Deterministic Random Number Generation: Pseudorandom Number Generator

[1]**Dinesh Kumar M**, U.G student, Panimalar Engineering College, India
[2]**Lithish Kumar**, U.G student, Panimalar Engineering College, India
[3]**Jeffrey Steve Paul**, U.G student, Panimalar Engineering College, India
[4]**Dwaraka Thejdeep** , U.G student, Panimalar Engineering College, India

Abstract
A random number generator is an algorithm designed to generate random numbers at any given point of time. Random number generators are used in gaming, cyber security, cryptography, big data analysis and various other fields. Random number generators are of two types- true random number generators (TRNGs) and pseudo random generators (PRNGs). In cryptography, PRNGs are preferred as they are more reliable than TRNGs. This paper covers the utilization of Pseudorandom Number Generators in various fields. Also, the paper covers the various types of algorithms used in PRNGs and their applications.
Keywords-Random Number Generator, Random Number, TRNG, PRNG

1.Introduction

A pseudorandom generator is a method that uses algorithms to simulate and generate random numbers. They are also known as deterministic random bit generators (DRBG) . But unlike a true random number generator, a PRNG makes use of several techniques and mathematical formulae to generate random numbers which are not completely random and the values are related to the initial value. Therefore, there is a predetermined key that can be used to decrypt the data that is encrypted. Logically, TRNG seems to be the safer type of RNG to use for encryption, but TRNGs have their drawbacks. TRNG is prone to easy wear and tear since it is subjected to complete randomness or entropy. PRNGs on the other hand are much more efficient when it comes to controlled encryption. A common yet complicated formula can be used to encrypt multiple data, this makes it time and cost efficient for the company using encryption in their products. The aim of PRNG is to create a complex yet time and cost efficient formula to encrypt data. Real time examples for the use of RNGs are games such as casino, bingo, card games,etc.

2. Related Works

Random numbers are used in multiple fields. A discrete representation can be obtained from pseudorandom number generators. [1]

By comparing cryptographically insecure pseudorandom on the basis of their time for computation, complexity of code and algorithms used, the speed of a PRNG's response time can be determined, or which gives the most similar random numbers, and which generator is applicable for a specific use. [2]

By using Field Programmable Gate Array (FPGA) of a PRNG that adopts the K-logistic map concept, without prioritizing the most important K decimal digits of a background orbit generated from the usual logistic map equation. These experiments show the development of a low-cost and highly efficient RNG. [3]

Pseudo-random number generators can be used to generate encryption keys, SSL connections, and database security. The methods of the processing of the above is unclear. [4]

A lightweight PRNG can also be implemented by the modification of the function of feedback (OFB). This can be used to measure the efficiency of a pseudorandom number generator. [5]

A 3-dimensional discrete chaotic system can be converted into a 6-dimensional chaotic generalized synchronic system on the basis of the chaotic system and a chaos generalized synchronization (GS) theorem, a 6-dimensional chaotic generalized synchronic system is developed. [6]

Time-Variant Recursion of Accumulators is a good technique for applications requiring high-quality pseudorandom sequences, and it is approved by the tests from the National Institute of Standards and Technology. [7]

The study of statistics of pseudorandom number sequence (PRS) generators reveals the quality/speed ratio compared to ordinary PRNG and TRNG. [8]

3. Techniques/Formulae used in pseudorandom number generation

-In 1946, John von Neumann suggested an early PRNG algorithm known as the middle-square method. As shown in Fig (1), this method, the input also known as the seed is squared and the middle value of the squared seed is taken as the output.

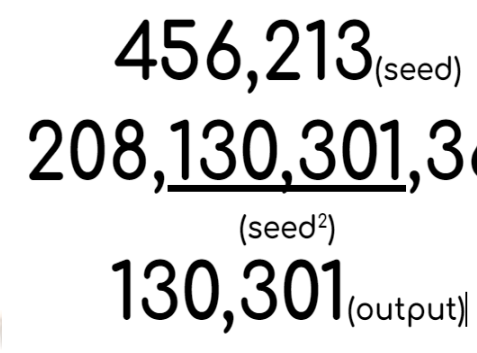

Fig (1)

The method had its flaws, such as if the seed is '000000' then the result is the same as the seed.

-Linear congruential generator (LCG) is another old algorithm that creates a sequence of random numbers (pseudo) which are measured using a piecewise linear equation that is discontinuous. LCG was invented by D.H. Lehmer in the year 1949.
Equation for LCG:
$L_n \equiv (a \cdot L_{n-1} + c) \bmod m$
Where,
L_0 is the seed value ($0<=L_0<m$)
m is the modulus ($m>0$),
a is the multiplier ($0<a<m$)
and c is increment ($0<=c<m$)

-Linear Feedback Shift Register (LFSR) is a type of shift register whose input bit is the linear function (XOR) of previous states. LFSR was developed by Solomon W. Golomb in 1967. As Fig (2) shows, LFSR consists of 3 parts namely the shift register, a linear feedback function and clock. The collections of bits initially generated are known as the shift registers. It is a simple way to generate pseudorandom numbers.

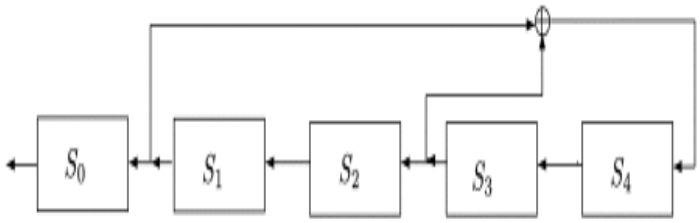

Fig (2)

-Blum Blum Shub Generator (BBS) is another pseudorandom number generator introduced by Lenore Blum, Manuel Blum and Michael Shub in 1986. This generator uses the following expression to generate a series of numbers:

$X_{n+1} = Xn^2 \bmod M$
Where,
$M = p*q$,
X_0 should be an integer where
X_0 should not be equal to 0 or 1
P = 3 (mod of 4)
q = 3 (mod of 4)
The figure Fig (3) shows the working of the BBS generator.

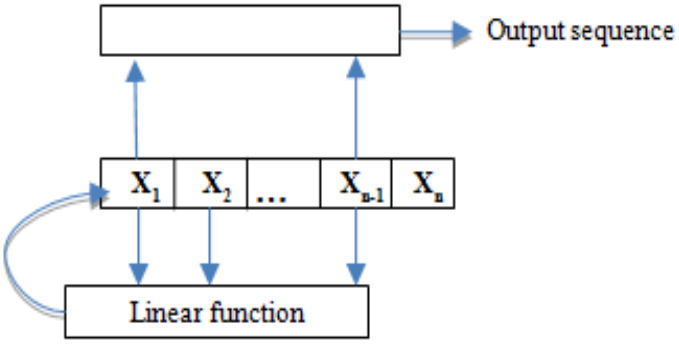

Fig (3)

-Lagged Fibonacci generators (LFG or LFib) is another algorithm used in pseudorandom number generation as a step up to the linear congruential generator. It is referred to as a lagged generator as it is always one step slower than the PRNG. As we all know, the Fibonacci series, which is also known as the golden ratio, has a unique sequence of numbers when implemented. The formula used in LFib is:

$S_n \equiv S_{n-j} * S_{n-k} \pmod m$

Where,

$0<j<k$

$m=2^n$ (n is a positive integer)

It is also known as a two-tap generator as 2 values are taken into consideration (shown in Fig (4)) for calculation (j and k).

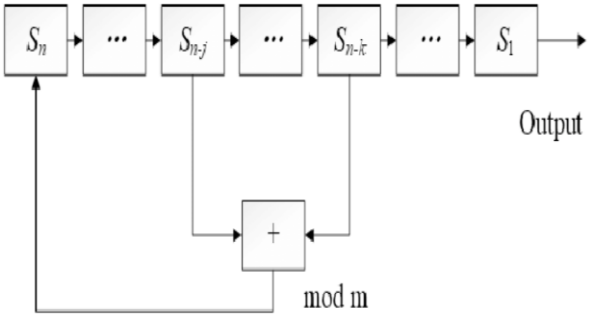

Fig (4)

4. PRNG in cryptography

PRNG in cryptography is called a cryptographic pseudorandom generator (CPRNG). It is also known as a cryptographically secure pseudorandom generator (CSPRNG). In cryptography, PRNGs are used to generate stream ciphers and sessions.

A PRNG should satisfy the following conditions to be used as a CPRNG:

-It should be able to withstand the state compromise extensions, that is, in case a part of the seed or the seed as a whole has been discovered, it should be able to prevent the reverse engineering to the seed.

-It should satisfy the next-bit test.

5. PRNG in games

Randomness in games is usually created using a PRNG. An algorithm is used to determine the outcome of an input in a game. For example, it can be used to determine whether an attack hits an opponent, whether you draw the card to win the game, etc,. In this day and age PRNG is widely used in games to make it more fun and profitable. RNG in games are used as a method for income in games, for example summoning for a character, cosmetics, etc,.

6. Algorithm for the usage of PRNG in encryption

This algorithm represents the encryption of a message using PRNG:

Step 1: Start the application/module

Step 2: Activate the program for pseudorandom number generation

Step 3: Feed the initial message into the PRNG program that is to be encrypted

Step 4: The input is converted into a random sequence of characters. A key to convert this randomly generated sequence is also generated.

Step 5: This key is sent to the end user i.e., the output receiver.

Step 6: Then, the random sequence is sent to the receiver side, where the key is used to convert the sequence into the original message.

Step 7. Stop the program

7. Conclusion

In this paper, we discussed the different techniques used in implementing pseudorandom number generators. These techniques show the evolution of pseudorandom number generators. The use of pseudorandom number generators in cryptography is also covered in this paper. The randomization in games using pseudorandom number generators is also explained.

Acknowledgement

We humbly thank our friends and mentors for motivating and guiding us towards the completion of this paper.

Funding: "This research received no external funding"

Conflicts of Interest: "The authors share the same opinions."

References

[1]PseudoRandom: A proposal of educational material for pseudorandom number generators learning by Veronica Laura; Maria Paula Dieser; 09-13 October 2017
[2]An Empirical Study of Non-Cryptographically Secure Pseudorandom Number Generators by Mehul Singh; Prabhishek Singh; Pramod Kumar; 13-14 March 2020
[3]FPGA Implementation of a Pseudorandom Number Generator Based on k – Logistic Map by Matheus M. de A. Kotaki; Maximiliam Luppe; 25-28 February 2020
[4]Pseudo Random Bit Generation Using Arithmetic Progression by Ankur ;Divyanjali;Trishansh Bhardwaj; 19-20 February 2015
[5]The development and researching of lightweight pseudorandom number generators by I.V.Chugunkov ;O.Yu.Novikova; V.A.Perevozchikov; S.S.Troitskiy; 02-03 February 2016
[6]A Generalized Stability Theorem for Continuous Chaos Systems and Design of Pseudorandom Number Generator by Xue Wang;Lequan Min; Mei Zhang; 19-20 December 2015
[7]A Pseudorandom Number Generator Based on Time-Variant Recursion of Accumulators by Victor R.Gonzalez; Fabio Pareschi; Gianluca Setti; Franco Maloberti; 18 August 2011
[8]Efficiency Comparison of Pseudorandom Number Generators Based on Strong Cryptographic Algorithms by Vladimir Grozov;Marina Budko;Alexei Guirik; Mikhail Budko; 05-09 November 2018