HCTL Open Int. J. of Technology Innovations and Research
HCTL Open IJTIR, Volume 4, July 2013
e-ISSN: 2321-1814
ISBN (Print): 978-1-62776-132-1

An Introduction to Pseudorandom Number Generator

Ankur [*] ***and Divyanjali*** [†]
ankur.rathi9@gmail.com and dvynjli@gmail.com

Abstract

Random numbers are important in every aspect of cryptography. We are evaluating the basic principles which are essential in the design of uniform random number generators, their most important quality requirements, their theoretical study, and their practical testing. National Institute of Standards and Technology (NIST) statistical test suite is the best test suite provided to test the accuracy of randomness of any Pseudo-random number generator. As the time passes the problem of generating unpredictable random sequences become a matter of research.

Keywords

Pseudorandom Number Generator.

[*] Student at Banasthali Vidyapith - Rajasthan, India
[†] Student at Banasthali Vidyapith - Rajasthan, India

e-ISSN: 2321-1814
ISBN (Print): 978-1-62776-132-1

Introduction

Random numbers is the output of a deterministic algorithm which is used to produce the numbers which must look like a random sequence and should satisfy all the criteria of randomness and the probability of occurrence of any number from the universe of numbers has an equal chance of being chosen [1]. An important utility of digital computer systems is the ability to generate random numbers. Pseudo-random numbers, frequently called as random numbers, but these pseudo-random numbers are generated with the help of an algorithms which are deterministic in nature or by the help of hardware (e.g. shift registers). However, the algorithms for generating pseudo-random numbers are deterministic in nature but the output of the algorithm cannot be determined in a polynomial time. Sequences generated, in a deterministic way, are usually called pseudo-random and Quasi-random number sequences [2].

As the time passes the problem of generating random numbers that should be unpredictable and have equal probability of occurrence from this vast universe of numbers become a matter of research and it is said that good random numbers are hard to find [3]. But generating random number from a random idea is not possible as it is quoted by the most famous scientist of computer science, Dr. Donald E. Knuth who also designs a test suite to test the random number generator like the NIST Test suite [4]. New rapid improvements in the telecommunication related technologies especially in the Internet and mobile communication networks have increased the use of information transmission, which sequentially presented new challenges for defending the information flow from unlawful eavesdropping. It has intensified the research activities in the field of cryptographic techniques [5, 6]. The safekeeping of many cryptographic systems depends upon the randomness of pseudo-random sequences which can be used as input (key or pin) quantities. These quantities must be of suitable size and random in the point that the probability of any particular sequence being selected must be sufficiently minimal to predict the upcoming sequences from a random number generator.

The NIST tests suite, a statistical package consisting of 16 tests that are especially developed to test the randomness of binary sequences based on cryptographic random or pseudo random bit generators generated by either hardware or software. These tests focus on a variety of different possible types of non-randomness that can exist in a binary sequence. John von Neumann first suggested the approach of random numbers using arithmetic operations on computer in about 1946, using the `middle-square` method [2], but this generator

e-ISSN: 2321-1814
ISBN (Print): 978-1-62776-132-1

was slow and sometimes not satisfactory. John Mauchly in an unpublished paper presented to a statistic conference in 1949, extended the middle square method after that the classical Linear congruential generator (LCG) was introduced it was the most widely used technique for non-cryptographic purpose, LCG method was introduced by D. H. Lehmer in 1949. Later Fibonacci, Additive, Multiple congruential many more generators were introduced but none of them can be used for cryptographic application because they are not cryptographically secure. Then after, first cryptographically secure pseudo-random number was introduced by Blum and Micai which was based on the un-tractability of discrete logarithm. Blum-Blum-Shub (BBS) [2, 7] is a generator based on quadratic residues proposed in 1986. It is the simplest and most widely used crypto-orientation PRNG. BBS is appropriate only for cryptography and not for simulation, because it is not very fast.

Types of PRNG

Cryptography and randomness are closely related. Perfect secrecy can be achieved if the key is secure, according to Kerckhoff's Principle guessing the key should be so difficult that there is no need to hide to encryption/decryption algorithm. Secrecy can be achieved if the key of the encipherment algorithm is truly a random number. There are two approaches to generating a long stream of random bits: using a natural random process, such as flipping a coin many times and interpreting heads and tails as 0-bits and 1-bits, or using a deterministic process with feedback. The first approach is called a true random number generator (TRNG); the second is called a pseudo-random number generator (PRNG).

True Random Number/Bit Generators

A true random number generator (TRNG) uses a non-deterministic source to produce randomness. Most function by measuring random natural processes, such as hardware using pulse detectors of ionizing radiations, gas discharge tubes, and/or leaky capacitors. Intel has developed a commercially available chip that samples thermal noise by amplifying the voltage measured across non-driven resistors [7]. Lavarnd is an open source project for creating truly random numbers using inexpensive hardware (including a low cost camera) and an open source code.

HCTL Open Int. J. of Technology Innovations and Research
HCTL Open IJTIR, Volume 4, July 2013
e-ISSN: 2321-1814
ISBN (Print): 978-1-62776-132-1

Pseudo-random Number Generation

A Pseudo-Random Bit Generator (PRBG) is a deterministic algorithm which, given a truly-random binary sequence of length n, outputs a binary sequence of length $l(n) > n$ which appears to be random, with $l(n)$ being a polynomial. The input to the PRBG is called the seed, and the output is called a pseudo-random bit sequence. A pseudo-random number generator uses this approach. The generated number is not truly random because the process that creates bit it is deterministic [8].

Cryptographically Secure PRNGs : A PRBG passes all polynomial-time statistical tests if no polynomial-time algorithm can correctly distinguish between an output sequence of the generator and a truly random sequence of the same length with probability significantly greater than 0.5. A PRBG passes the next-bit test if there is no polynomial-time algorithm which, on input of the first l-bits of an output sequence s can predict the $(l+1)$st bit of s with probability significantly greater than 0.5. James Reeds [9] enumerate two distinguished standards of randomness. The cryptography standard has to do with predictability. It is less important whether the sequence is uniformly distributed, but it is essential that knowing part of the sequence does not contribute any knowledge about other parts. This requirement is called unpredictability.

Shamir : Shamir [10] was the first to publish a PRNG that is proved to be cryptographically strong in the sense of un-predictability. Shamir's scheme is based on the RSA public-key encryption function. The hardness of this generator is based on upon the integer factorization:

- $N = pq$, where p and q are secret large prime numbers.
- A seed S.
- A sequence of keys K_1, K_2,.., such that $\psi(N)$ and all the K_is are pair wise relatively prime.

The random numbers sequence is: $R1 = \frac{S1}{K1}(modN)$, $R2 = \frac{S1}{K1}(modN)$, ...

Blum Blum Shub : Blum-Blum-Shub (BBS) [11] is a generator based on quadratic residues proposed in 1986. It is the simplest and most widely used cryptographically secure PRNG. BBS is appropriate only for cryptography and not for simulation, because it is not very fast. BBS parameters are two large prime numbers p and q such that $p = q = 3(\text{mod } 4)$. $n = p * q$ is called the

Blum integer. Let n = p*q, and x be random element of QRn (just select a random element of Z_n, check if it's relatively prime with n, and square it).

Let $x_1 = x, x_2 = f_n(x), x_3 = f_n(x_2), ..., x_l = f_n(x_l)$. Output the least significant bit for each x_i: $X_{n+1} = X_n^2 \bmod n$

PRNG for Simulation

The Linear Congruential Method : Linear congruential generator (LCG) is the most widely used technique for non-cryptographic use. This method is introduced by D. H. Lehmer in 1949 [12].
Select four variables m, a, c, X_0,
where, m = modulus; $m > 0$, a = multiplier; $0 \leq a < m$, c = increment; $0 \leq c < m$, X_0 = starting value; $0 \leq X_0 < m$.
The required sequence of random number can be obtained by the following equation:
$X_{n+1} = (aX_n + c) \bmod m$, for $n \geq 0$.

Additive Number Generator : This method is introduced by G. J. Mitchell and D.P. Moore in 1958 (unpublished article). $X_n = (X_{n-24} + X_{n-55}) \bmod m$; for n $\geq$ 55.
Where, M is even, $X_0, ..., X_{54}$ are arbitrary integers not all even. The constants 24 and 54 are special values that happen to have the property that least significant bits (X_n mod 2) will have a period of length $2^{55} - 1$.

Quadratic Congruential Generator : It is proposed by R. R. Coveyou. It is used to get more random numbers.
$X_{n+1} = (dX_{n+2} + aX_n + c) \bmod m$
The conditions on d, a and c required for the maximum period (which matches the Modulus) are:

1. c must be relatively prime to the modulus.
2. a is equal to 1 modulo every odd prime factor of the modulus.
3. d is equal to 0 modulo every odd prime factor of the modulus.

Quality Criteria and Testing

Various statistical tests can be applied to the output of any pseudorandom number generator to attempt to verify the randomness of the generated sequence.

HCTL Open Int. J. of Technology Innovations and Research
HCTL Open IJTIR, Volume 4, July 2013
e-ISSN: 2321-1814
ISBN (Print): 978-1-62776-132-1

Randomness of a random sequence can be described and calculated in terms of probability. The likelihood of an outcome (known as priori) from a statistical test, when applied to a truly random sequence, can be explained in probabilistic terms. There are an infinite number of possible statistical tests, capable of detecting the presence or absence of a `pattern` from a true random sequence, which if detected, would indicate that the sequence is not to be considered as a random number. Because of the availability of a number of tests for detecting a true random sequence, no specific finite set of tests can be considered to be `complete` in its own. The results of statistical testing must be cross-checked systematically to avoid incorrect conclusions about a specific random sequence generator.

To get the confidence from a newly developed pseudo random bit generators, and to make sure that they are cryptographically secure, they should be subjected to a range of statistical tests especially designed to detect and analyse their specific characteristics as expected from their truly random sequences. There are several options available for such analysis. The four most popular options are:

1. NIST suite of statistical tests [4],
2. The DIEHARD suite of statistical tests [13],
3. The Crypt-XS suite of statistical tests [14] and
4. The Donald Knuth's statistical tests set [15].

Various efforts based on the principal component analysis show that not all the above mentioned suites are needed to implement at a time as there are redundancy in the statistical tests (i.e., all the tests are not independent). The results demonstrated that the NIST statistical tests suite contains a adequate number of nearly self-sufficient and independent statistical tests, which are capable of detecting any deviation from the randomness [16]. Hence for analysing the randomness of the proposed pseudo random bit generator (PRBG) NIST can be trusted.

These are the following test of NIST:

1. Frequency Test
2. Frequency Test within a Block
3. Runs Test: Test for the Longest Run of Ones in a Block

4. Binary Matrix Rank Test
5. Discrete Fourier Transform (Spectral) Test
6. Non-overlapping Template Matching Test
7. Overlapping Template Matching Test
8. Maurer's `Universal Statistical` Test
9. Linear Complexity Test: Serial Test
10. Approximate Entropy Test
11. Cumulative Sums (Cusum) Test
12. Random Excursions Test: Random Excursions Variant Test

NIST Final Analysis Report after performing the above mentioned results this is only a portion.

```
------------------------------------------------------------------------------
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
------------------------------------------------------------------------------
   generator is <./data/bbs50bits.txt>
------------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
------------------------------------------------------------------------------
108  95 103  98  91  99  96 104 105 101  0.982958    988/1000    Frequency
108 111  95  92 106 118 103  96  87  84  0.286836    991/1000    BlockFrequency
103 115  80  77  93 111 100 112  97 112  0.060875    992/1000    CumulativeSums
101  98 106 104  83  97  91 103 110 107  0.745908    988/1000    CumulativeSums
 93 110 107  92  93  99 106 104 102  94  0.908760    986/1000    Runs
126  86  95 107  92  92 108 103  94  97  0.217857    972/1000 *  LongestRun
114  99 103  95 103  94  90 123  84  95  0.221317    992/1000    Rank
146  89  95  97 102  96 104  92  89  90  0.002105    960/1000 *  FFT
108 102  93 102  89 111  96  98 110  91  0.775337    991/1000    NonOverlappingTemplate
101  94 100 107 115  94  97  86 106 100  0.751866    986/1000    NonOverlappingTemplate
 99  87 116  97 104  92 109  93  96 107  0.626709    987/1000    NonOverlappingTemplate
 97 108 100  92 106  89 106  84 110 108  0.585209    988/1000    NonOverlappingTemplate
100 100  77 115 101  95  99 109 106  98  0.435430    988/1000    NonOverlappingTemplate
110  94 101  99  94  92 114 100 111  85  0.554420    987/1000    NonOverlappingTemplate
 99  93 106  94  92 111 106 112  89  98  0.727851    989/1000    NonOverlappingTemplate
 98 101  91 113  97  88 101 106 108  97  0.818343    988/1000    NonOverlappingTemplate
```

Conclusion

There are several pseudo-random number generators, but none of them is very good. Some are slow, some are not sufficiently random and others are not cryptographically secure. As we have mention many generators above there some of them can be used for Simulation purpose or some of them for Cryptographic purpose because cryptographically PRNGs are most of the time slow and if they are used for simulation it is the wastage of resources.

References

[1] James E. Gentle, `Random Number Generation and Monte Carlo Methods`, Springer (1998).

[2] D. E. Kunth, `The Art of Computer Programming, Vol 2: Semi-Numerical Algorithms`, 2nd ed., Addison-Wesley (1981).

[3] Stephen K. Park, Keith W. Miller, `Random Number Generators: Good ones are hard to find`, Communications of the ACM 31, 1192-1201.

[4] Runkin et al., `Statistical test suite for random and pseudo random number generators for cryptographic applications`, NIST special publication 800-22, 2001.

[5] Schneier B., `Applied Cryptography-Protocols, algorithms and source code in C`, John Wiley & Sons, New York, USA, 1996.

[6] Menezes A.J., Oorschot P.C.V. and Vanstone S.A., `Handbook of Applied Cryptography`, CRC Press, Boca Raton, 1997.

[7] Jun, B. and Kocher P., `The Intel Random Number Generator`, Intel White Paper, April 1987.

[8] J. A. Reeds, `Cracking a random number generator`, Cryptologia, 1(1), 1977.

[9] A. Shamir, `On the generation of cryptographically strong pseudo-random sequences`, In Proc. ICALP, pages 544-550. Springer, 1981.

[10] Lenore Blum, Manuel Blum, and Michael Shub., `A Simple Unpredictable Pseudo-Random Number Generator`, SIAM Journal on Computing, volume 15, pages 364-383, May 1986.

[11] D. H. Lehmer, `Mathematical Method in large-scale computing unit`, Proceeding of the second symposium on Large-scale Digital Computing Machinery, Harvard University Press, Cambridge, Massachusetts. 141-146 (1951).

[12] http://csrc.nist.gov/groups/ST/toolkit/rng/documents/nissc-paper.pdf

[13] Marsaglia G. Diehard statistical tests, http://stst.fsu.edu/pub/diehard, 1995.

e-ISSN: 2321-1814
ISBN (Print): 978-1-62776-132-1

[14] Gustafson H., `A computer package for measuring the strength of encryption algorithms`, J. Computer Security, vol. 13, pp. 687-697, 1994.

[15] Knuth D., `The art of computer programming: semiemperical algorithms`, Addison Wesley, Reading, USA, 1998.

[16] Soto J., `Statistical testing of random number generators`. Proceedings of 22nd National Information System Security Conference, http://csrc.nist.gov/groups/ST/toolkit/rng/documents/nissc-paper.pdf