Int. J. Math. And Appl., 8(4)(2020), 85–90

ISSN: 2347-1557

Available Online: http://ijmaa.in/

International Journal of *Mathematics And its Applications*

Review Paper on Random Number Generators Used in Cryptosystems

Vidya S. Umadi[1,*] and A. M. Sangogi[2]

1 Department of Mathematics, VSMSRK Institute of Technology, Nipani, Karnataka, India.

2 Department of Mathematics, Shaikh College of Engineering & Technology, Belagavi, Karnataka, India.

Abstract: A variety of random number generator techniques has been developed during the past years and has been used in different cryptosystems. This paper reviews the existing methods of random number or sequence generator techniques and proposes a new technique using chaos theory.

Keywords: Random Numbers, Cryptosystems, Sequence generator techniques.

1. Introduction

Random number generation (RNG) is a process which through a device, generates a sequence of numbers or symbols that cannot be reasonably predicted better than by a random chance. There is two types of random number generators which are presently being utilized and these are generators known as 'pseudo-random' and 'true random' generators. The pseudo-random generators generate a new number starting from an initial number by means of a predetermined algorithm. By applying successively the same algorithm on the last number generated, there is obtained a sequence of numbers which are all predetermined by the chosen algorithm and the initial number. Thus there is obtained with a very good approximation, one sequence of random numbers which are mutually independent and uniformly distributed in a given interval. The disadvantage of this method is that it is difficult to obtain a plurality of mutually independent sequences.

PGP random number generator

- PGP uses a complex and powerful scheme for generating random numbers. PGP generates random numbers from the content and timing of user keystrokes.
- General purpose application to protect (encrypt and/or authenticate) files.
- Can be used to protect e-mail messages.
- Can be used by corporations as well as individuals.
- Based on strong cryptographic algorithms (IDEA, RSA, SHA-1).

* *E-mail: vidyaumadi@gmail.com*

Hybrid random number generation

- The random number generator uses both methods to calculate the random number. For example it uses a true random number as ‘s’ initial value (often referred to as ”seed”) for an algorithm, that generates pseudorandom numbers.
- This method is the most commonly used method in OSes, since it is fast and generates very useful results.

Pseudo Random Number Generator

- Pseudorandom is defined as having the appearance of randomness, but nevertheless exhibiting a specific, repeatable pattern.
- Pseudorandom numbers using an algorithm based on the one in ANSI X9.17.
- Pseudorandom numbers are used to generate session keys used to generate initialization vectors (IVs) for use with the session key in CFB mode encryption.

Cyclic Random Number Generator

- All software-only pseudo-random number generators are cyclic, but the cycle may be long enough to be sufficient for practical purposes.
- A PRNG is a finite-state machine. It has some number of bits of internal state, no inputs that provide new information, and operates deterministically.

A true random generator usually utilizes a pulse generator (random or periodic but asynchronous with a clock) feeding a toggle flip-flop which changes its state at each pulse that it receives and a further flip-flop 45 which samples the first flip-flop during a transition of the clock. Seeing that the output of the first flip-flop is as often in its state representative of a_0 than a_1, there is obtained at the output of the second flip-flop a signal having a probability equal to 2. Furthermore, provided that the frequency of the clock signal is smaller than that of the pulse generator, each of the samples is independent from the others.

Random number generators have applications in gambling, statistical sampling, computer simulation, cryptography, completely randomized design, and other areas where producing an unpredictable result is desirable.

A method of generating random numbers which comprises sampling in a random manner a pseudo random generator. The random number generator comprises a pseudo-random number generator for generating pseudo-random numbers or most practical uses of random numbers. All true random number generators require a physical Source of entropy to produce random numbers, as distinct from algorithmically generated pseudorandom numbers, which are deterministic by nature of their source.

2. Existing Random Number Generators

(1). **Random Number Generators Using Monte Carlo Method:** Random number generators are very useful in developing Monte Carlo method simulations, as debugging is facilitated by the ability to run the same sequence of random numbers again by starting from the same randomised

$$x_0 = 0 \ \text{ given}, \ \ x_{n+1} = p_1 x_n + p_2 (mod \ \ N), \ \ \ n = 0, 1, 2, \ldots \tag{*}$$

The notation **mod N** means that the expression on the right of the equation is divided by N, and then replaced with the remainder.

$$x_0 = 79, \quad N = 100, \quad P_1 = 263, \quad and \quad P_2 = 7 \ \text{ Then}$$

$$x_1 = 79 * 263 + 71 \ (mod \ \ 100) = 20848 \ (mod \ \ 100) = 48,$$

$$x_1 = 48 * 263 + 71 \ (mod \ \ 100) = 12695 \ (mod \ \ 100) = 95,$$

$$x_1 = 95 * 263 + 71 \ (mod \ \ 100) = 25056 \ (mod \ \ 100) = 56,$$

$$x_1 = 56 * 263 + 71 \ (mod \ \ 100) = 14799 \ (mod \ \ 100) = 99.$$

(2). **Random Number Generation Using the Logistic Lattice:** Pseudo-random number generators based on non-linear dynamics. The recommended generator uses a special "re-mapped" form of the logistic equation at each of a finite set of nodes arranged in a 1- or 2-dimensional ring. Each node receives small perturbations from adjacent nodes by means of diffusive coupling equations the so called coupled map lattice. The generator is notable as the first to use non-linear dynamics in a lattice with a continuous state at each node. Time and space are discrete, but the state is continuous, unlike the cellular automata that Wolfram used to produce random numbers. It is a new source of pseudo-random numbers, unlike other current sources.

The logistic equation is $f\,(x) = 4x\,(x-1)$, $0 = x = 1$, known as the logistic equation, is historically interesting as one of the earliest proposed sources of pseudo-random numbers.It had a known algebraic distribution, so that iterated values could be transformed to the uniform distribution.

This article has presented a new pseudo-random number generator based on an interesting problem from non-linear dynamics namely the use of a variation of the logistic equation at each node of a coupled 1-dimensional or 2-dimensional lattice. There is statistical and theoretical evidence that this generator provides excellent pseudo-random numbers.

(3). **Pseudorandom Number Generator Using Probabilistic Polynomial:** A family of deterministic polynomial time computable functions $G_K : \{0,1\}^K \to \{0,1\}^{p(k)}$ for some polynomial p, is a **pseudorandom number generator** (PRNG, or PRG in some references), if it stretches the length of its input $\ (p(k) > k$ for any k) if its output is computationally indistinguishable from true randomness, i.e. for any probabilistic polynomial time algorithm A, which outputs 1 or 0 as a distinguisher.

$$\left|\Pr_{x \leftarrow \{0,1\}^k}[A(G(x)) = 1] - \Pr_{r \leftarrow \{0,1\}^{p(k)}}[A(r) = 1]\right| < \mu(k)$$

for some negligible function μ. There is an equivalent characterization: For any function family $G_K : \{0,1\}^K \to \{0,1\}^{p(k)}$. G is a PRNG if and only if the next output bit of G cannot be predicted by a polynomial time algorithm.

A **forward-secure** PRNG with block length $t(k)$, $G_K : \{0,1\}^K \to \{0,1\}^k \times \{0,1\}^{t(k)}$, where the input string δ_i with length k is the current state at period i, and the output (δ_{i+1}, y_i) consists of the next state δ_{i+1}, and the pseudorandom output block y_i of period i.

(4). **Random Number Generator Using Hexadecimal Digits of Pi:** The integer part of pi is taken as random number every time and the fraction part of pi value is multiplied by 16. He process is recursive and any length random numbers are used for sub keys in few cryptosystems. The initial values of sub keys of the blowfish algorithms are listed below.

Take fractional part and multiply by 16 which is 2.265482457436691 keep the integer 2 outside **2**.

P[0]	243f6a88	P[9]	38d01377
P[1]	85a308d3	P[10]	be5466cf
P[2]	13198a2e	P[11]	34e90c6c
P[3]	03707344	P[12]	c0ac29b7
P[4]	a4093822	P[13]	c97c50dd
P[5]	299f31d0	P[14]	3f84d5b5
P[6]	082efa98	P[15]	b5470917
P[7]	ec4e6c89	P[16]	9216d5d9
P[8]	452821e6	P[17]	8979fb1b

Table 1. **32-bit hexadecimal representation of initial values of sub-keys**

Fractional part of above number is $0.265482457436691 * 16$ which is 4.247719318987059 keep **4** outside.

Fractional part of above number is $0.247719318987059 * 16 = 3.963509103792947$ keep **3** outside.

Fractional part of above number is $0.963509103792947 * 16 = 15.41614566068716$ keep **15** outside and so on.

Fractional part of above number is $0.41614566068716 * 16 = 6.658330570994483$ keep **6** your random number is 2 4 3 1 5 6 and so on.

This is called hexadecimal fraction of pi.

(5). **Random Number Generation Using LFSR:** Linear feedback shift registers are introduced along with the polynomials that completely describe them. The application note describes how they can be implemented and techniques that can be used to improve the statistical properties of the numbers generated.

A two-dimensional random number generator for use in electronic applications is constructed of a shift-register random number generator using the coefficients of a primitive polynomial of degree k to generate Sequences of random binary numbers, and a Second random number generator to provide an index to an array of Storage locations for Storing the Sequences of random binary numbers generated by the shift-register random number generator.

LFSR is specified entirely by its polynomial. For example, a 6th degree polynomial with every term present is represented with the equation $x^6+x^5+x^4+x^3+x^2+x+1$. There are $2^{(6-1)} = 32$ different possible polynomials of this size. Just as with numbers, some polynomials are prime or primitive. We are interested in primitive polynomials because they will give us maximum length periods when shifting. A maximum length polynomial of degree n will have $2^n - 1$ different states. A new state is transitioned to after each shift. Consequently, a 6th degree polynomial will have 31 different states. Every number between 1 and 31 will occur in the shift register before it repeats. In the case of primitive 6th degree polynomials, there are only six.

Every primitive polynomial will have an odd number of terms, which means that every mask for a primitive polynomial will have an even number of 1 bit. Every primitive polynomial also defines a second primitive polynomial, its dual. The dual can be found by subtracting the exponent from the degree of the polynomial for each term. For example, given the 6th degree polynomial, x^6+x+1, its dual is $x^{6-6}+x^{6-1}+x^{6-0}$, which is equal to x^6+x^5+1. In Table 1, polynomials 1 and 2, 3 and 4, 5 and 6 are the dual of each other.

LFSRs can never have the value of zero, since every shift of a zeroed LFSR will leave it as zero. The LFSR must be initialized, i.e., seeded, to a nonzero value. When the LFSR holds 1 and is shifted once, its value will always be the value of the polynomial mask. When the register is all zeros except the most significant bit, then the next several shifts will show the high bit shift to the low bit with zero fill. For example, any 8-bit shift register with a primitive polynomial will eventually generate the sequence 0×80, 0×40, 0×20, 0×10, 8, 4, 2, 1 and then the polynomial mask.

3. Generating Pseudo-Random Numbers with LFSR

In general, a basic LFSR does not produce very good random numbers. A better sequence of numbers can be improved by picking a larger LFSR and using the lower bits for the random number. For example, if you have a 10-bit LFSR and want an 8-bit number, you can take the bottom 8 bits of the register for your number. With this method you will see each 8-bit number four times and zero, three times, before the LFSR finishes one period and repeats. This solves the problem of getting zeros, but still the numbers do not exhibit very good statistical properties. Instead you can use a subset of the LFSR for a random number to increase the permutations of the numbers and improve the random properties of the LFSR output. Shifting the LFSR more than once before getting a random number also improves its statistical properties. Shifting the LFSR by a factor of its period will reduce the total period length by that factor.
He relatively short periods of the LFSRs can be solved by XORing the values of two or more different sized LFSRs together. The new period of these XORed LFSRs will be the LCM (least common multiple) of the periods.
The unpredictability of the LFSRs can be increased by XORing a bit of ”entropy” with the feedback term. Some care should be taken when doing this—there is a small chance that the LFSR will go to all zeros with the addition of the entropy bit. The zeroing of the LFSR will correct itself if entropy is added periodically. This method of XORing a bit with the feedback term is how CRCs (cyclic redundancy checks) are calculated.
Chaos Theory: Chaos theory is the field of study in mathematics that studies the behavior of dynamical systems that are highly sensitive to initial conditions. Chaos is the science of surprises, of the nonlinear and the unpredictable. It teaches us to expect the unexpected. While most traditional science deals with supposedly predictable phenomena like gravity, electricity, or chemical reactions.
Lorenz equations: The Lorenz attractor discussed below is generated by a system of three differential equations such as: $\frac{dx}{dt} = \sigma y - \sigma x$; $\frac{dy}{dt} = \rho x - xz - y$; $\frac{dz}{dt} = xy - \beta z$, where x, y and z make up the system state, t is time, and σ, β & $\rho\sigma$, β & ρ are the system parameters. Five of the terms on the right hand side are linear, while two are quadratic; a total of seven terms.
Rössler equations: The defining equations of the Rössler system are: $\frac{dx}{dt} = -y - x$; $\frac{dy}{dt} = x + ay$; $\frac{dz}{dt} = b + z(x - c)$. Rössler studied the chaoticattractor with $a = 0.2$, $b = 0.2$ and $c = 5.7$ though properties of $a = 0.1$, $b = 0.1$ and $c = 14$ have been more commonly used since. Another line of the parameter space was investigated using the topological analysis. It corresponds to $b = 2$, $c = 4$ and a was chosen as the bifurcation parameter. How Rössler discovered this set of equations was investigated by Letellier and Messager.chaotic maps can be successfully used as a building block forcryptographic random number generators.

Requirements

1. Data stuffing with random numbers used in cryptographic system so that the intended receiver can only decipher the messages.
2. Noise is always random which need to be modeled.
3. Population study requires random selection of participants.

Solutions

1. To have random number generation policy.
2. Already, there are number of random generation techniques are available.

3. We want to generate RNG using chaos theory.

Methodology

1. To generate random number using chaos theory (**Lorenz equations & Rossler equations**).
2. We required to solve differential equations, but computer can not directly solve differential equations.
3. Therefore modified Euler's method which is numerical method of solving differential equations, will be used in our study.
4. C & MATLAB will be used to implement the tasks.
5. The Randomness generated sequence will be tested using packages like Diehard, NIST test suit, cryptx, ENT.

References

[1] https://en.wikipedia.org/wiki/Chaos_theory

[2] The Math Works, *Common generation methods*, Retrieved 2011-10-13.

[3] Donald Knuth, *Chapter 3-Random Numbers*, he Art of Computer Programming, Vol. 2, (1997).

[4] James Clewett, *Random Numbers*, Numberphile, Brady Haran.

[5] Some Basic Cryptographic Requirements For Chaos-Based Cryptosystemsgonzalo Alvarez Institution de FísicaAplicada, Consejo Superior de InvestigacionesCientíficas, Serrano 144, 28006 Madrid, Spain

[6] S. P. Vadhan, *Describing a dedicated Linux system call Pseudo randomness*, Foundations and Trends in Theoretical Computer Science, (2012).

[7] E. N. Lorenz, *Deterministic nonperiodic flow*, J. Atmos. Sci., 20(2)(1963), 130-141.

[8] Otto E. Rössler, *Chaotic behavior in simple reaction system*, Zeitschriftfür Naturforschung A., 31(3-4)(1976), 259-264.

[9] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, PA, Philadelphia, (1992).

[10] C. H. Vincent, *The generation of truly random binary numbers*, J. Physics E, 3(6)(1970), 594-598.

[11] The MathWorks, *Common generation methods*, Retrieved 2011-10-13.

[12] W. A. Wagenaar, *Generation of random sequences by human subjects: a critical survey of the literature*, Psychological Bulletin, 77(1)(1972), 65-72.

[13] Ran Halprin and Moni Naor, *Games for Extracting Randomness*, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, (2009).