I J C T A, 8(5), 2015, pp. 1817-1822

A Fast Color based Nonlinear 8 bit Pseudo-Random Number Generator Enabling Diffusionfor Lightweight Cryptographic Applications

Panchami.V*, Varghese Paul, Amithab Wahi*** and Ebin Raju******

Abstract: Pseudo-Random number generator (PRNG) plays an important role in cryptographic applications. In lightweight cryptographic applications and devices have various constrains such as low battery power, memory. All pseudo-random generators consumes battery power, memory and it even slowdowns the cryptographic devices. In this paper we designed a fast and simple nonlinear Pseudo-Random number generator, PANCH for lightweight cryptographic applications to enable diffusion. It can produce 2^8 unique 8 bit nonlinear numbers without overlapping in one seed. The seed act as the key for the PANCH and it ranges from 0 to 2^{32}. Due to its nonlinearity the sequence is not at all predictable. We have conducted 2 statistical tests Diehard battery test and Big Crush tests for uniqueness, randomization and find no failures and have noticed that is much faster than other PRNG used in various cryptographic applications and will resist from all known attacks. Our algorithm has wide range of applications such it is used to generate substitution boxes in encryption algorithms in cryptography.

General Terms: Security, lightweight Cryptography, Encryption, Decryption, Pseudo-Random number generator

Keywords: Pseudo-Random number Generator(PRNG), Nonlinear functions, Cryptography, Diffusion, ARGB

1. INTRODUCTION

Pseudo-Random Number Generators(PRNGs) are one of the basicelement in cryptographic algorithms and security protocols for generating the sequence of random numbers. Pseudorandom number generator which is used for cryptography be implemented in hardware device or software for computing, for generating asequence of numbers. The initial inputs of these generators are called seeds, since our application area is cryptography these seeds help to produce same sequence of numbers. To implement PRNG, there are two methods, using hardware device like shift registers, clocks, Logical gates, but there many constrains such as space to store the random values and computational overheads are high in hardware implementations. In software implementation the memory space is not at all a problem and for reducing computational overhead the random generators can use binary operations such as XOR, AND, NOT, circular shift. Based on the random number generation the pseudorandom number generator are of two types linear and nonlinear. In linear based generators the distribution of random numbers should be in a linear or in uniform manner. But in nonlinear the distribution of random numbers is in a nonlinear pattern. Next is dependent up on the occurrence of random numbers there are two types,True random number generatorsand unique random number. True random generators are those which produces the random numbers having collision, that

* Assistant Professor Computer Science Toc H Institute of Science and Technology, Arakunnam, Ernakulam, India, *Email: panchamam036@gmail.com*

** Professor CS/IT, Research Toc H Institute of Science and Technology, Arakunnam, Ernakulam, India, *Email: vp.itcusat@gmail.com*

*** Professor Information Technology Bannari Amman Institute of Technology, Sathyamangalam, Erode, Tamilnadu, *Email: awahi@rediffmail.com*

**** MTech Scholar Computer Science TocH Institute of Science and Technology, Ernakulam, Kerala, *Email: ebinraju@gmail.com*

means the occurrence of random numbers are more one in the sequence. But in unique random number generators there is no collision in random sequence, there is no duplicate values of a number. The true random number generators are more secure than the unique random number generator, because the probability of prediction is much higher in true random number generator.If we apply nonlinearity in unique random number generators then it will the most efficient and secure generators in producing random numbers in cryptographic applications.

Red, Green, Blue (RGB) Color concept is used in generating the random numbers. In RGB color space each color is represented as 8-bit, here we add an extra bit for intensity too. So each color is represented as 32 bit. For each 8 bit color we produce a random number. Our algorithm is represented as ARGB color model, which generates 4, 8 bit random numbers. Our PANCH algorithm can be applied not only to cryptographic application but to generate unique 4, colors too.

We designed our algorithm for lightweight cryptographic devices, these devices have many constraints like memory, speed, power consumption and battery consumption. For constrained devices, while usingpseudo-random number generators and standardcryptographic algorithms consumes high battery consumption and power consumption, the device become slow too.There are various research going on the area of lightweightcryptography studies new algorithms to overcome these problems while designing it.

In this paper section 2 explain about theliterature survey of existing pseudo-random generators, section 3 explains the notations and theoretical background of the new pseudo-random number generator, section 3 deals with proposed system PANCH, in results, test and analysis is explained in section 4 and section 5 deals with conclusion and future work.

2. LITERATURE SURVEY

A good PRNG [1] can be designed and built following a few simple guide lines. The characteristics of good PRNG are as follows: a) Since processing power is not limited to the extent as it was a few decades ago, PRNG algorithms [3,4] must still be short and efficient. This will allow pseudo random numbers[3,4] to be generated in only a few clock cycles to allow the processor to continue with the main calling function or program[4]. b) Since PRNG's are in the form of a mathematical function [5], it is noted that they will, at some stage, begin to repeat themselves. It is this period of a PRNG that must be as long as possible [3,4]. c) There are statistical tests in use that can test the possibility of randomness [1] [3] with high levels of accuracy. The sequence produced from a PRNG should be checked against these tests, and pass them[1] [3]. d) By analyzing the outputs of a PRNG [4], it should not be possible to predict the next number that will be generated [4]. e) A random number sequence, in its binary representation [4], must have, on average, an equal amount of 1's and 0's. Furthermore, there must be no noticeable patterns in the bit string.[1] [4]. f) A PRNG must be seeded with a value [4], and given the same value, the same sequence of numbers must be produced.

The first reliable PRNG algorithm was proposed by D. H. Lehmer in 1949 [1], called the Linear Congruential Generator (or Linear Congruential Method, LCM). This method has been one of the most well known and widely used methods for generating random sequences. However, this method is not without flaws. It is well know that the sequence generated forms a lattice structure in 3-space. One of the most famous poor PRNG was IBM's RANDU which did not have a full period and it had some extremely non-random characteristics [7]. Knuth described it as "really horrible". "It is well-known that all linear congruential generators suffer from the inherent flaw that, in 3-space1 for example, the points (Z_i, Z_{i+1}, Z_{i+2}), (Z_{i+1}, Z_{i+2}, Z_{i+3}), (Z_i, Z_{i+3}, Z_{i+3}), all fall on a finite – and possibly small – number of parallel (hyper)planes." [5]. The LCM is of the form of a recursive method and is as follows: $x_n + 1 = (a * x_n + c) \bmod m$, where m is the modulus; a is the multiplier; c is the increment and $X0$ is the starting value, or seed. If a, c and m are chosen correctly, it is possible for the generator to have a maximum period of length m [1].

A very popular shift register generator is the Mersenne Twister2 [1], developed in 1997 by Makota Matsumoto and Takuji Nishimura from Keio University, this algorithm produces a sequence of 219937 – 1 numbers and has 623-dimensional equidistribution (compared to the 5-dimensional equidistribution of LCM generators). This generator is not suitable for cryptographic applications due to the fact that is it possible to analyze the out and recognize the numbers as being non-random. The developers of this algorithm do however advise that a secure hashing function be used with the output or a simple linear transformation to help get around this issue. It is, however, a very good PRNG for other applications such as Monte Carlo simulations. This PRNG is fast becoming the PRNG of choice for such application. [6] As stated earlier, all numbers produced by a PRNG are dependent on previous numbers produced, hence the fact that they are deterministic. The LCM and similar type PRNGs(attempt to) hide this shortfall by using the mod operator. If one obtained a number from a random sequence and also knew the inner workings of one of these generators, it would still be almost impossible to calculate previous numbers in the sequence. This is because the mod operator disregards some of the seeding value. Example: Say we have a seed: xn = 1234, a = 105, c = 0, and m = 106. By using the LCM method: $x_n + 1 = (a^* x_n + c) \bmod m$, we are left with a result of 400 000. As one can see, we are left with a result that has "lost" some of its data. There are of course brute force methods to recover the seed [10], but with a good choice of seed, a, c and m values, this task can be made tedious.

3. NOTATION AND THEORY

Let $F_2 = GF(2)$ be the finite field with two elements {0, 1}. In PANCH we use the field operations are "exclusive or" as $\oplus$, "AND" as $\wedge$, "NOT" as $\neg$ and "circular shift" as <<. In binary field {0, 1}, 0 is treated as false value and 1 is treated as true value like "ON" and "OFF" states.

Theorem *In order that a nonsingular $n \times n$ binary matrix T produce all possible non-null $1 \times n$ binary vectors in the Sequence $\beta, \beta T, \beta T^2, \ldots$ .for every non-null initial $1 \times n$ binary vector β, it is necessary and sufficient that, in the group of nonsingular $n \times n$ binary matrices, the order of T is $2^n - 1$.*

Proof: First, the necessity: If the period of $\beta, \beta T, \beta T^2, \ldots$ is $k = 2^n - 1$ then for $\beta T^k = \beta$ every $1 \times n$ binary vector β, so the null space of the matrix $T^k + I$ is the whole space, and thus $T^k + I$ must be the zero matrix, that is, $T^k = I$. If $T^j = I$ for some $j < k$, then the period of $\beta, \beta T, \beta T^2, \ldots$ would be less than $2^n - 1$. Then the sufficiency: If the order of T is $k = 2^n - 1$, then the matrices $T, T^2, T^3 \ldots T^k$ are nonsingular and distinct, and through the characteristic polynomial of T and Euclid's algorithm, each of them can be represented as a polynomial in T of degree $< n$. Since there are $k = 2^n - 1$ non-null polynomials in T of degree $< n$, they must be, in some order, the distinct nonsingular matrices $T, T^2, T^3 \ldots T^k$. In particular, if a polynomial in T is a singular matrix, then it must reduce, through T's characteristic polynomial, to the zero matrix. It follows that the period of $\beta, \beta T, \beta T^2$ must $k = 2^n - 1$, because $\beta T^j = \beta$ for some non-null _ and $j < k$, would mean that $T^k + I$ is singular.

Proposition 3.1 Let $x_0, x_1, \ldots. x_2$, be a list of 2^t-bit values, $t < n$, such that every value appears 2^{nt} times, except for 0, which appears $2^{n\ldots\ldots t}$ times, except for 0, which appears $2^{n\ldots\ldots t} 1$ times. Then, for every fixed bit k the associated sequence has period $2^n - 1$.

Proof. Suppose that there is k a and a $p \mid 2^n 1$ such that the k th bit of $x_0, x_1, \ldots. x_2$, has period p (that is, the sequence of bits associated with the k-th bit is made by $(2^n 1)/p$ repetitions of the same sequence of p bits). The k-th bit runsthrough $2^{n\ldots\ldots t} 1$ zeroes and $2^{n\ldots\ldots 1}$ ones (as there is a missing zero in the output sequence). This means that, $(2^n 1)/p \mid 2^{n\ldots\ldots 1}$, too, as the same number of ones must appear in every repeating subsequence, and since $(2^n 1)/p$ is odd this implies $p = 2^n - 1$.

Corollary 3.1 Every bit of the output of PANCH generator has full period.

4. PANCH

In this section we present PANCH produces 2^8 unique 8 bit random numbers with one seed. The input of PANCH is the seeds which ranges from 0 to $2^{8.}$ So with four different seeds PANCH produces four, 8 bit sequence of 2^8 unique random numbers. Since PANCH is applicable for cryptographic applications, the basic building block of our algorithm are binarys operation such as XOR, AND, NOT and circular shift operations. These operations gives fastness to PANCH.

4.1. PANCH Initialization

PANCH is have 4 inputs, these inputs act as seeds $SEED_A$, $SEED_R$, $SEED_G$, $SEED_{B.}$ The range of these seeds varies from 0 to 2^{32} bits. Then we apply a mod function to these four seeds to get the first four random numbers A, R, G, B which are of 8 bit. These 4 numbers A, R, G, B act as the initialization value of PANCH.

Algorithm 4.1 INTIALIZATION ($SEED_A$, $SEED_R$, $SEED_G$, $SEED_B$)

$a \leftarrow SEED_A \bmod 2^8$

$r \leftarrow SEED_R \bmod 2^8$

$g \leftarrow SEED_G \bmod 2^8$

$b \leftarrow SEED_B \bmod 2^8$

RETURN (a, r, g, b)

4.2. PANCH Algorithm

PANCH is designed for producing four unique random numbers of 8 bit numbers with four seeds $SEED_A$, $SEED_R$,$SEED_G$,$SEED_B$. After initialization process PANCH produces the first four random numbers *a*, *r*, *g*, *b*. These four numbers will be the input to the PANCH_ROUND_FUCTION, for the next 64 Rounds. In our algorithm PANCH uses four functions to get random numbers with the binary operations AND, NOT, XOR and left circular shift. These binary operations is very simple, fast and efficient, by using these operations in PANCH ensures simplicity, fastness and efficency. PANCH is designed to use in cryptographic applications, mainly in lightweight devices.

Algorithm 4.2 PANCH_ROUND_FUNCTION

FOR $i \leftarrow 2$ TO 64 DO

RAND_COLOR (a, r, g, b)

$a \leftarrow [(a \wedge r) \oplus (r \wedge g) \oplus (a \wedge c)] + a << 8$

$r \leftarrow r < 16 \oplus g$

$g \leftarrow g << 4 \oplus d$

$b \leftarrow [(r \wedge g) \oplus (\neg g \wedge b)] + r << 6$

RETURN (a, r, g, b)

5. RESULT OF PANCH GENERATOR

5.1. Results of Diehard Battery Test and Big Crush Test on PANCH

The diehard tests are a battery of statistical tests for measuring the quality of a random number generator. They were developed by George Marsaglia over several years and first published in 1995 [11].PANCH generators have no systematic failure on both statistical tests,but only about half of the reverse generators

have no systematic failure. Moreover, thedistribution of standard generators degrades smoothly. The Table.1 shows the results of diehard test on PANCH and found no systematic failures. And the Table.2 shows the results of Big Crush test on PANCH and it also found no systematic failures.These generators were few enough so that we could apply both Crush and Dieharder. We have noticed that in Figure. 1 we compare instead the two scores (Crush and Dieharder) available. The mostremarkable feature is there are no points in the upper left corner: there is no generator thatis considered good by Crush but not by Dieharder. On the contrary [11], Crush heavily penalizes [2](in particular because of the score on the reverse generator) a large number of generators.The generators we will select in the end all belong to the small cloud in the lower left corner,where the two test suite agree.

Table 1
Results of Diehard on PANCH

Algorithm	*Failures*			*W*
	S	*R*	+	
(21, 17, 2, 243)	67	56	123	281
(155,32, 227, 181)	77	54	131	59
(110, 231,170, 60)	66	66	132	65
(217, 86,105,79)	60	75	135	89
(181,163, 4, 211)	63	74	137	155
(127, 112, 243, 59)	74	69	143	233
(204, 253, 164, 100)	86	58	144	79
(138, 27, 65, 155)	82	62	144	99
(78, 34, 145, 93)	78	69	148	275
(37, 213, 209, 196)	85	64	148	363
(71, 107, 74, 160)	65	86	152	69
(232, 39,242, 48)	84	69	154	265
	88	65	157	81
	77	81	160	167
	82	80	162	77
	85	78	163	255
	92	75	167	111

Table 2
Results of Big Crush on PANCH

Algorithm	*Failures*			*W*
	S	*R*	+	
(21, 17, 2, 243)	34	35	69	45
(155,32, 227, 181)	36	35	71	187
(110, 231,170, 60)	35	37	72	441
(217, 86,105,79)	34	39	73	103
(181,163, 4, 211)	40	34	74	567
(127, 112, 243, 59)	41	33	74	195
(204, 253, 164, 100)	39	35	77	291
	43	34	79	241
	42	37	80	177
	38	43	81	49

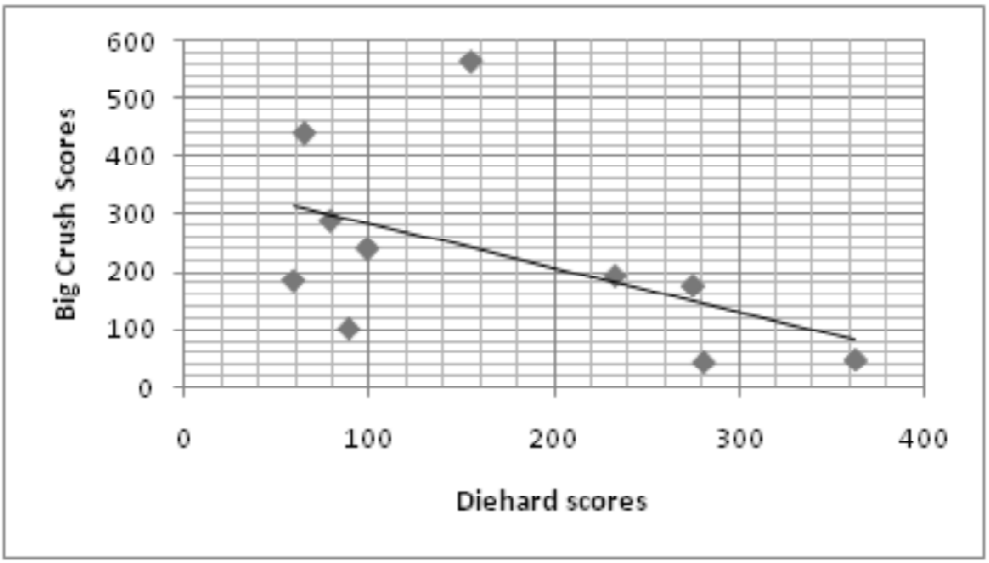

Figure 1: Comparison of Diehard test and Big Crush Test

6. CONCLUSION AND FUTURE WORK

A nonlinear Pseudo-random generator PANCH is proposed and tested on the basic we conclude that the generator is an excellent choice for a lightweight devices in cryptographic and non cryptographic applications. Our PANCH is based on recursion and have four seeds.The two statistical tests shows best results (in BigCrush and in Diehard), and its period is so large and the probability of overlap-ping sequences is practically zero. Nonetheless, thestate space is reasonably small, so that seeding it with high-quality bits is not too expensive,and recovery from states with a large number of zeroes happens quickly. PANCH is designed with basic logical functions such as XOR, AND, NOT etc. So it is lightweight in nature too. The generator isalso blazingly fast (it is actually the fastest generator we tested). In PANCH we produces 64, unique random numbers with one seed, thus with 3 seeds we produced such 64 random numbers. In future anyone can extend this work to 255 unique, 8 bit random number with one seed which length varies from 0 to 255.

References

[1] Andrew Cronwright, “Literature Review: Pseudo Random Number Generation and Random Event Validation through Graphical Analysis”602c2954.

[2] Leonard Marsaglia, George, 1995, The Marsaglia Random Number CDROM, with The Diehard Battery of Tests of Randomness, produced at Florida State University under a grant from The National Science Foundation.

[3] Jansson, B., Random Number Generators, 1966.

[4] Atreya, M., RSA Security article: Intoduction to Cryptography (part 4): Pseudo Random Number Generators, November 2000

[5] Park, S. K., Miller, K. W., Random Number Generators: Good Ones Are Hard to Find, Communications of the ACM, October 1988, Volume 31, Number 10.

[6] Matsumoto M., Nishimura T., Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator, January 1998, Keio Universit, ACM Transactions on Modelling and Computer Simulation, Volume 8, Number 1, Pages 3-30.

[7] Entacher, K., Linear Congruential Generator: LCG, Online: http://crypto.mat.sbg.ac.at/results/karl/server/node3.html, June 2000.

[8] Entacher, K., Bad Subsequences of Well_known Linear Congruential Pseudorandom Number Generators, January 1998, University of Salzburg, ACM Transactions on Modelling and Computer Simulation, Volume 8, Number 1, Pages 61-70.

[9] Kelsey, J., Schneier, B., Ferguson, N., Yarrow-160: Notes on the Design and Analysis of the Yarrow Cryptographic Pseudorandom Number Generator, Sixth Annual Workshop on Selected Areas in Cryptography, Springer Verlag, August 1999.

[10] Ferguson, N., Schneier, B., Practical Cryptography, 2003, pages 155-184.

[11] SebastianoVigna, “An experimental exploration of Marsaglia’sxorshift generators, scrambled”.