Analysis of Pseudorandom Number Generator over Elliptic Curve

Yin Yin Myat, Ohmmar Min Mon
Computer University, Taung- Ngu
myatyinyin@gmail.com, ohnmarminmon2009@gmail.com

Abstract

Pseudorandom Number Generators (PRNGs) are computational algorithms that produce long sequence of random results, which are determined by a shorter initial seed or key. Elliptic curve cryptosystem is a public key cryptosystem. It is replacing with RSA in most of the cryptosystems because it has the same level of security with smaller key sizes. This system presents an extract of results taken from analyzing Nonlinear Filter Generator (NLFG) and Linear Congruential Generator (LCG) based on Elliptic Curves defined over prime fields. Randomness is a crucial resource for cryptography so outputs of these two algorithms are evaluated using statistical tests for randomness. This system will be implemented by using Visual C$^{\#}$ programming language.

Keywords: Pseudorandom number generators, Elliptic curve, Statistical tests.

1. Introduction

Cryptography is an important aspect of communications security. Almost all cryptographic protocols depend on random values to create keys, cookies, tokens, initialization vectors, and so on. The heart of every cryptosystem, there is an important part which is used to create unpredictable data such as session keys. Using pseudorandom number generator is quite common for this purpose.

Pseudorandom Number Generator (PRNG) is a deterministic algorithm, which takes an input of random bits, and expands those to a larger number of pseudorandom bits. Pseudorandom number sequence has very important applications in cryptography such as key generation [6]. There are some conventional methods for generating pseudorandom number sequences. Among them, Nonlinear Filter Generator (NLFG) and Linear Congruential Generator (LCG) are used in this system.

Elliptic curve cryptography is a very efficient technology for public key cryptosystem. Elliptic curve cryptosystem uses smaller key sizes for the security [4]. This system is an approach, which generates nonlinear and linear pseudorandom sequence from elliptic curve.

Verification of the statistical quality of pseudo-random data by detecting deviations from true randomness is based on statistical testing. These tests may be useful in determining whether or not a generator is suitable for a cryptographic application [5].

2. Related Work

Communication complexity with bias has the following remarkable application, first developed by Babai, Nisan and Szegedy [1].The calculation program for the points on elliptic curve using elementary operation of inversion. Random numbers are used throughout theoretical computer science as a way to make algorithms simpler and more efficient, but in the real world, it is extremely difficult to generate truly random numbers [2]. A generator in which no bit in the output can be computed assuming that the discrete log problem over an elliptic curve is hard [3]. Generators suitable for use in cryptographic applications may need to meet stronger requirements than for other applications. In particular, their outputs must be unpredictable in the absence of knowledge of the inputs. In practice, people use pseudorandom number generators that generate truly randomness sequence [5]. A random sequence has the uniform distribution, the unpredictability and no dependences between elements of a sequence [6].

3. Background Theory

In this section, Elliptic Curve Cryptography, Pseudorandom number generator and Statistical test are presented as background theory.

3.1. Elliptic Curve Cryptography (ECC)

ECC is a cryptosystem based on the finding points on an elliptic curve over a field. ECC is based on a discrete logarithm problem in elliptic curve groups. Finite field GF(p), where p>3 and p is a prime number. General formula for elliptic curve E over GF(p) can be written in the following standard form [2]:

$$E: y^2 = x^3 + ax + b \in GF(p) \qquad (1)$$

where, a and b are the curve parameters over GF(p) such that $4a^3 + 27b^2 (\mathrm{mod}\ p) \neq 0$.

3.1.1. Point Addition

One of the operations on an elliptic curve over prime field is adding two points. If the two points P and Q $\in$ E, where P or Q $\neq$ O and Q $\neq$ -P, then $R= P + Q = (x_R, y_R)$.

$$\begin{aligned} x_R &= \left(\lambda^2 - x_P - x_Q\right)(\mathrm{mod}\ p), \\ y_R &= \lambda\left(x_P - x_R\right) - y_P\ (\mathrm{mod}\ p), \end{aligned} \qquad (2)$$

where, $\lambda = \dfrac{y_P - y_Q}{x_P - x_Q}(\mathrm{mod}\ p)$ (3)

λ is the slope of the line through P and Q.

3.1.2. Point Doubling

Another property of elliptic curve over prime field is doubling a point. If $P = (x_P, y_P) \in E$ and $y \neq 0$, then $R= 2P= (x_R, y_R)$.

$$\begin{aligned} x_R &= \left(\lambda^2 - 2x_P\right)(\mathrm{mod}\ p), \\ y_R &= \lambda\left(x_P - x_R\right) - y_P\ (\mathrm{mod}\ p), \end{aligned} \qquad (4)$$

where, $\lambda = \dfrac{3x_P^2 + a}{2y_P}(\mathrm{mod}\ p)$ (5)

a is one of the parameters chosen with the elliptic curve and λ is the tangent on the point P.

3.2. Pseudorandom Number Generator

A pseudorandom number generator (PRNG) is an algorithm for generating a sequence of numbers that approximates the properties of random number [6].

3.2.1. Non-Linear Filter Generator over Elliptic Curve

A linear feedback shift register should not be used in cryptographic work because the outputs are completely linear, leading to fairly easy cryptanalysis. A way to destroy the linearity in LFSR is to use a nonlinear filter function f with single output z and j inputs. It is called a nonlinear filter generator (NLFG). Nonlinear pseudorandom sequence based on LFSR and *R*-block.

Firstly, construct a linear recursive sequence over E that is called LFSR-EC sequence $\underline{P}$. Let E be an elliptic curve over GF(p) of order r, $f(x)= x^n - c_{n-1}x^{n-1} - \ldots - c_1x - c_0 \in Z_r[x]$ and $\underline{P} = \{P_k\}$ where $P_k = (x_k, y_k) \in E$. For fixed $c_{-1} \in Z_r$ and given (n+1) – tuple $(Q, P_0,\ldots, P_{n-1}) \in E^{n+1}$, if the sequence satisfies the following linear recursive relation:

$$P_{n+k} = \sum_{i=0}^{n-1} c_i P_{i+k} + c_{-1}Q,\ k = 0,1,\ldots, \qquad (6)$$

$\underline{P}$ is called an nth order linear recursive sequence over E or simply EC sequence for short; f(x) is called a characteristic polynomial of $\underline{P}$ over Z_r and $(Q, P_0, P_1,\ldots, P_{n-1})$ is called an initial state of $\underline{P}$.

Then, choose a nonlinear function f which is called *R*-block that is built of elements over $GF(2^m)$. The elements of $\underline{P}$ are input to the nonlinear function f and it generates a nonlinear pseudorandom sequence $\underline{z} = \{z \in GF(2^m)\}$ as an output key stream of generator. The formula:

$$R_H(A, B) = H(e_A + B) \in GF(2^m) \qquad (7)$$

where A, B are input elements $GF(2^m)$, e_A is an address of cell containing element A in H table. The output of R-block is the essence of reading the contents of cells in H table after summation of an input element B and an address of cell containing input element A.

The binary pseudorandom sequence $\underline{b}$ is obtained by using the function h. The function h maps the elements of $GF(2^m)$ to the elements of GF(2). The function h is defined as following [6].

$$h(z) = \begin{cases} 0, & 0 \le z \le \dfrac{2^m - 1}{2} \\ 1, & \text{otherwise} \end{cases} \qquad (8)$$

3.2.2. Linear Congruential Generator over Elliptic Curve

A Linear Congruential Generator represents one of the oldest and best-known pseudorandom number generator algorithms. In linear congruential generator over an elliptic curve, the sequence of points $X_0, X_1, X_2,\ldots$ is computed, where the points are determined by the recurrence relation for some elliptic curve E, constant a, point B, and point X_0. The generator is defined by the recurrence relation:

$$X_{i+1}= aX_i + B \qquad (9)$$

In this case, the equation produces points, so some function of the points must be output. If the sequence is generated by a linear congruence over an elliptic curve and the reduction is based on the difficulty of the discrete log over an elliptic curve, it does not seem obvious that it is possible to contrast a CSPRB generator.

The pseudorandom sequence which should be examined is the same expect output y-coordinate(X_i) instead of log(X_i).

$$y_i = y_i\text{-coordinate}(X_i) \qquad (10)$$

In this generator the bit release function is r(X) is:[3].

$$r(X)= \text{lsb}\ (\log_G(X)) \qquad (11)$$

3.3. Statistical Tests

Various statistical tests can be applied to a binary sequence that produces PRNGs, to evaluate the sequence is random or not. Each test is run over a large number of sets of bits from the file to be tested. Each test is based on a calculated test statistic value, which is a function of the tested data. The test statistic is used to calculate a P-value. P-value is probabilistic value which lies in the unit interval, i.e, in the range between 0 and 1.If the P-value is greater than or equal to the defined significance (α), the conclusion that the sequence is random. There are 16-test which are recognized from NIST Suite [5].

4. Overview of the Proposed System

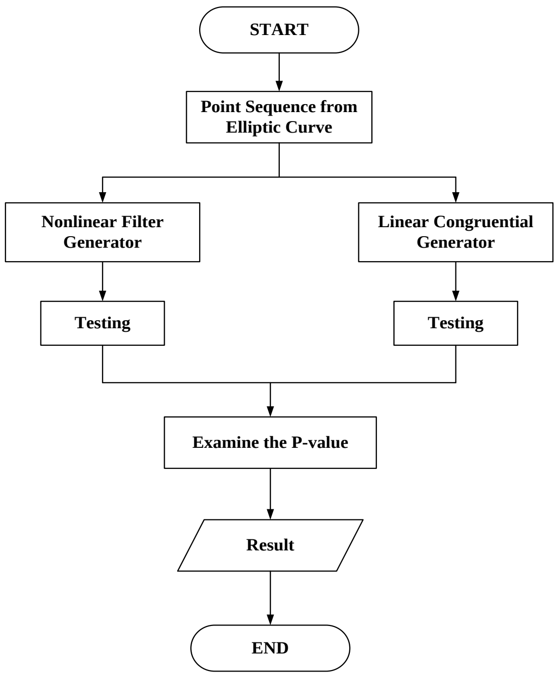
START
Point Sequence from Elliptic Curve
Nonlinear Filter Generator
Linear Congruential Generator
Testing
Testing
Examine the P-value
Result
END

Figure 1. System design

This system is implemented for "Analysis of Pseudorandom Number Generator over Elliptic Curve" and shown in Figure 1. It consists of three stages.

First stage is to generate point sequence from elliptic curve. The methods to generate point sequence on an elliptic curve defined over prime field are addition of distinct points: illustrated in eq: 2 and 3 and doubling a point itself: illustrated in eq: 4 and 5 (See sections 3.1.1 and 3.1.2).This point sequence is used as input into the two generators.

The second is to produce pseudorandom sequence from Nonlinear Filter Generator (NLFG) and Linear Congruential Generator (LCG).

The function of NLFG can be divided into three parts.

(1) Accept a point sequence from elliptic curve as input: illustrated in eq: 6.This recursive relation gives the following LFSR-EC sequence:
P=(1P,2P,13P,2P,3P,....).

(2) Extract a nonlinear pseudorandom sequence from the point sequence by using a non-linear function called R-block: illustrated in eq: 7.This sequence is:
z = (6,6,9,11,12,12,....).

(3) Extract a binary pseudorandom sequence from the nonlinear pseudorandom sequence by using binary mapping function: illustrated in eq: 8.This sequence is:
b=(0,0,1,1,1,1,1,1,.....)

The function of LCG can be divided into three parts:

(1) Accept a point sequence from elliptic curve as input: illustrated in eq: 9.This linear recursive equation gives the following mixed EC point sequence:
X_i=(11P,15P,12P,20P,...)

(2) Extract linear pseudorandom sequence from the mixed point sequence by using output function: illustrated in eq: 10. This sequence is:
y= (5,3,4,1,....).

(3) Extract binary pseudorandom sequence from the linear pseudorandom sequence by applying the bit release function : illustrated in eq: 11. This sequence is: r= (1,0,0,1,1,0,....).

The final stage is to test output binary sequences from two generators with statistical tests and examine the test result to result out the quality of the generators.

5. Statistical Test Evaluation

The following tests are used to test the output binary sequence from two generators.

5.1. Frequency (Mono-bit) Test

Determine the proportion of ones and zeros for the entire sequence .The zeros and ones of the input sequence (**ε**) are converted to values of –1 and +1and are added together to produce S_n [5].

- Compute the test statistic $S_{obs} = \frac{|S_n|}{\sqrt{n}}$. (12)
- Compute P-value = erfc $\left(\frac{S_{obs}}{\sqrt{2}}\right)$. (13)

Example

(input) ε = 1011010101 (bit sequence from PRNG)
(input) n = 10 (the length of big string)
(processing) $S_n = 2$
(processing) $S_{obs} = 0.6324$ (test statistic value)
(output) P-value =0.5271.

(conclusion) Since the P-value $\geq$ 0.01, accept the sequence is random.

5.2. Frequency Test within a Block

Determine whether the frequency of ones in an M-bit block is approximately M/2. Partition the input sequence into N = $\left\lfloor \frac{n}{M} \right\rfloor$ non-overlapping blocks. Discard any unused bits. Determine the proportion π_i of ones in each M-bit block using the equation [5]:

$$\pi_i = \frac{\sum_{j=1}^{M} \varepsilon_{(i-1)M+j}}{M}, \ for \quad 1 \leq i \leq N . \qquad (14)$$

- Compute the $\chi^2(\text{obs}) = 4M \sum_{i=1}^{N} \left(\pi_i - \frac{1}{2}\right)^2$ (15)
- Compute P-value $= \text{igmac}\left(N/2, \chi^2(\text{obs})/2\right)$. (16)

Example

(input) ε = 0110011010(bit sequence from PRNG)
(input) n = 10 (the length of big string)
(input) M = 3 (the length of each block)
(processing) N = 3 (the number of block)
(processing) $\chi^2(\text{obs}) = 1$ (test statistic value)
(output) P-value = 0.8012

(conclusion) Since the *P*-value $\geq$ 0.01, accept the sequence is random.

5.3. Runs Test

Determine the total number of zero and ones runs in the entire sequence. Compute the pre-test proportion π of ones in the input sequence [5].

- Compute the test statistic $V_n(\text{obs}) = \sum_{k=1}^{n-1} r(k) + 1$, $r(k)=0$ i*f* $\varepsilon_k = \varepsilon_{k+1}$, an*d* r(k)=1 otherwis*e*. (17)
- Compute P-value $= \text{erfc}\left(\frac{\left|V_n(\text{obs}) - 2n\pi(1-\pi)\right|}{2\sqrt{2n}\pi(1-\pi)}\right)$. (18)

Example

(input) ε =1001101011(bit sequence from PRNG)
(input) n = 10(the length of big string)
(input) π = 6/10=3/5 (proportion of ones)
(processing) $V_{10}(\text{obs}) = 7$ (test statistic value)
(output) P-value= 0.1472

(conclusion) Since the P-value $\geq$ 0.01, accept the sequence is random.

5.4. Cumulative Sums (Cusum) Test

Determine the cumulative sum of partial sequence occurring in the tested sequence. The zeros and ones of the input sequence (ε) are converted to values X_i of –1 and +1. Compute partial sums S_i of successively larger subsequences, each starting with X_1 (if mode = 0) or X_n (if mode = 1).

- Compute the test statistic z $=\max_{1\leq k\leq n} |S_k|$.
- P-value=

$$1 - \sum_{k=\left(\frac{-n}{z}+1\right)/4}^{\left(\frac{n}{z}-1\right)/4} \left[\Phi\left(\frac{(4k+1)z}{\sqrt{n}}\right) - \Phi\left(\frac{(4k-1)z}{\sqrt{n}}\right)\right] + \sum_{k=\left(\frac{-n}{z}-3\right)/4}^{\left(\frac{n}{z}-1\right)/4} \left[\Phi\left(\frac{(4k+3)z}{\sqrt{n}}\right) - \Phi\left(\frac{(4k+1)z}{\sqrt{n}}\right)\right] \qquad (19)$$

Example

(input) ε =1100100100(bit sequence from PRNG)
(input) n =10(the length of big string)
(input) mode=0 (forward) || mode=1(reverse)
(processing) z = 1.6 (forward) || z = 1.9 (reverse)
(output) P-value=0.219(forward)|| P-value=0.116(reverse)

(conclusion) Since the P-value $\geq$ 0.01, accept the sequence is random.

6. Experimental Result

In this system, binary pseudorandom sequences are generated from two generators by using point sequences over elliptic curve based on prime field. These binary sequences are tested with defined four tests: Frequency Test, Frequency Test within a Block, Runs Test, Cumulative Sums Test. Finally, the statistical results that are generated from each of these runs are called P-values. A P-value is the probability that a perfect random number generator would have produced a sequence less random than the sequence that was tested. For each sequence, a set of P-values is produced. Based on these P-values, a conclusion regarding the quality of the sequences can be made. For a fixed significance level, α (0.01,0.001), a certain percentage of P-values are expected to indicate failure. A sequence passes a statistical test whenever the P-value≥ α and fail otherwise. For example, P-value≥0.01, and the conclusion would be that the sequence was random otherwise non-random. A statistical test result is described in table 1.

Table 1. A Statistical test result

Statistical Test	P-value for NLFG	P-value for LCG
Frequency Test	0.750	0.620
Frequency Test within a Block	0.860	0.520
Runs Test	0.476	0.319
Cumulative Sums Test	0.850	0.428

7. Conclusion

Computer-based systems for pseudorandom number generation are widely used, but they may meet some statistical tests for randomness. This system proposes the concept of using two algorithms of pseudorandom number generator over elliptic curve for generating secure pseudorandom numbers. numbers. This sequence has uniform distribution, unpredictability and no dependences between elements of a sequence. Statistical analysis of the output is needed to have confidence in the algorithm. So, pseudorandom sequences which pass one or more several statistical tests for randomness in this system and may be useful in many circumstances. Thus, NLFG and LCG over elliptic curve are cryptographically strong generators.

7. References

[1] Babai, Nisan, and Szegedy, "Concrete Models of Compution", Aprial 15, 2003.

[2] Certicom Research, "Standards for Efficient Cryptography 1: Elliptic Curve Cryptography", Version 1.0, September 20, 2000.

[3] H.Sean, "Linear Congruential Generators Over elliptic Curves", School of Computer Science, Carnegie Mellon University, May, 1994.

[4] M.A.Salie, C.Omlin, "Elliptic Curve Cryptography", University of the Western Cape, Private Bag X17, Bellville, 7535.

[5] R. Andrew, S. Juan, N, Jame, "A Statistical Test Suite For Random and Pseudorandom Number Generators for Cryptographic Applications", NIST Special Publication 800-22, May 15, 2001.

[6] Tun Myat Aung, "Nonlinear Filter Generator over Elliptic Curve", Center of Advances Science and Technology, University of Computer Studies, Yangon, Myanmar.

[7] http://en.wikipedia.org/wiki/ Pseudorandom number generator.