International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:7, No:7, 2013

Deterministic Random Number Generators for Online Applications

Natarajan Vijayarangan and Prasanna S. Bidare

***Abstract*—**Cryptography, Image watermarking and E-banking are filled with apparent oxymora and paradoxes. Random sequences are used as keys to encrypt information to be used as watermark during embedding the watermark and also to extract the watermark during detection. Also, the keys are very much utilized for 24x7x365 banking operations. Therefore a deterministic random sequence is very much useful for online applications. In order to obtain the same random sequence, we need to supply the same seed to the generator.

Many researchers have used Deterministic Random Number Generators (DRNGs) for cryptographic applications and Pseudo Noise Random sequences (PNs) for watermarking. Even though, there are some weaknesses in PN due to attacks, the research community used it mostly in digital watermarking. On the other hand, DRNGs have not been widely used in online watermarking due to its computational complexity and non-robustness. Therefore, we have invented a new design of generating DRNG using Pi-series to make it useful for online Cryptographic, Digital watermarking and Banking applications.

***Keywords*—**E-tokens, LFSR, non-linear, Pi series, pseudo random number.

I. Introduction

A random number generator (RNG) is a computational or physical device designed to generate a sequence of numbers or symbols that lack any pattern. There are two main approaches to generating random numbers using a computer: Deterministic Random Number Generators (DRNGs) and True Random Number Generators (TRNGs) [1]. The approaches have quite different characteristics and each has its pros and cons. For our applications related to online E-commerce sectors, we are interested only DRNGs. In today's Internet, any new application and or service is to offer scalability as one of the main offering.

The next to follow is the on line security process to address one time security needs. Hence, the entire process of Crypto mechanisms, viz., key management needs to be dynamic and look ahead in nature. In such market, one of the main concerns is number management to undertake on line Cryptography. A robust Pseudo Random Number Generator (PRNG) suits the technological and mathematical relevance - like scaling, economics and adaptability. Some of immediate application is Internet based banking and transaction. Such requirements are best suited for Outsourcing to a strategic partner from Banking and Financial institutions.

The generation of pseudo-random numbers is an important and common task in computer programming. While Cryptography and certain numerical algorithms require a very high degree of apparent randomness, many other operations only need a modest amount of unpredictability. Weaker forms of randomness are used in hash algorithms and in creating amortized searching and sorting algorithms. In our applications, we need PRNGs to satisfy a strong amount of unpredictability, a truly deterministic and an efficient periodicity.

Also, we require large bit bandwidth numbers to satisfy hack resistance Crypto solutions. Hence, though the computing theory and hardware are known, but to generate copious amount random numbers on line and or on demand is quite a challenge. Further, we need to find out a process that needs to be sensitive to computational number crunching and also proportionate to time and power (computing as well electrical power).

For Indian / American / European financial institutions, everyday bank servers generate at least 30 million authentication tokens for their customers. In order to generate a large amount of random numbers required by the servers, we require a set of efficient PRNG algorithms having 3 main characteristics: Latency, Power and Speed of availability. However, the existing methods of PRNGs would find difficulty to fulfill all of the characteristics as well statistical properties. Therefore the present invention brings out a new deterministic (pseudo) random number generator (DRNG) to produce cryptographically secure and robust watermarking sequences for online applications.

II. Research Methodology

The DRNG in accordance with this invention is based on Leibniz Pi series [2], [5] that is secure and robust and it is useful for Cryptographic and Digital watermarking applications. Initially, there is taken a finite sequence S_n from a partial sum of Leibniz Pi infinite series ($\sum(-1)^{(n+1)}$ /(2n-1), where n =1,2,…). In order to achieve the randomness, the sequence S_n is passed into a layer of 3 operations: Irreversible shuffling, Non-linear function and Linear Feedback Shift Register (LFSR). In each operation, the entropy of S_n increases. After passing S_n through all three operations, the output satisfies statistical tests of randomness. Since the entire flow of operations is not bijective, it is difficult to predict S_n from the given output. Therefore, the output is random as well as secure. Some statistical properties of DRNG using Leibniz Pi and PN sequences have been observed and analyzed. Since the proposed scheme performs well in terms of correlation and randomness properties, it is recommended to use the proposed

Natarajan Vijayarangan and Prasanna S Bidare are with TCS Innovation labs, CTO Organisation, TCS Ltd, India (e-mail: n.vijayarangan@tcs.com, prasanna.bidare@tcs.com).

DRNG in Spread spectrum based Digital watermarking techniques as well as Cryptographic applications. Analogously, Ramanujan Pi value instead Leibniz Pi have been attempted in the invented DRNG and then the performance and outcomes are also good.

According to this invention, there is a method to find a new deterministic random number generator (DRNG) and also a system to implement the DRNG. The method of this invention comprises 4 primary operations in a sequential order: A) Selecting S_n from Pi series B) Irreversible Shuffling function C) Non-linear function and D) LFSR.

Typically, the operation A computes a finite sequence S_n from a partial sum of infinite Pi series. The operation B has one-way functions and it cannot be deshuffled. The operation C is not bijective and the operation D is bijective w.r.to the primitive polynomial used in LFSR as well as output of LFSR.

In one embodiment of the invention, the process A → B → C → D is not bijective which will be shown in the subsequent sections. Particularly, Ramanujan Pi series are used and tested as per the method and system in accordance with this invention. However, the system in accordance with this invention performs better than PN sequence. In particular, the seeds of this system are secure and is suitable for Information security and Digital watermarking applications [13].

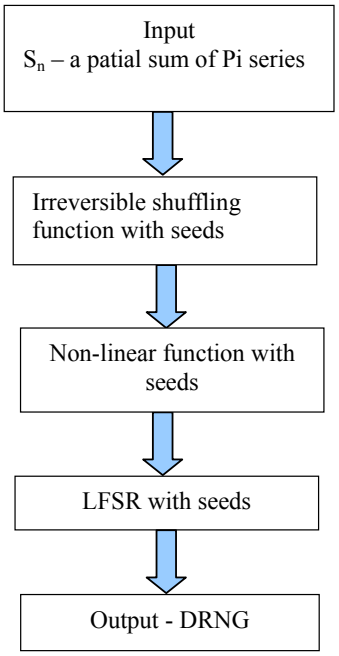

Fig. 1 Invented DRNG process

III. Description of the Invention

The method and system by which a deterministic random number sequence is generated are now described. The invention discloses a new DRNG process used to compute and generate a random sequence that is suitable for cryptographic and digital watermarking applications. The invented DRNG process consists of 4 operations: Selecting S_n from Pi series, Irreversible shuffling, Non-linear function and LFSR. Each operation is explained as follows.

A. Selecting S_n from Pi Series

A finite sequence S_n is selected from a partial sum of infinite Pi series $\sum a_i$ where n is as per the requirement of random numbers and each S_n consists of higher order decimal values. Then a sequence S is obtained where $S= \{S_1, S_2,\ldots,S_n\}$ for a large size n so that S_n converges to Pi value as $n \rightarrow \infty$ (where $S_n = a_1 + a_2 + \ldots + a_n$). For instance, the inventors have implemented and analyzed using a finite sequence S_n from a partial sum of Leibniz or Ramanujan Pi infinite series [3], [14].

Leibniz Pi series means $\prod/4 = \sum(-1)^{(n+1)}/(2n-1)$ (where n =1,2,…). Ramanujan Pi series means $1/\prod = (\sqrt{8} / 9801) (p(n) / q(n))$, where $p(n) = (4n)! (1103+26390\ n)$, $q(n) = (n!)^4\ 396^{4n}$ (where n = 1,2,…).

B. Irreversible Shuffling of Bits

The Shuffling function takes S_n as input and shuffles the bits of S_n followed by a one-way masking. The one-way masking function (like discrete logarithm / integer factorization) is used in shuffling function, so that it cannot be deshuffled. Hence the inverse of the shuffling cannot be easily accomplished. It follows that the shuffling function is cryptographically secure. Due to diffusion property of the shuffling function, the output gets random. Moreover, two seed values are used in the shuffling function: First seed is based on primitive polynomials and second seed on prime numbers. Users supply both seeds during the execution of shuffling operations [6].

For example, the input x is shuffled and transformed in the following manner. Here, the seed values are 0xFBDB1169 (32-bit primitive polynomial) and 0xFFFFFFFB (prime number).

Step 1
y=(x & 0x0000FF00) << 8 | (x >> 8) & 0x0000FF00 | x & 0xFF0000FF;
y=(y & 0x00F000F0) << 4 | (y >> 4) & 0x00F000F0 | y & 0xF00FF00F;
y=(y & 0x0C0C0C0C) << 2 | (y >> 2) & 0x0C0C0C0C | y & 0xC3C3C3C3;
y=(y & 0x22222222) << 1 | (y >> 1) & 0x22222222 | y & 0x99999999;

Step2
z= ((x)&(y))& 0xFBDB1169;
z=(z & 0x0000FF00) << 8 | (z >> 8) & 0x0000FF00 | z & 0xFF0000FF;
z=(z & 0x00F000F0) << 4 | (z >> 4) & 0x00F000F0 | z & 0xF00FF00F;
z=(x & 0x0C0C0C0C) << 2 | (x >> 2) & 0x0C0C0C0C | z & 0xC3C3C3C3;
z=(x & 0x22222222) << 1 | (z >> 1) & 0x22222222 | z & 0x99999999;

Step 3
$x = x^z \bmod 0xFFFFFFFB;$

C. Non-linear Function

The output of the shuffling function is the input to the non-linear function. Since the non-linear function diffuses the input, the entire sequence of bits increases entropy. Given non-linear function is irreversible so that input and output cannot be correlated. Further, seed values could also be incorporated in the non-linear function in order to improve the strength of the function.

For example, consider $f(n) = ((n^2 + n + c)$ & 0xFFFFFFFF) where c is a constant value provided by users. Then compute $g(m) = (m^n)$ mod p, where m = f(n) and p is prime. The output g(m) and input n cannot be correlated and the entropy of g(m) is relatively higher than n. As we discussed two functions f(n) and g(m), users could consider the constant c and prime p as seeds.

D. LFSR

The output of the nonlinear function is given as input to the LFSR [11], [12]. Primitive polynomials over $GF(2^n)$ are useful in the design of LFSRs for generating sequences of maximum periodicity. All generated primitive polynomials are highly dense as well as random (highly dense means more number of tabs used). In the LFSR operation, the given primitive polynomial acts as seed so that it is secure to generate different combination of sequence of bits.

Each time, an input value of n-bits is executed with a primitive polynomial of degree n and the linear shift process undergoes for different cycles. Thus, even if the input bits are identical, the output will be random.

For instance, the primitive polynomial used in LFSR is $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ over $GF(2^{32})$. This is an irreducible polynomial of degree 32 whose period is 2^{32}-1. Experimentally, it is tested that the output of LFSR round gets more random, when each 32-bit input is shifted with the given primitive polynomial for 4 to 15 cycles.

Inverse of LFSR can be obtained with respect to the same primitive polynomial used for LFSR as well as output of LFSR. Therefore LFRS is bijective. The inverse operation consumes a little bit time. For a large size of n, it is not an easy task to find the input from LFSR output without knowing a primitive polynomial employed in the LFSR operation.

E. Invented DRNG – Non-Invertible

The invented DRNG is capable of producing a random sequence with high entropy. The process starts from S_n that undergoes Irreversible shuffling, Non-linear and LFSR operations and produces the output as a deterministic random number sequence. It is theoretically and practically impossible to predict n and S_n from a given random sequence due to nonlinear and one-way properties involved in the DRNG process.

The tool in accordance with this invention can be used in Cryptographic applications and Digitial watermarking. Unlike DRNG, the sequence PN can be realized by LFSR with Exclusive OR operations and is good for spread spectrum communications. When an attacker comes to know what kind of polynomial is used in the sequence PN, the input and output processes of PN could be judged – a vulnerable attack. Whereas in the invented DRNG, even the attacker comes to know the core operations in LFSR, it is difficult to judge the behavior of nonlinear and discreet logarithm properties involved in the DRNG process. Therefore, the invented DRNG is cryptographically secure, performs well in terms of robustness in watermarking applications and E-banking applications due to its correlation properties.

IV. Analysis of DRNG (invented) and PN

PN possesses good Autocorrelation and Cross correlation properties. It is well suited for Digital Watermarking, but not for Cryptographic applications. Sometimes, the PN sequences may not have good entropy values. Whereas the invented DRNG performs well in terms of *Autocorrelation, Cross Correlation, Entropy, Compression, Frequency, Arithmetic Mean, Serial Correlation, Chi square distribution and Monte Carlo value tests*. The invented DRNG is compared with PN and Alternating sequences as follows:

TABLE I
Comparison of Different Sequences

Alternating sequence	Invented DRNG	PN
The sequence follows a pattern	No pattern	No pattern
Not random	Random	Random
Not at all secure	Secure for Cryptographic and watermarking applications	Secure for watermarking applications. It can be used for Cryptographic applications with risk.
Suitable for spread spectrum based digital watermarking application	Suitable for spread spectrum based digital watermarking applications	Suitable for spread spectrum based digital watermarking application.
Recovery of watermarked image is good with error level 2-3%	Recovery of watermarked image is good with error level 5-6%	Recovery of watermarked image is good with error level 8-9%
Image smoothness is good	At par	At par

Let us discuss the implementation results of this invention. The entropy and randomness tests [7], [9] have been performed on the input and output files of the invented DRNG. The entropy of the output file is at maximum level (7.9 bits per byte) and it was always higher than that of the corresponding input file.

The following results are analyzed using Statistical tests on the input and output files of the invented DRNG.

TABLE II
INPUT FILES OF DRNG PROCESS WITH STATISTICAL PROPERTIES

Input file size (KB)	Entropy (bits per byte)	Optimum compression	Chi square distribution	Arithmetic mean value of data bytes	Monte Carlo value for Pi with error	Serial correlation coefficient
Leibniz Pi series = 192	4.31	200000 byte file by 46%	10209131.42	68.1466 (127.5 = random)	3.6287162 87 (error 15.51%)	0.355382 (totally uncorrelated = 0.0)
Leibniz Pi series = 391	4.33	400000 byte file by 45%	20325590.43	69.9801 (127.5 = random)	3.6302763 03 (error 15.56%)	0.383419 (totally uncorrelated = 0.0)
Ramanujan Pi series = 128	3.68	130790 byte file by 54%	2659228.87	48.9852 (127.5 = random)	4.0000000 00 (error 27.32%)	0.267845 (totally uncorrelated = 0.0)
Ramanujan Pi series = 256	3.66	263188 byte file by 54%	5377266.89	48.9443 (127.5 = random)	4.0000000 00 (error 27.32%)	0.262759 (totally uncorrelated = 0.0)

TABLE III
THE CORRESPONDING OUTPUT FILES OF DRNG PROCESS WITH STATISTICAL PROPERTIES

Output file size (KB)	Entropy (bits per byte)	Optimum compression	Chi square distribution	Arithmetic mean value of data bytes	Monte Carlo value for Pi with error	Serial correlation coefficient
192	7.999012	200000 byte file by 0%	274.08	127.5066 (127.5 = random)	3.12987129 9 (error 0.37%)	-0.003409 (totally uncorrelated = 0.0)
391	7.999509	400000 byte file by 0%	272.38	127.5831 (127.5 = random)	3.13629136 3 (error 0.17%)	0.001422 (totally uncorrelated = 0.0)
128	7.998604	130790 byte file by 0%	254.3	127.4260 (127.5 = random)	3.16689604 6 (error 0.81%)	0.004917 (totally uncorrelated = 0.0)
256	7.999243	263188 byte file by 0%	275.66	127.6200 (127.5 = random)	3.13322998 4 (error 0.27%)	-0.000643 (totally uncorrelated = 0.0)

V. INDUSTRIAL APPLICATIONS

The invented DRNG described above finds a number of applications in Information Security [4], Digital watermarking [8], [10] and online banking. Some specific areas where our process can be applied are:

1. Session key generation
2. Signature protocols
3. Client-Server protocols
4. Encryption
5. Spread spectrum communications
6. Digital Signal Process
7. E-tokens

VI. CONCLUSION

Unlike, deploying regular approach such as linear congruential methods and probability distributions, we created a new design in PRNG [13] and implemented it for online applications. Keeping in mind, let us assume a great demand in future, our global business models would need at least billions of random sequences for e-transactions and authentication tokens per day. Hence, we have been working on the generation of billions of random sequences using Pi series.

REFERENCES

[1] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, Handbook of Applied Cryptography, CRC Press, 2001.
[2] Beckmann, P. A History of Pi, 3rd ed. New York: Dorset Press, 1989.
[3] Berndt, B. C. Ramanujan's Notebooks, Part IV.New York: Springer-Verlag, 1994.
[4] Douglas Stinson, Theory and Practice:Cryptography, CRC Press 2002.
[5] George Marsaglia, On the randomness of Pi and other decimal expansions, InterStat, #5, October,2005.
[6] Henry S. Warren, Jr, Hackers Delight, Addison-Wesley, 2002.
[7] John Walker, ENT, A pseudorandom Number Sequence Test Program.http://www.fourmilab.ch/random/
[8] Michael Arnold, Martin Schmucker, Stephen D. Wolthusen, Techniques and applications of Digital Watermarking and Content Protection, Artech House Inc, 2003.
[9] NIST special publication 800-22 (2001): A statistical test suite for random and pseudorandom number generators for cryptographic applications, National Institute of Standards and Technology. http://csrc.nist.gov/groups/ST/toolkit/rng/documents/SP800-22b.pdf.
[10] Rafael C. Gonzalez, Richard E.Woods, Steven L. Eddins, Digital Image Processing Using MATLAB, Pearson Education Inc, 2007.
[11] Vijayarangan N. and Kasilingam S., Random number generation using primitive polynomials, Proceedings of SSCCII-2004,Italy.
[12] Vijayarangan N. and Vijayasarathy R., Primitive polynomials testing methodology, Jour. of Discrete Mathematics & Cryptography, Vol.8, No.3, pp. 427-435, 2005.
[13] Vijayarangan N. and Srinivasa Rao C., Deterministic random number generator for cryptography and digital watermarking, USPTO 20090193065.
[14] Shu-Ju Tu and Ephraim Fischbach, A study on the randomness of Pi, International Journal of Modern Physics C, 16, No. 2, 281–294, 2005.