# A PSEUDO RANDOM NUMBER GENERATOR USING POINT OPERATIONS ON ELLIPTIC CURVES

Tapas Kumar Ghosh

Assistant Professor

Department of Computer Science

Bankura Sammilani College, Bankura, India

Abstract-- A pseudo random number generator which uses points on an elliptic curve $E_p$ (a, b) over a finite field $F_p$ (p is prime) is proposed. The PRNG outputs a sequence $<N_i>$ of numbers obtained by merging x and y coordinates of random points on E. As seed a random point P on E is chosen first, the subsequent numbers are generated from it. The generated sequence of numbers has passed the FIPS 140-2 specified statistical tests.

**Keywords –** Pseudo Random number generator, Elliptic Curve.

**Section-I- Introduction**

Unpredictable and uniformly distributed sequences of numbers that cannot be efficiently regenerated are referred to as Random numbers. Simulation of random number sequences is a very popular and well studied field of mathematics and computer science. Such a random simulation often called pseudo random number generator (PRNG), is an algorithm which generates a sequence of numbers whose properties approximate the properties of sequences of Random numbers. Pseudo random number generators generate sequences which are not truly random because the complete sequence can be generated again using the seed. But a true random number generator generates sequences which can never be regenerated. True Random numbers may come from a weak physical process like coin flipping, dice throwing, thermal noise, nuclear decay or quantum process such as agitation of electrons. In today's modern world Random numbers are used in various disciplines like bioinformatics systems, statistical sampling, and obviously in various applications of cryptography such as key generation, encryption and obtaining digital signature. For cryptographically secure Random numbers it is to be noted that period of the sequence must be very long to counter various types of attacks like brute force attack, man in the middle attack, etc. This paper suggests a cryptographically secure pseudo random number generation algorithm with a long period using points on an elliptic curve. After this brief definition in section-I, related works will be summarized in section-II. In section-III proposed model of PRNG is given. Section-IV will discuss several statistical tests performed on the output of the proposed PRNG. Section-V concludes the paper.

**Section-II - Related Work**

In 1985 Neil Koblitz [1] & Victor Miller [2] independently arrived at the decision that elliptic curves can be used to develop a strong secure public key cryptosystem. ECC applications are currently the subject of extensive investigation as means for increasing security in transmission and reception of data over an insecure communication channel. With much less key size ECC may provide same level of security as that of RSA [3]. Applications like smart cards, embedded systems etc, which has limited space for implementation of modular computation, widely uses elliptic curve public key cryptography over finite fields. Recently algorithms for the elliptic curve point operations, like add, double, multiply can be implemented easily in both hardware and software. Some Pseudo Random Number Generators which uses Elliptic Curve Point Operations are discussed below.

**A.** The linear congruential generator: In 1994, Sean Hallgren [4] discussed about a pseudo-random number generator based on

a subgroup of points of an elliptic curve defined over GF (q) where q is prime. This generator is known as the Linear Congruential Generator on Elliptic Curves (EC-LCG). Let E be a non-singular elliptic curve defined over a prime finite field $F_q$ that is a rational curve given by the simplified Weierstrass equation as:

$$E: y^2 = x^3+ax+b \pmod q \text{ for some a, b} \in F_q \text{ with } 4a^3+27b^2 \neq 0 \pmod q.$$

We know that the set of points (including the special point O at infinity) on the elliptic curve forms an Abelian group with the geometric point addition rule on cubic curves as the operation between two points with O as the identity element. For a given point $G \in E(F_q)$, the EC-LCG is a sequence $\langle V_n \rangle$ of points on the elliptic curve E defined by the relation:

$$V_n = V_{n-1} + G = nG + V_0 : n \in N$$

Where $V_0 \in E(F_q)$ is the initial value or seed.

DUAL_EC_DRBG (Dual Elliptic Curve Deterministic Random Bit Generator): It was one of the four cryptographically secure Pseudo Random Number Generator standardized in NIST SP 800-90A [5]. It makes use of two points P(x, y) & Q(x, y) on a non super singular elliptic curve $E(F_q)$ for generation of random numbers. In this generator the iteration key k is produced successively as $k_i = x([k_{i-1}]P)$ on the other hand the output sequence is generated as $t_i = x([k_i]Q)$ where t is the truncation function. Due to a potential backdoor this generator was withdrawn in 2014.

**B.** The Pseudo Random Bit Sequence Generator-A: A modified Dual-EC generator with increased periodicity named PBSG-A is published in [6]. In PBSG-A, the iteration key k is modified as: $k_{i+1} = [k_i]G+[i]C$ where C=x(e[G]) and "e" is the seed value. In addition to two point multiplication operations the modified algorithm requires a finite field multiplication of iteration number "i" and the value "C" to be carried out in each iteration. This increases both the hardware complexity and the time complexity of the system.

**C.** The Pseudo Random Bit Sequence Generator-B: This generator is also presented in [6] uses a modification of the EC-LCG to make the periodicity independent of the order of the point so that the output sequence does not have any symmetric property to make the cryptanalysis easier. For better security the authors assumed that the seed value and the initial point to be kept secret.

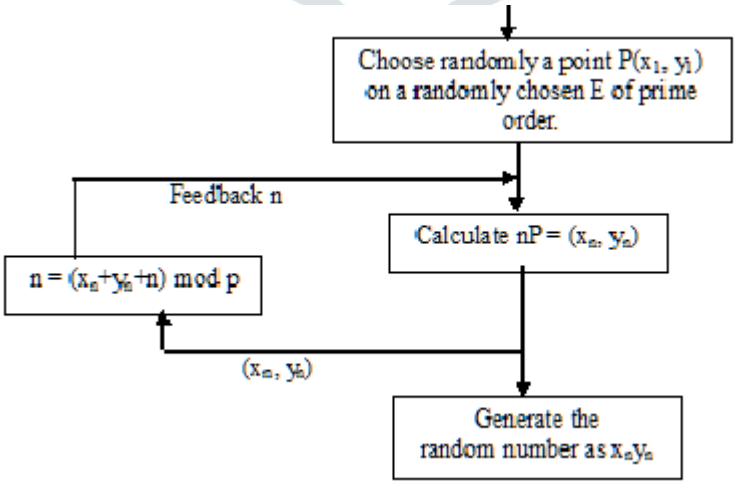
**Section-III – Proposed Pseudo Random Number Generator**

The simplified Weierstrass form of a nonsupersingular elliptic curve E over a field K [7] is defined by the equation:

$$E: y^2 = x^3+ax+b \pmod p \text{ where a, b} \in K \text{ and } \Delta = 4a^3+27b^2 \neq 0$$

In elliptic curves, multiplying a point P with an integer n, can be implemented as, successive n-1 times addition of P with itself. This is called nP operation. Using add double rule this can be done efficiently. In our proposed system we use elliptic curves of prime order only and over a prime field. A plenty of such curves exist under any prime field.

The diagram shown below will explain the functioning of proposed generator.




The above module works as follows:

In the initialization module the elliptic curve E over a prime field $F_p$ is chosen. E should be chosen in such a way so that order of

E is prime. Take a point P($x_1$, $y_1$) on E randomly. Next step is Set n=1. The next module compute nP and generates the sequence of random numbers by merging the values of P($x_n$ , $y_n$) as $x_ny_n$. In each iteration a new seed is generated from this module using the formula-

$$\text{Next value of } n = (x_n+y_n+n) \%p$$

Each time this new value of n is fed back into the nP computing module to get the desired randomness. The choice of a prime order curve gives us the facility to choose any point P as a generator of the underlying group of points of E. As our aim is to create a random number generator using point operations on E, such a choice of P may utilize all points of E in the proposed generator resulting a long period sequence of pseudo random numbers. However there is a very little probability that nP will give the point at infinity. That case could be handled by selecting another random point on E. Following [8] the least period of the generator can be calculated as:

Let in the $r^{th}$ cycle we get next value of n as

$$n_{r+1} = ( x_r+y_r+r)\%p$$

Where ($x_r$, $y_r$) is the point obtained from nP module.

Similarly in the $s^{th}$ cycle with s>r we get next value of n as

$$n_{s+1} = ( x_s+y_s+s)\%p$$

Where ($x_s$, $y_s$) is the point obtained from nP module. Let us suppose that the values $x_ry_r$ and $x_sy_s$ be the same. In order to get the next values $x_{r+1}y_{r+1}$ and $x_{s+1}y_{s+1}$ as same we must have

$$n_{r+1} = n_{s+1}$$

$$\text{i.e. } ( x_r+y_r+r)\%p = ( x_s+y_s+s)\%p$$

$$\text{i.e. } r \equiv s\%p$$

$$s = r+qp.$$

So the sequence will repeat only after p cycles.

**Section-IV - Statistical Tests**

Major problems in pseudo random number generators are their failure in statistical pattern detection tests. Existence of correlation in successive values, lack of uniformity in distribution, output sequence with poor dimensional distribution etc are the common problems. This section checks whether the experimental results of randomness properties of the sequences generated in previous section conforms to the standard FIPS 140-2 specified tests. We check the randomness properties using the following five basic statistical tests from [9] & [10] as:

1. Frequency (Monobit) Test: It tests the ratio of 0's and 1's for the total sequence. This test tells us that whether the number of ones ($n_1$) and zeroes ($n_0$) are roughly the same as it would be expected from that of a truly random sequence of length n. The test statistic is obtained as

$$X_1 = \frac{(n_0-n_1)^2}{n}$$

2. 3-bit Poker Test: This test checks whether non overlapping 3 bit subsequences, of the bit sequence, of generated numbers appear approximately, the same number of times. The statistic used is

$$X_2 = \frac{2^m}{k}\left(\sum_{i=1}^{2^m} n_i^2\right) - k$$

   Where m=length of each subsequence, $n_i$ is frequency of each subsequence, k is number of subsequences.

3. Serial Test: The purpose of this test is to determine whether the number of occurrences of $2^m$ m-bit overlapping pattern is approximately the same as would be expected for a random sequence. For a two bit test[3] the statistic used is

$$X_3 = \frac{4}{n-1}\left(n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2\right) - \frac{2}{n}\left(n_0^2 + n_1^2\right) + 1$$

   Which approximately follows a $\chi^2$ distribution with 2 degrees of freedom if n≥21. [9]

   In one of our sample test mentioned last of this section we have obtained the values as 4887, 5045, 5045, 5118 respectively for $n_{00}$, $n_{01}$, $n_{10}$, $n_{11}$. So value of $X_3$ is 1.99.

4. Runs Test: The main aim of this test is to count total number of zero and one run in the entire sequence. A run of length

n means n consecutive identical bits bounded before and after with a bit of the opposite value. The purpose of this test is to determine whether the number of runs of ones and zeroes of various lengths is as expected for a random sequence. The expected number of gaps (or blocks) of length i in a random sequence of length n is $e_i = (n-i+3)/2^{i+2}$. Let k be equal to the largest integer i for which $e_i \geq 5$. Let $B_i$, $G_i$ be the number of blocks and gaps respectively, of length i in s, for each i, $1 \leq i \leq k$. the statistic used is

$$X_4 = \sum_{i=1}^{k} \frac{(B_i - e_i)^2}{e_i} + \sum_{i=1}^{k} \frac{(G_i - e_i)^2}{e_i}$$

5. Longest run of 1's: This test count the longest run of 1's within N bit blocks, and show that whether it is consistent with the length of longest run of 1's that would be expected in a random sequence.

For testing the randomness we take 30 prime order elliptic curves over a prime field GF(14615016373309029182036848327162830196559325429 83) with a random point on each of them. The first 63 generated numbers from each of them is tested as specified in FIPS 140-2[11]. FIPS is referred to as the United States Government Federal Information Processing Standard, A publication of the National Institute of Standards and Technology. It specifies US Government's criteria on cryptographic modules for sensitive but unclassified use. For reference result from the curve defined by $y^2=x^3+90511688764072634235450357552629407119905584185 1*x+11156673411828934385529845269787421626 34550465324$ over GF(14615016373309029182036848327162830196559325429 83) with initial point as (14132957010062720805809639683992498070558314616 25, 58428533423546697074180209045715805669760625111 9 ) is summarized in the table given in the next page.

Table 1: FIPS 140-2 statistical test result of the generated sequence of 20096 bits data.

| Statistical Test | Required Interval | Output Value(X) |
|---|---|---|
| Monobit Test | 9725 < X < 10275 | 10164 |
| Poker Test | 2.16 < X < 46.17 | 6.953 |
| Runs Test for '1' | Run = 1, 2243 ≤ X ≤ 2657<br>Run = 2, 1135 ≤ X ≤ 1365<br>Run = 3, 542 ≤ X ≤ 708<br>Run = 4, 251 ≤ X ≤ 373<br>Run = 5, 111 ≤ X ≤ 201<br>Run ≥ 6, 111 ≤ X ≤ 201 | 2509<br>1250<br>645<br>322<br>167<br>153 |
| Runs Test for '0' | Run = 1, 2243 ≤ X ≤ 2657<br>Run = 2, 1135 ≤ X ≤ 1365<br>Run = 3, 542 ≤ X ≤ 708<br>Run = 4, 251 ≤ X ≤ 373<br>Run = 5, 111 ≤ X ≤ 201<br>Run ≥ 6, 111 ≤ X ≤ 201 | 2563<br>1256<br>646<br>286<br>141<br>153 |
| Long Run Test | Run ≥ 26, X=0 | 0 |
| Result | | Passed |

**Section-V: Conclusion**

An algorithm for computationally fast and cryptographically secure Pseudo Random Number Generator is proposed and described in this paper. It is based on simple point operations on elliptic curves with high prime orders. The implementation of FIPS 140-2 specified randomness tests outputs a good result. Thus for a secure two party communication this PRNG can be used. To get a strong security level the initial parameters E, P, p should be exchanged secretly.

References:

1. Neal Koblitz, “Elliptic Curve Cryptosystems”, Mathematics of Computation, Vol-48, 1987, pp-203-209
2. V.Miller, “Uses of Elliptic Curve in Cryptography”, Advances in cryptology-CRYPTO’85, vol-218, Springer Heidelberg, 1986, pp. 417-426.
3. Omar Reyad, Zbigniew Adam Kotulski, ”Pseudo Random Sequence Generation from Elliptic Curves over a Finite Field of Characteristic-2” , https://researchgate.net/publication
4. Sean Hallgren, “Linear Congruential generators Over Elliptic Curves”, 1994, CMU-CS-94-143.
5. https://en.m.wikipedia.org, “Dual_EC_DRBG”.
6. P.P.Deepthi & P.S.Sathidevi, “New stream ciphers based on elliptic curve point multiplication”, Computer Communications, vol. 32, 2009, pp. 25-33, doi 10.1016/j.comcom.2008.09.002.
7. Darrel Hankerson, Alfred Menezes, Scott Vanstone “Guide to Elliptic Curve Cryptography”, Springer Professional Computing, 2004.
8. L.P.Lee & K.W.Wong, “A Random Number Generator Based on elliptic Curve Operations”, Computers and Mathematics with Applications, Vol-47, Issues 2-3, pp. 217-226.
9. Random.org (Randomness and integrity service LTD).
10. Alfred J Menezes, Paul C. van Oorschot, Scott A Vanstone, “A Handbook of applied Cryptography”. CRC Press
11. Cryptographic Module Validation (CMV) programs at NIST, http://csrc.nist.gov/cryptval/.