20339%E2%80%93342)

13. Zheng Y, Zheng J (2018) Chaotic random sequence generated from tent map on variant maps. J Math Comput Sci
Google Scholar (https://scholar.google.com/scholar?q=Zheng%20Y%2C%20Zheng%20J%20%282018%29%20Chaotic%20random%20sequence%20generated%20from%20tent%20map%20on%20variant%20maps.%20J%20Math%20Comput%20Sci)

14. Valtierra JL, Tlelo-Cuautle E, Rodríguez-Vázquez Á (2017) A switched-capacitor skew-tent map implementation for random number generation. Int J Circuit Theory Appl 45(2):305–315
Google Scholar (https://scholar.google.com/scholar?q=Valtierra%20JL%2C%20Tlelo-Cuautle%20E%2C%20Rodr%C3%ADguez-V%C3%A1zquez%20%C3%81%20%282017%29%20A%20switched-capacitor%20skew-tent%20map%20implementation%20for%20random%20number%20generation.%20Int%20J%20Circuit%20Theory%20Appl%2045%282%29%3A305%E2%80%93315)

15. Palacios-Luengas L, Pichardo-Méndez JL, Díaz-Méndez JA, Rodríguez-Santos F, Vázquez-Medina R (2019) PRNG based on skew tent map. Arabian J Sci Eng 44(4):3817–3830
CrossRef (https://doi.org/10.1007/s13369-018-3688-y)
Google Scholar (http://scholar.google.com/scholar_lookup?title=PRNG%20based%20on%20skew%20tent%20map&author=L.%20Palacios-Luengas&author=JL.%20Pichardo-M%C3%A9ndez&author=JA.%20D%C3%ADaz-M%C3%A9ndez&author=F.%20Rodr%C3%ADguez-Santos&author=R.%20V%C3%A1zquez-Medina&journal=Arabian%20J%20Sci%20Eng&volume=44&issue=4&pages=3817-3830&publication_year=2019)

16. Hu HanPing, Liu LingFeng, Ding NaiDa (2013) Pseudorandom sequence generator based on the Chen chaotic system. Comput Phys Commun 184(3):765–768
MathSciNet (http://www.ams.org/mathscinet-getitem?mr=3007059)
CrossRef (https://doi.org/10.1016/j.cpc.2012.11.017)
Google Scholar (http://scholar.google.com/scholar_lookup?title=Pseudorandom%20sequence%20generator%20based%20on%20the%20Chen%20chaotic%20system&author=HanPing.%20Hu&author=LingFeng.%20Liu&author=NaiDa.%20Ding&journal=Comput%20Phys%20Commun&volume=184&issue=3&pages=765-768&publication_year=2013)

17. Li Ping, Li Zhong, Halang Wolfgang A, Chen Guanrong (2006) A multiple pseudorandom-bit generator based on a spatiotemporal chaotic map. Phys Lett A 349(6):467–473
CrossRef (https://doi.org/10.1016/j.physleta.2005.09.060)
Google Scholar (http://scholar.google.com/scholar_lookup?title=A%20multiple%20pseudorandom-bit%20generator%20based%20on%20a%20spatiotemporal%20chaotic%20map&author=Ping.%20Li&author=Zhong.%20Li&author=Wolfgang%20A.%20Halang&author=Guanrong.%20Chen&journal=Phys%20Lett%20A&volume=349&issue=6&pages=467-473&publication_year=2006)

18. Sun Fuyan, Liu Shutang (2009) Cryptographic pseudo-random sequence from the spatial chaotic map. Chaos, Solitons Fractals 41(5):2216–2219
CrossRef (https://doi.org/10.1016/j.chaos.2008.08.032)
Google Scholar (http://scholar.google.com/scholar_lookup?title=Cryptographic%20pseudo-random%20sequence%20from%20the%20spatial%20chaotic%20map&author=Fuyan.%20Sun&author=Shutang.%20Liu&journal=Chaos%2C%20Solitons%20Fractals&volume=41&issue=5&pages=2216-2219&publication_year=2009)

19. Stoyanov B, Kordov K (2015) Novel secure pseudo-random number generation scheme based on two tinkerbell maps. Adv Stud Theor Phys 9(9):411–421
CrossRef (https://doi.org/10.12988/astp.2015.5342)
Google Scholar (http://scholar.google.com/scholar_lookup?title=Novel%20secure%20pseudo-random%20number%20generation%20scheme%20based%20on%20two%20tinkerbell%20maps&author=B.%20Stoyanov&author=K.%20Kordov&journal=Adv%20Stud%20Theor%20Phys&volume=9&issue=9&pages=411-421&publication_year=2015)

20. Hua Z, Zhou Y, Pun C-M, Philip Chen CL (2015) 2D Sine Logistic modulation map for image encryption. Inf Sci 297:80–94
Google Scholar (https://scholar.google.com/scholar?q=Hua%20Z%2C%20Zhou%20Y%2C%20Pun%20C-M%2C%20Philip%20Chen%20CL%20%282015%29%202D%20Sine%20Logistic%20modulation%20map%20for%20image%20encryption.%20Inf%20Sci%20297%3A80%E2%80%9394)

21. Hua Z, Zhou Y (2016) Image encryption using 2D Logistic-adjusted-Sine map. Inf Sci 339:237–253
CrossRef (https://doi.org/10.1016/j.ins.2016.01.017)
Google Scholar (http://scholar.google.com/scholar_lookup?title=Image%20encryption%20using%202D%20Logistic-adjusted-Sine%20map&author=Z.%20Hua&author=Y.%20Zhou&journal=Inf%20Sci&volume=339&pages=237-253&publication_year=2016)

22. Avaroglu Erdinc (2017) Pseudorandom number generator based on Arnold cat map and statistical analysis. Turkish J Electr Eng Comput Sci 25(1):633–643
CrossRef (https://doi.org/10.3906/elk-1507-253)
Google Scholar (http://scholar.google.com/scholar_lookup?title=Pseudorandom%20number%20generator%20based%20on%20Arnold%20cat%20map%20and%20statistical%20analysis&author=Erdinc.%20Avaroglu&journal=Turkish%20J%20Electr%20Eng%20Comput%20Sci&volume=25&issue=1&pages=633-643&publication_year=2017)

23. Barash L, Shchur. LN (2006) Periodic orbits of the ensemble of Sinai-Arnold cat maps and pseudorandom number generation. Phys Rev E 73(3)
Google Scholar (https://scholar.google.com/scholar?q=Barash%20L%2C%20Shchur.%20LN%20%282006%29%20Periodic%20orbits%20of%20the%20ensemble%20of%20Sinai-Arnold%20cat%20maps%20and%20pseudorandom%20number%20generation.%20Phys%20Rev%20E%2073%283%29)

24. Couchot J-F, Heam P-C, Guyeux C, Wang Q, Bahi JM (2014) Pseudorandom number generators with balanced gray codes. In: 11th International Conference on Security and Cryptography (SECRYPT). IEEE, pp 1–7
Google Scholar (https://scholar.google.com/scholar?q=Couchot%20J-F%2C%20Heam%20P-C%2C%20Guyeux%20C%2C%20Wang%20Q%2C%20Bahi%20JM%20%282014%29%20Pseudorandom%20number%20generators%20with%20balanced%20gray%20codes.%20In%3A%2011th%20International%20Conference%20on%20Security%20and%20Cryptography%20%28SECRYPT%29.%20IEEE%2C%20pp%201%E2%80%937)

25. Mascagni M, Cuccaro SA, Pryor DV, Robinson ML (1995) A fast, high quality, and reproducible parallel lagged-Fibonacci pseudorandom number generator. J Comput Phys 119(2):211–219
Google Scholar (https://scholar.google.com/scholar?q=Mascagni%20M%2C%20Cuccaro%20SA%2C%20Pryor%20DV%2C%20Robinson%20ML%20%281995%29%20A%20fast%2C%20high%20quality%2C%20and%20reproducible%20parallel%20lagged-Fibonacci%20pseudorandom%20number%20generator.%20J%20Comput%20Phys%20119%282%29%3A211%E2%80%93219)

26. Orue AB, Montoya F, Hernández Encinas L (2010) Trifork, a new pseudorandom number generator based on lagged fibonacci maps
Google Scholar (https://scholar.google.com/scholar?q=Orue%20AB%2C%20Montoya%20F%2C%20Hern%C3%A1ndez%20Encinas%20L%20%282010%29%20Trifork%2C%20a%20new%20pseudorandom%20number%20generator%20based%20on%20lagged%20fibonacci%20maps)

27. Gebhardt Friedrich (1967) Generating pseudo-random numbers by shuffling a Fibonacci sequence. Math Comput 21(100):708–70
MathSciNet (http://www.ams.org/mathscinet-getitem?mr=223064)
CrossRef (https://doi.org/10.1090/S0025-5718-1967-0223064-6)
Google Scholar (http://scholar.google.com/scholar_lookup?title=Generating%20pseudo-random%20numbers%20by%20shuffling%20a%20Fibonacci%20sequence&author=Friedrich.%20Gebhardt&journal=Math%20Comput&volume=21&issue=100&pages=708-70&publication_year=1967)

28. Chernov N, Markarian R (2006) Chaotic billiards. No. 127. American Mathematical Soc
Google Scholar (https://scholar.google.com/scholar?q=Chernov%20N%2C%20Markarian%20R%20%282006%29%20Chaotic%20billiards.%20No.%20127.%20American%20Mathematical%20Soc)

29. Lee JJ, Lee S, Yoon T (2016) Improvement of KMRNG Using n-Pendulum. In: International conference on intelligent computing. Springer, Cham, pp 670–681
Google Scholar (https://scholar.google.com/scholar?q=Lee%20JJ%2C%20Lee%20S%2C%20Yoon%20T%20%282016%29%20Improvement%20of%20KMRNG%20Using%20n-Pendulum.%20In%3A%20International%20conference%20on%20intelligent%20computing.%20Springer%2C%20Cham%2C%20pp%20670%E2%80%93681)

30. Everson RM (1986) Chaotic dynamics of a bouncing ball. Physica D: Nonlinear Phenom 19(3):355–383
Google Scholar (https://scholar.google.com/scholar?q=Everson%20RM%20%281986%29%20Chaotic%20dynamics%20of%20a%20bouncing%20ball.%20Physica%20D%3A%20Nonlinear%20Phenom%2019%283%29%3A355%E2%80%93383)

31. Gangyi H, Jin P, Weili K (2019) A novel algorithm for generating pseudo-random number. Int J Comput Intell Syst 12(2):643–648
Google Scholar (https://scholar.google.com/scholar?q=Gangyi%20H%2C%20Jin%20P%2C%20Weili%20K%20%282019%29%20A%20novel%20algorithm%20for%20generating%20pseudo-random%20number.%20Int%20J%20Comput%20Intell%20Syst%2012%282%29%3A643%E2%80%93648)

32. Rukhin A, Soto J, Nechvatal J, Smid M, Barker E (2001) A statistical test suite for random and pseudorandom number generators for cryptographic applications. Booz-Allen and Hamilton Inc Mclean Va
Google Scholar (https://scholar.google.com/scholar?q=Rukhin%20A%2C%20Soto%20J%2C%20Nechvatal%20J%2C%20Smid%20M%2C%20Barker%20E%20%282001%29%20A%20statistical%20test%20suite%20for%20random%20and%20pseudorandom%20number%20generators%20for%20cryptographic%20applications.%20Booz-Allen%20and%20Hamilton%20Inc%20Mclean%20Va)

Copyright information

About this paper

Cite this paper as:
Paul A., Kandar S., Dhara B.C. (2021) Generation of Pseudorandom Sequence Using Regula-Falsi Method. In: Giri D., Buyya R., Ponnusamy S., De D., Adamatzky A., Abawajy J.H. (eds) Proceedings of the Sixth International Conference on Mathematics and Computing. Advances in Intelligent Systems and Computing, vol 1262. Springer, Singapore. https://doi.org/10.1007/978-981-15-8061-1_31

- First Online 11 December 2020
- DOI https://doi.org/10.1007/978-981-15-8061-1_31
- Publisher Name Springer, Singapore
- Print ISBN 978-981-15-8060-4
- Online ISBN 978-981-15-8061-1
- eBook Packages Intelligent Technologies and Robotics Intelligent Technologies and Robotics (R0)

Personalised recommendations