PSEUDORANDOM NUMBERS AND ELLIPTIC CURVES OVER FINITE FIELDS

Thesis

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

KARUNA KAMATH K

DEPARTMENT OF MATHEMATICAL AND COMPUTATIONAL SCIENCES
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA,
SURATHKAL, MANGALORE - 575 025

DECEMBER 2012

DECLARATION

by the Ph.D. Research Scholar

I hereby declare that the Research Thesis entitled **PSEUDORANDOM NUMBERS AND ELLIPTIC CURVES OVER FINITE FIELDS** which is being submitted to the **National Institute of Technology Karnataka, Surathkal** in partial fulfillment of the requirements for the award of the Degree of **Doctor of Philosophy** in **Mathematics** is a bonafide report of the research work carried out by me. The material contained in this Thesis has not been submitted to any University or Institution for the award of any degree.

MA05P02 KARUNA KAMATH K
(Register Number, Name Signature of the Research Scholar)
Department of Mathematical and Computational Sciences

Place: NITK, Surathkal
Date:

CERTIFICATE

This is is to *certify* that the Research Thesis entitled, **PSEUDORANDOM NUMBERS AND ELLIPTIC CURVES OVER FINITE FIELDS** submitted by **Karuna Kamath K (Register Number: MA05P02)** as the record of the research work carried out by her, is *accepted as the Research Thesis submission* in partial fulfillment of the requirements for the award of the degree of **Doctor of Philosophy**.

Dr.Murulidhar N N
Chairman - DRPC
Professor and Head
Department of Mathematical
and Computational Sciences

Dr. B.R. Shankar
Research Guide
Department of Mathematical
and Computational Sciences

Acknowledgement

I am genuinely obliged to many persons who have greatly inspired and supported me during my research work at National Institute of Technology Karnataka.

I am indebted to my research guide Dr. B.R Shankar, for showing his extraordinary patience and perseverance in order to motivate me, and for providing his immense knowledge and wisdom to my easy and ready access. In all humbleness and reality of my position as a student to him I acknowledge his guidance as his commitment made to me. His guidance helped me during my research work and writing the thesis. His wisdom, knowledge and commitment inspired and motivated me.

I would like to thank Dr.Murulidhar N N,Professor and Head,MACS Department for extending all the facilities during my research work.

Dr. S. M. Hegde, Professor, Department of MACS, National Institute of Technology Karnataka and Dr. Lakshman Nandagiri, Professor and Head, Department of Applied Mechanics and Hydraulics, National Institute of Technology Karnataka are reverently remembered for their significant and momentary comments as RPAC members during my research work.

The co-operation and support showed by all of the MACS, NITK faculty shall always be fondly remembered by me and for making me feel at-home during my stay in there.

I am greatly thankful to The President, Nitte Education Trust, Mangalore and The Principal, NMAM Institute of Technology, Nitte, for facilitating the research study besides resourcing with the necessary and essential support.

I would like to thank Dr. K.M.Hegde, Director, Department of Computer Applications, NMAM Institute of Technology, Nitte, for his constant support and guidance.

My sincere thanks to Mr. Manjunath Pai, Associate Professor,Department of Electronics and Communication Engineering, NMAMIT, Nitte for his support and idea.

My very special thanks are extended to Mr. Raveeshwar for availing me his wonderful technical support and my fellow-research-scholars, Dr. Vadiraja Bhatta and Mrs. Sushma Palimar shall always be fondly recalled for their friendly and supporting companionship, and my sincere thanks to all those who came in as help at one time or the other in this research study of mine.

Abstract

Random and pseudorandom numbers are extensively used in simulation and statistical modeling systems, in controlling computational processes, and in computer games. Many mathematical optimization methods and game theory apply random and pseudorandom elements. Pseudorandom binary sequences are also widely used in information security algorithms. Pseudorandom number generation is the art and science of deterministically generating a sequence of numbers that is hard to differentiate from a true random sequence. This thesis studies some methods of random number generation. In the first section we describe the most commonly used pseudorandom number generators to provide the necessary background.

Elliptic curves are rich mathematical structures which have shown themselves to be incredibly useful in a wide range of applications. Most of the products and standards that use public-key cryptography for encryption and digital signatures use RSA. The key length for secure RSA use has increased over recent years, and this has put a heavier processing load on applications using RSA. Elliptic curve cryptography can provide the same level and type of security as RSA but with much shorter keys. In the third, fourth and fifth chapters new pseudorandom number generators are developed using elliptic curves over finite fields and the existing generators. The emphasis will be on the length of the sequences produced by such generators and the statistical properties to ensure their usage in cryptographic application.

An interesting property of numbers is that almost all numbers become palindromes quickly after repeated reversal and addition of its digits. But there are some numbers which are an exception to this. These numbers are called Lychrel numbers. In the next chapter two algorithms are presented to generate secret keys with palindrome and Lychrel numbers. In the last chapter the deployment of secret keys for security purpose in stream ciphers and stegnography are studied.

Contents

Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin.

John von Neumann

Chapter 1

Introduction

Random and pseudorandom numbers are widely used in simulation and statistical modeling systems, in controlling computational processes and in computer games. Many mathematical optimization methods including game theory, apply random and pseudorandom elements. Pseudorandom binary sequences are also widely used in information security algorithms. Security systems today are built on increasingly strong cryptographic algorithms. However, the security of these systems is dependent on generating secret quantities for passwords and cryptographic keys. Generation of unguessable "random" secret quantities for security use is a crucial but intricate task.

1.1 Random Numbers

One of the important elements of modern informational technology are random and pseudorandom binary sequences and their generation. The term randomness is often used in statistics to signify well-defined statistical properties like lack of bias or correlation. Informally, a sequence of numbers is said to be truly random if there is no correlation among the subsequent elements and it is not possible to predict the next number in the sequence with absolute certainty. According to Knuth(Knuth 1997), a sequence of random numbers is a set of independent numbers with a specified distribution and a specified probability of falling in any given range of values. For Schneier,(Schneier 1996) it is a sequence that has the same statistical properties as random bits, is unpredictable and cannot be reliably reproduced. A concept that

is present in both of these definitions and that must be emphasized is the fact that numbers in a random sequence must not be correlated. Knowledge of one or some of the numbers of a random sequence must not help predicting the other ones. True random numbers are rarely used in computation, because it is difficult to generate the same sequence again. The lack of reproducibility would make validation of programs that use these numbers extremely difficult.The degree of randomness required is determined by the type of application.
A pseudorandom sequence is obtained by using an algorithm or a set of equations. This is not truly random as each element of the sequence is completely determined. They only appear random to an observer who has no knowledge about the algorithm used. They are computed from a mathematical formula or simply taken from a precalculated list. They are referred to as "pseudorandom" because given a particular function and a "seed" value, the same sequence of numbers can be generated by the function.

1.2 Statistical Testing

A sequence of numbers generated cannot be trusted to judge by ourselves whether it is random or not. Some unbiased mechanical tests must be applied. Every sequence that is to be used extensively, should be tested carefully. Various statistical tests can be applied to a sequence to attempt to compare and evaluate the sequence to a truly random sequence. Randomness is a probabilistic property; that is, the properties of a random sequence can be characterized and described in terms of probability. The likely outcome of statistical tests, when applied to a truly random sequence, is known a priori and can be described in probabilistic terms. There are number of possible statistical tests, each assessing the presence or absence of a "pattern" which, if detected, would indicate that the sequence is nonrandom. Because there are many tests for judging whether a sequence is random or not, no specific finite set of tests is deemed "complete". In addition, the results of statistical testing must be interpreted with some care and caution to avoid incorrect conclusions about a specific generator.

1.3 IP Core of Statistical Test Suite of FIPS 140-2

Federal Information Processing Standard (FIPS),(Hasegawa and Umeno 2002) publication for cryptographic modules specifies four statistical tests for randomness. Instead of making the user select appropriate significance levels for these tests, explicit bounds are provided that the computed value of a statistic must satisfy. A single bit stream of 20000 bits, output from a generator is subjected to each of the four tests.

1.3.1 The Mono Bit Test

The number of one's is counted in a bit stream of 20,000 bits.
Statistic X: the number of one's (zero's) in the bit stream.
Acceptance Region:9,725 < X < 10,275.

1.3.2 The Poker Test

A stream of 20,000 bits is divided into 5,000 non overlapping consecutive 4-bit segments. The total number of patterns of 4-bit segments is $2^4 = 16$. The number of occurrences of each of the 16 possible patterns is counted. Let f_i, (i = 0 to 15) be the number of occurrences of each pattern.
Statistic X:

$$X = (16/5000)*(\sum_{i=0}^{15} f_i^2) \text{ -5000}$$

Acceptance Region: 2.16 < X < 46.17.

Integer Version of Poker's Test

This version of the test is implemented by Vancak,(Vancak 2009).

1. count an appearance f_i of each group of four bits in the sequence of 20000 bits.
2. Compute $\sum_{i=0}^{15} f_i^2$
3. The sequence would not pass the test if $1563175 < \sum_{i=0}^{15} f_i^2 < 1576929$, or atleast one combination of bits exceeds 428.

1.3.3 The Long Runs Test

The length of longest run is determined from 20,000 bit pattern.
Statistic X: The length of the longest run in bit stream of 20,000 bits (both of one and zero).

Acceptance Region: X $<$ 26

1.3.4 The Runs Test

A run is defined as a sequence of consecutive bits of either all ones or all zeros. In this test, the number of runs in a bit stream of 20,000 bits is counted. The run length of 1, 2, 3, 4, 5, and 6 or greater than 6 of either one or zero is counted.

Statistic X : The number of runs of each length appears in the bit stream.

Acceptance Region:

Length of Runs	Acceptance Region
1	2,315 - 2,685
2	1,114 - 1,386
3	527 - 723
4	240 - 384
5	103 -209
6+	103- 209

Table 1.1 Acceptance regions for runs test

1.4 NIST Statistical Test Suite

NIST (NIST 2001) test suite is used as a bench mark by National Institute of Standards and Technology, in the evaluation of possible candidate generators for the Advanced Encryption Standard(AES). The NIST Test Suite is a statistical package consisting of 16 tests that were developed to test the randomness of binary sequences

produced by random or pseudorandom number generators.
The tests are

- The Frequency (Mono bit) Test
- Frequency Test within a Block
- The Runs Test
- Test for the Longest-Run-of-Ones in a Block
- The Binary Matrix Rank Test
- The Discrete Fourier Transform (Spectral) Test
- The Non-overlapping Template Matching Test
- The Overlapping Template Matching Test
- Maurer's "Universal Statistical" Test
- The Lempel-Ziv Compression Test
- The Linear Complexity Test
- The Serial Test
- The Approximate Entropy Test
- The Cumulative Sums Test
- The Random Excursions Test
- The Random Excursions Variant Test.

Table 1.2 describes the general characteristics of each of the statistical tests.

Statistical Test	Defect Detected
Frequency	Too many zeroes or ones
Cumulative Sums	Too many zeroes or ones at the beginning of the sequence
Longest Runs Of Ones	Deviation of the distribution of long runs of ones
Runs	Large (small) total number of runs indicates that the oscillation in the bit stream is too fast (too slow)
Rank	Deviation of the rank distribution from a corresponding random sequence, due to periodicity
Spectral	Periodic features in the bit stream
Non-overlapping Template Matchings	Too many occurrences of non-periodic templates
Overlapping Template Matchings	Too many occurrences of m-bit runs of ones
Universal Statistical	Compressibility (regularity)
Random Excursions	Deviation from the distribution of the number of visits of a random walk to a certain state
Random Excursion Variant	Deviation from the distribution of the total number of visits (across many random walks) to a certain state
Approximate Entropy(Ap En)	Non-uniform distribution of m-length words. Small values of ApEn(m) imply strong regularity
Serial	Non-uniform distribution of m-length words. Similar to Approximate Entropy
Lempel-Ziv Complexity	More compressed than a truly random sequence
Linear Complexity	Deviation from the distribution of the linear complexity for finite length (sub)strings

Table1.2 The general characteristics of each of the statistical tests in the NIST Suite.

A statistical test is formulated to test a specific *null hypothesis*, H_0 and an *alternative hypothesis* H_a. For each test applied , a decision or conclusion is derived that accepts or rejects the null hypothesis.During a test, a *test statistic value* is computed on the data . The *test statistic* is used to calculate a *P-value* that summarizes the strength of the evidence against the null hypothesis. If a *P-value* for a test is determined to be equal to 1, then the sequence appears to have perfect randomness. A *P-value* of zero indicates that the sequence appears to be completely non-random. A significance level α can be chosen for the tests. If P-value $> \alpha$, then the null hypothesis is accepted; i.e., the sequence appears to be random. If P-value $< \alpha$, then the null hypothesis is rejected; i.e., the sequence appears to be non-random. The parameter α denotes the probability of the Type I error. Typically, α is chosen in the range [0.001, 0.01].

1.5 Elliptic Curves

Elliptic curves are rich mathematical structures which have shown themselves to be remarkably useful in a range of applications including integer factorization and primality testing (Lenstra 1987, Menezes et al. 1997). One potential use of elliptic curves is in the definition of public-key cryptosystems that are close analogs of existing schemes (Hankerson et al. 2004). In this way, variants of existing schemes can be devised that rely for their security on a different underlying hard problem.
Most of the products and standards that use public-key cryptography for encryption and digital signatures use RSA. The key length for secure RSA use has increased over recent years, and this has put a heavier processing load on applications using RSA. This burden has ramifications, especially for electronic commerce sites that conduct large numbers of secure transactions.
Elliptic Curve Cryptography (ECC), was discovered in 1985 by Miller and Koblitz as an alternative mechanism for implementing public-key cryptography independently. The principal attraction of ECC, compared to RSA, is that it appears to offer equal security for a far smaller key size, thereby reducing processing overhead.

1.6 Fibonacci Sequence

The Fibonacci sequence,

$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987.....$
is defined by the recurrence relation

$F_n = F_{n-1} + F_{n-2}$ with seed values $F_0 = 0$ and$F_1 = 1$.

1.6.1 Lagged Fibonacci Generators

Lagged Fibonacci generator in one among the pseudorandom number generators discussed in Knuth's well known exposition on pseudorandom number generation (Knuth 1997). This class of random number generator is aimed at being an improvement on the 'standard' linear congruential generator. These are based on a generalization of the Fibonacci sequence. The Fibonacci sequence may be described by the recurrence relation:

$S_n = S_{n-1} + S_{n-2}$

Here, the new term is the sum of the last two terms in the sequence.

This formula is generalized to give a family of pseudorandom number generators of the form: $S_n = S_{n-j}$ o S_{n-k}(mod m), where, j > k > 0. Instead of two initial values, j initial values, S_0, S_1, S_2 ... , S_{j-1}, are needed in order to compute the next element in the sequence. In this expression j and k are called 'lags'. m is usually a power of 2, often 2^{32} or 2^{64}. The operator o denotes a general binary operation. These generators are very sensitive to initialization.

If the operation used is addition, then the generator is described as an Additive Lagged Fibonacci Generator(ALFG), if multiplication is used, it is a Multiplicative Lagged Fibonacci Generator(MLFG), and if the $\bigoplus$ operation is used, it is called a Two-tap generalised feedback shift register(GFSR.)

Mascagni and Cuccaro (Mascagni et al. 1995a), proved that Lagged Fibonacci generators have a maximum period of (2^k - 1)* 2^{M-1} if addition or subtraction is used, and (2^k-1)*k if $\bigoplus$ operations are used to combine the previous values. If, on the other hand, multiplication is used, the maximum period is (2^k - 1)* 2^{M-3}, or 1/4 of period

of the additive case.
For the generator to achieve this maximum period, the polynomial,
y = $x^k + x^j + 1$ must be primitive over the integers modulo 2. Good candidates for values of j and k can be found in several sources (Knuth 1997).
The popular pairs are: (7, 10), (5, 17), (24, 55), (65, 71),(128, 159),
(6, 31), (31, 63), (97, 127), (353, 521), (168, 521), (334, 607), (273, 607), (418,1279)
The initialization of LFGs is a very complex problem. The output of LFGs is very sensitive to initial conditions, and statistical defects may appear initially but also periodically in the output sequence unless extreme care is taken.

1.7 Lychrel Numbers

An interesting property of decimal numbers is that almost all numbers become palindromes quickly after repeated digit reversal and addition. For example,

- 56 becomes palindrome after one iteration: 56+65 = 121.
- 57 becomes palindrome after two iterations: 57+75 = 132, 132+231 = 363.
- 59 becomes a palindrome after 3 iterations: 59+95 = 154, 154+451 = 605, 605+506 = 1111.
- 89 take an unusually large 24 iterations to reach the palindrome 8813200023188.
- 10,911 reaches the palindrome 4668731596684224866951378664 after 55 steps.
- 1,186,060,307,891,929,990 takes 261 iterations to reach a 119 digit palindrome.

This is the currently known world record for the "Most Delayed Palindrome Number". Some numbers cannot form palindromes after repeated digit reversal and addition. Such numbers are called *Lychrel* numbers. A Lychrel number is a natural number which cannot form a palindrome through the iterative process of repeatedly reversing its decimal digits and adding the resulting numbers. This process is called the *196-algorithm.* No Lychrel numbers are known, though many numbers are suspected

Lychrels, the smallest being 196. Walker, (Walker 1990) has computed 196 to a number of one million digits after 2,415, 836 iterations without reaching a palindrome. It is conjectured that 196 and other numbers which have not yet yielded a palindrome are Lychrel numbers, but no number has yet been proven to be Lychrel. Numbers which have not been demonstrated to be non-Lychrel and informally called "candidate Lychrel" numbers. Jason Doucette, Ian Peters and Benjamin Despres have found other Lychrel candidates in 2005. The first few candidate Lychrel numbers are: 196, 295, 394, 493, 592, 689, 691, 788, 790, 879, 887, 978, 986, 1495, 1497, 1585, 1587, 1675, 1677, 1765, 1767, 1855, 1857, 1945, 1947 and 1997. Because 196 is the lowest candidate Lychrel number it has received the most attention.

Chapter 2

Elliptic Curves over Finite Fields

2.1 Elliptic Curves

A cubic equation of form $y^2 = x^3 + ax^2 + bx + c$ is called an Elliptic curve. Using suitable transformation of the coordinates this can be expressed as $y^2 = x^3 + ax + b$, with $4a^3 + 27b^2 \neq 0$, called the standard form. Here a and b are fixed constants, x and y vary over $\mathbb{R}$ or $\mathbb{Q}$ or a finite field $\mathbb{F}_p$, where p is a fixed prime. Let $E(K)$ denote the set of all points in K x K that lie on the elliptic curve over the finite field K. A special point, O, called the point at infinity is added to $E(K)$. If the group law is defined as below, $E(K)$ turns out to be an abelian group, which in general is a product of two cyclic groups (Koblitz 1994).

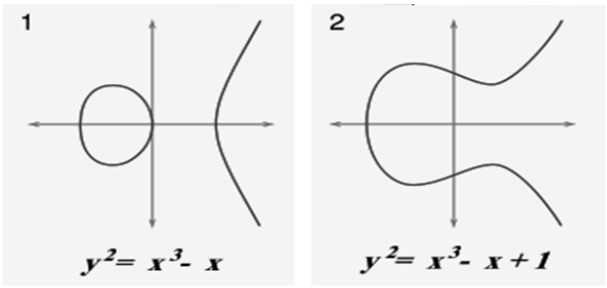


Fig 2.1 Graphs of curves $y^2 = x^3 - x$ and $y^2 = x^3 - x + 1$

2.2 Group Law

The group law is described (Blake et al. 2005, Silverman 2009, Crandall and Pomerance 2005) as follows:
Let E be an elliptic curve defined over the field K. There is a chord-and-tangent rule for adding two points in $E(K)$ to give a third point in $E(K)$. Together with this addition operation, the set of points $E(K)$ forms an abelian group with 'O' serving as its identity. This group is used in the construction of elliptic curve cryptographic systems.The details are given below.

2.2.1 Point Addition

Point addition is the addition of two points J and K on an elliptic curve to obtain another point L on the same elliptic curve. Let J = (x_1, y_1) and K = (x_2, y_2) be two distinct points on an elliptic curve E. Then the sum L, of J and K, is defined as follows. First draw a line through J and K; this line intersects the elliptic curve at a third point. Then L is the reflection of this point about the x-axis.

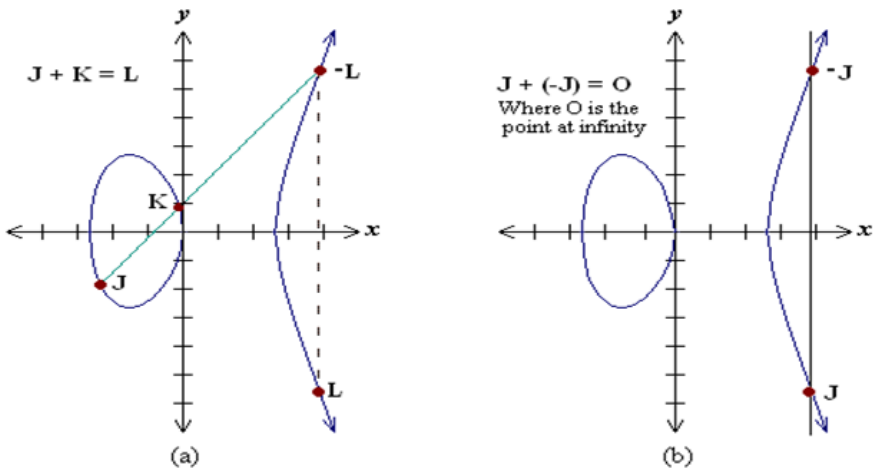


Fig 2.2 Point addition on Elliptic curve

2.2.2 Point Doubling

Point doubling is the addition of a point J on the elliptic curve to itself to obtain another point L on the same elliptic curve.

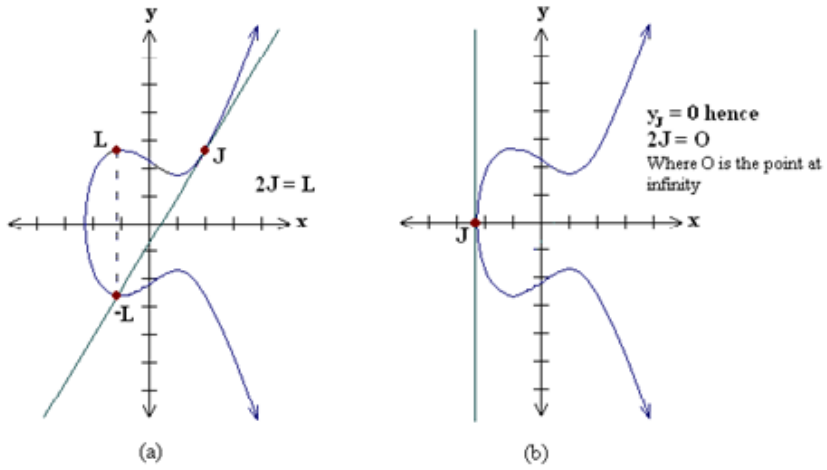


Fig 2.3 Point doubling

2.2.3 Group law for $y^2 = x^3 + ax + b$

The group law is described (Blake et al. 2005, Silverman 2009, Crandall and Pomerance 2005) as follows: Let E: $y^2 = x^3 + ax + b$ be an elliptic curve and K be a field.

1. Identity: $P + O = O + P = P$ for all $P \in E(K)$.
2. Negatives: If $P = (x, y) \in E(K)$, then $(x, y) + (x, -y) = O$. The point $(x, -y)$ is denoted by $-P$ and is called the negative of P.
3. Addition of points: Let $P = (x_1, y_1) \in E(K)$ *and* $Q = (x_2, y_2) \in E(K)$, then $P + Q = (x_3, y_3)$, where $x_3 = [(y_2 - y_1)/(x_2 - x_1)]^2 - x_1 - x_2$
   and $y_3 = [(y_2 - y_1)/(x_2 - x_1)](x_1 - x_3) - y_1$.

4. Point doubling: Let $P = (x_1, y_1) \in E(K)$, where $P \neq -P$.

   Then $2P = (x_3, y_3)$, where $x_3 = [(3x_1^2 + a)/2y_1]^2 - 2x_1$ and

   $y_3 = [(3x_1^2 + a)/2y_1](x_1 - x_3) - y_1$.

2.2.4 Group Order

Let E be an elliptic curve defined over F_q. The number of points in $E(F_q)$, denoted by $\#E(F_q)$ is called the *order* of E over F_q (Hankerson et al. 2004, Blake et al. 2005) Since E is quadratic in y, has at most two solutions for each $x \in F_q$, therefore $\#E(Fq) \in [1, 2q + 1]$.The following theorem gives the bounds for order of an elliptic curve.

Theorem 2.2.1. *(Hasse) Let E be an elliptic curve defined over F_q. Then $q + 1$-$2\sqrt{q} \leq \#E(F_q) \leq q + 1 + 2\sqrt{q}$. The interval $[q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q}]$ is called the Hasse interval.An alternate formulation of Hasse's theorem is the following:*
if E is defined over Fq, then $\#E(Fq) = q + 1 - t$ where $|t| \leq 2\sqrt{q}$;
t is called the trace of E over F_q. Since $2\sqrt{q}$ is small relative to q,we have # E(Fq) $\approx q$.

The next result determines the possible values for $\#E(F_q)$.

Theorem 2.2.2. *(Admissible orders of elliptic curves) Let $q = p^m$ where p is the characteristic of F_q. There exists an elliptic curve E defined over F_q with $\#E(F_q) = q + 1 - t$ if and only if one of the following conditions holds:*

1. *$t \neq 0(modp)$ and $t^2 \leq 4q$.*
2. *m is odd and either*
   - *$t = 0$; or*
   - *$t^2 = 2q$ and $p = 2$; or*
   - *$t^2 = 3q$ and $p = 3$*
3. *m is even and either*
   - *$t^2 = 4q$; or*

- $t^2 = q$ *and* $p \equiv 1$ *mod* 3*;*
- $t = 0$ *and* $p \equiv 1$ *mod* 4*;*

A consequence of this theorem is that for any prime p and integer t satisfying $|t| \leq 2\sqrt{p}$, there exists an elliptic curve E over Fp with $\#E(F_p) = p + 1 - t$.
This is illustrated in the following example.

Example 2.1 : (orders of elliptic curves over F_{37}) Let p = 37. Table 1 lists, for each integer n in the Hasse interval $[37 + 1 - 2\sqrt{37}, 37 + 1 + 2\sqrt{37}]$, the coefficients (a, b) of an elliptic curve E :$y^2 = x^3 + ax + b$ defined overF_{37} with $\#E(F_{37}) =$ n.

n	(a, b)	n	(a, b)	n	(a, b)	n	(a, b)	n	(a, b)
26	(5, 0)	31	(2, 8)	36	(1, 0)	41	(1, 16)	46	(1, 11)
27	(0, 9)	32	(3, 6)	37	(0, 5)	42	(1, 9)	47	(3, 15)
28	(0, 6)	33	(1, 13)	38	(1, 5)	43	(2, 9)	48	(0, 1)
29	(1, 12)	34	(1, 18)	39	(0, 3)	44	(1, 7)	49	(0, 2)
30	(2, 2)	35	(1, 8)	40	(1, 2)	45	(2, 14)	50	(2, 0)

Table 2.1 the admissible orders $\#E(F_{37}) =$ n. of elliptic curves
E: $y^2 = x^3 + ax + b$ defined over F_{37}.

2.2.5 Group Structure

The following theorem describes the group structure of E(F_q). We use Z_n to denote a cyclic group of order n.

Theorem 2.2.3. *Let E be an elliptic curve defined over* F_q*. Then* $E(F_q)$ *is isomorphic to* $Z_{n1} \bigoplus Z_{n2}$ *where* n_1 *and* n_2 *are uniquely determined positive integers such that* n_2 *divides both* n_1 *and q-1. Note that* $\#E(F_q) = n_1 n_2$*. If* $n_2 = 1$*, then E(Fq) is a cyclic group. If* $n_2 > 1$*, then E(Fq) is said to have rank 2. If* n_2 *is a small integer (e.g.,* $n =$ *2,3 or 4), we say that E(Fq) is almost cyclic. Since* n_2 *divides both* n_1 *and q-1, one expects that E(Fq) is cyclic or almost cyclic for most elliptic curves E over Fq .*

Example 2.2(group structure) The elliptic curve E:$y^2 = x^3 + 4x + 20$ defined over F_{29} has $\#E(F_{29}) = 37$. $E(F_{29})$The following shows that the multiples of the point,P = (1,5) generate all the points in F_{29}.

0P =O	8P = (8, 1 0)	16P = (0, 22)	24P = (16, 2)	32P = (6, 17)
1P = (1, 5)	9P = (14, 23)	17P = (27, 2)	25P = (19, 16)	33P = (15, 2)
3P = (20, 3)	11P = (10, 25)	19P = (2, 6)	27P = (13, 6)	35P = (4, 10)
4P = (15, 27)	12P = (19, 13)	20P = (27, 27)	28P = (14, 6)	36P = (1, 24)
5P = (6, 12)	13P = (16, 27)	21P = (0, 7)	29P = (8, 19)	
6P = (17, 19)	14P = (5, 22)	22P = (3, 28)	30P = (24, 7)	
7P = (24, 22)	15P = (3, 1)	23P = (5, 7)	31P = (17, 10)	

Example 2.3 (group structure) Consider F_2^4 as represented by the reduction polynomial $f(z) = z^4 + z + 1$. The elliptic curve E:$y^2 + xy = x^3 + z^3x^2 + (z^3 + 1)$ defined over F_2^4 has $\#\mathrm{E}(F_2^4) = 22$ .Since 22 does not have any repeated factors, F_2^4 is cyclic. The point P = (z^3, 1) = (1000, 0001) has order 11; its multiples are shown below.

0P = O	3P =(1100, 0000)	6P =(1011, 1001)	9P =(1001, 0110)
1P =(1000, 0001)	4P =(1111, 1011)	7P =(1111, 0100)	10P =(1000, 1001)
2P =(1001, 1111)	5P =(1011, 0010)	8P =(1100, 1100)	

2.3 Elliptic Curve Cryptography

The computational overhead of the RSA-based approach to public-key cryptography increases with the size of the keys. As algorithms for integer factorization have become more and more efficient, the RSA based methods have had to resort to longer keys. Elliptic curve cryptography can provide the same level and type of security as RSA (or Diffie-Hellman) but with much shorter keys. A comparison of the key sizes for three different approaches to encryption for comparable levels of security against brute-force attacks is given by (NIST 2001). What makes this table all the more significant is that for comparable key lengths the computational burdens of RSA and ECC are comparable.

Key Size in bits		
Symmetric Encryption	RSA and Diffie-Hellman	Elliptic Curve
80	1024	60
112	2048	224
128	3072	256
192	7680	384
256	15360	512

Table 2.2 a comparison of key sizes needed to achieve equivalent level of security with three different methods.

Koblitz, (Koblitz 1994) and Miller, (Miller 1986) proposed the Elliptic Curve Cryptosystem. Since that time, ECC has received considerable attention from mathematicians around the world, and no significant weaknesses in the algorithm have been demonstrated. ECC not only permits the reduction of the key size and, but also, ECC is able to do operations very fast. In addition, processing power can be reduced in ECC . These entire features allow ECC to be a convenient environment for smart cards(Woodbury et al. 2000). The point addition in elliptic curves is the basic operation to make it used in cryptography. Because of the much smaller key sizes involved, ECC algorithms can be implemented on smart cards without mathematical coprocessors. Contact less smart cards work only with ECC because other systems require too much induction energy. Since shorter key lengths translate into faster handshaking protocols, ECC is also becoming increasingly important for wireless communications(Lauter 2004).

2.4 Elliptic Curve Discrete Logarithm Problem

This is an analog based on elliptic curves over finite fields of public key cryptosystems which use the multiplicative group of a finite field(Koblitz 1994, Enge 1999).The hardness of the discrete logarithm problem on elliptic curves has offered an advance

in cryptography, and there is computational evidence that suggests that it is even more secure than classical techniques. Let E be an elliptic curve over a finite field F_p. $P, Q \in E$. For $k < p$, let Q = kP. It is relatively easy to find Q given P and k, but hard to determine k given Q and P. This is called discrete logarithm problem. The following example illustrates the discrete logarithm problem.
Example 2.4 E: $y^2 = x^3 + 9x + 17$, p = 23. Q = (4, 5) P = (16, 5)
P = (16, 5); 2P = (20, 20); 3P = (14, 14); 4P = (19, 20); 5P = (13, 10);
6P = (7, 3); 7P = (8, 7); 8P = (12, 17); 9P = (4, 5). The discrete logarithm of Q to base P is 9. In real application k will be so large that it is very difficult to determine k by trial and error.

2.5 Elliptic Curve Digital Signature Algorithm (ECDSA)

Signature algorithm is used for authenticating a device or a message sent by the device. For example consider two devices A and B. To authenticate a message sent by A, the device A signs the message using its private key. The device A sends the message and the signature to the device B. This signature can be verified only by using the public key of device A. Since the device B knows A's public key, it can verify whether the message is indeed sent by A or not.

ECDSA is a variant of the Digital Signature Algorithm (DSA) that operates on elliptic curve groups (Hankerson et al. 2004, Koblitz 1994, Johnson and Menezes 2001). For sending a signed message from A to$B$, both have to agree upon Elliptic Curve domain parameters. Sender A must have a key pair consisting of a private key d_A (a randomly selected integer less than n, where n is the order of the curve, an elliptic curve domain parameter) and a public key $Q_A = d_A$ * G (G is the generator point, an elliptic curve domain parameter). An overview of ECDSA process is defined below.

2.5.1 Signature Generation

For signing a message m by sender$A$, using A's private key d_A

1. Calculate e = HASH (m), where HASH is a cryptographic hash function, such

as SHA-1(Secure Hash Algorithm 1).

2. Select a random integer k from [1, n-1] such that gcd(k,n) =1.
3. Calculate r = x_1 (mod n), where (x_1, y_1) = k * G. If r = 0, go to step 2
4. Calculate s = k^{-1} (e + d_A r)(mod n). If s = 0, go to step 2.
5. The signature is the pair (r, s)

2.5.2 Signature Verification

For B to authenticate$A$'s signature, B must have A's public key Q_A

1. Verify that r and s are integers in [1, n-1]. If not, the signature is invalid.
2. Calculate e = HASH (m), where HASH is the same function used in the signature generation.
3. Calculate w = s^{-1} (mod n).
4. Calculate u_1 = ew (mod n) and u_2 =rw (mod n).
5. Calculate (x_1,y_1) = u_1G + u_2 Q_A.
6. The signature is valid if x_1 = r (mod n), invalid otherwise.

Example 2.5 Let p = 114973; the elliptic curve E :$y^2 = x^3 - 3x + 69424$ and a base point G = (11570, 42257) with order n = 114467. d_A = 86109 then Q_A= d_AG = (6345, 28549); and the message m = "worldof".The hash value e = 1789679805, the signature for the message m is (r, s) as following:

ECDSA SIGNATURE :

1. Select k = 84430 such that $1 \leq k \leq n-1$.
2. Compute kG = (11705, 10585), r = 31167 (mod 114973).
3. Compute s = k^{-1}(e+ d_A r) = 82722 (mod 114973).

ECDSA VERIFICATION:

1. Compute w = s^{-1} = 83035 (mod 114973).
2. Compute
   u_1 = ew = 71001 (mod 114973)
   u_2 = rw = 81909 (mod 114973)
3. Compute
   u_1G = (66931, 53304)
   u_2Q = (88970, 41780), u_1G+u_2Q = (31167, 31627) and
   v = 31167 (mod 114973).
4. v = 31167 (mod 114973)
   r = 31167 (mod 114973).
   We obtain v = r, that is accept the signature.

2.6 Elliptic Curve Diffie Hellman Cryptosystem

ECDH is a key agreement protocol that allows two parties to establish a shared secret key that can be used for private key algorithms (Schroeppel et al. 1995). Both parties exchange some public information to each other. Using this public data and their own private data, these parties calculate the shared secret. Any third party, who doesn't have access to the private details of each device, will not be able to calculate the shared secret from the available public information. An overview of ECDH process is defined below. For generating a shared secret between A and B using ECDH, both have to agree up on Elliptic Curve domain parameters.

2.6.1 Key Exchange

E be an elliptic curve over a finite field F_q,where q is either a prime p or an integer of the form 2^m. P = (x, y) be a point on E with order n. The key exchange between A and B can be accomplished as follows:

1. A selects an integer n_A less than n. This is A's private key. A generates a public key $P_A = n_A$P, a point on E.

2. B sleets a private key n_B and computes a public key $P_B = n_B$P.

3. A generates the secret key K = n_A P_B and B generates the secret key K = n_B P_A.

Example 2.6 Let E: $y^2 = x^3 - 4$ over p = 211. P = (2, 2) be a point with order 240. The private key of A be $n_A = 121$.
$P_A = 121(2, 2) = (115, 48)$ is the public key.
Let $n_B = 203$ be B's private key and $P_B = 203(2, 2) = (130, 203)$ be public key.
The shared secret key is 121(130, 203) = =203(115, 48) = (161, 69).
Since it is practically impossible to find the private key n_A or n_B from the public key K, it is not possible to obtain the shared secret for a third party.

2.7 Elliptic Curve Massey Omura Encryption

Elliptic curve analog of the Massey-Omura system is described (Koblitz 1994) as follows:
Key generation: This scheme only needs two entities to agree on an elliptic curve E over a finite field F_q with N =#E(F_q). Entities need not to publish their public keys.

2.7.1 Encryption

1. A represents the message m as a point $M \in$ E(F_q).

2. A chooses a secret integer r_A with gcd(r_A,N) = 1 and computes $M_1 = r_A$M, and Sends M_1 to B.

3. B chooses a secret integer r_B with gcd (r_B,N) = 1, computes $M_2 = r_B\ M_1$, and sends M_2 to A.

4. A computes $r^{-1}\mathrm{A} \in Z_N$, $M_3 = r^{-1}\mathrm{A}\ M_2$, and sends M_3 to B.

2.7.2 Decryption

1. B computes $r_B^{-1}\ M_3 = r_B^{-1}\ r_A^{-1}\ r_B\ r_A\mathrm{M} = \mathrm{M}$.

2. B recovers the message$m$ from the point M.

2.8 Elliptic Curve ElGamal Cryptosystem

Key generation: Entity B selects a random integer d_B from the interval[1, n-1] as his private key, and publishes $Q_B = d_B\mathrm{G}$ as the public key, where G $\in$ E, an elliptic curve E over a finite field F_q.

2.8.1 Encryption

1. Represent the message m as a point M in E(F_q).

2. Select a random integer r from interval [1, n-1] and compute $C_1 = \mathrm{rG}$.

3. Compute $C_2 = \mathrm{r}Q_B + \mathrm{M}$. ($C_1$, C_2) is the cipher text sent to B.

2.8.2 Decryption

1. M = C_2- $d_B\ C_1$, because
   C_2 - $d_B\ C_1 = \mathrm{r}Q_B + \mathrm{M} - d_B\mathrm{rG} = \mathrm{r}d_B\mathrm{G} + \mathrm{M} - d_B\mathrm{rG}$.

2. Recover the message m from the point M.

2.9 Elliptic Curve Primality Testing

Goldwasser and Kilian, (Goldwasser and Kilian 1999) used elliptic curves for primality testing. The Elliptic curve primality test is an elliptic curve version of the classical Pocklington Lehmer primality test.

Theorem 2.9.1. *Suppose $N > 1$ is an integer co-prime to six. Let E denote an elliptic curve over $\mathbb{Z}$ / $N\mathbb{Z}$. Assume that there exists an integer m which has a prime divisor q with $q > (N^{1/4} + 1)^2$. If a point $P \in E(\mathbb{Z}$ / $N\mathbb{Z})$ can be found such that $mP=O$ and $[m/q]$ $P \neq O$, then N is prime.*

Example 2.9 Let $N = 907$. E: $y^2 = x^3 + 10x - 2$; $q = 71$,
q > $(N^{1/4} + 1)^2 = 42.1$.
Let P = (819, 784), 71P = O, 71 is prime.
Therefore 907 is a prime.

2.10 Factoring Integers with Elliptic Curves

Lenstra,(Lenstra 1987) gave new drive to the study of Elliptic curve by developing an efficient factoring algorithm using elliptic curves. It turned out to be very efficient for factoring large numbers. The method is analogous to Pollard's (p - l)-method, which attempts to find a non-trivial divisor of a given integer n > l.
Let n > 1 be an integer. Let E denote an elliptic curve over $\mathbb{Z}$ / $N\mathbb{Z}$. For some point P $\in$ E($\mathbb{Z}$ / $N\mathbb{Z}$), if kP = O, k some integer, with slope u/v then gcd(v, n) is a factor of n.
Example 2.10 Consider the factorization of the number n = 2773. We choose an elliptic curve say $y^2 = x^3 + 4x + 4$ (mod 2773).
P = (1, 3). The point 2P is (1771, 705). Next we try to calculate 3P, but fail. The slope is 702/ 1770, and gcd (1770, 2773) = 59. Therefore 59 is a factor of 2773.
In the next chapters we discuss the application of elliptic curves in develpoing pseudorandom number generators.

Chapter 3

Pseudorandom Numbers

Random numbers are useful for a variety of purposes, such as generating data encryption keys, simulating and modeling complex phenomena, for selecting random samples from larger data sets and so on. They have also been used aesthetically, for example in literature and music, and are of course ever popular for games and gambling. Unfortunately, true random numbers are very difficult to generate, especially on computers that are typically designed to be deterministic. This brings us to the concept of pseudorandom numbers, which are numbers generated from some random internal values, and that are very hard for an observer to distinguish from true random numbers.

3.1 Pseudorandom Number Generators

As the word 'pseudo' suggests, pseudorandom numbers are not random. Essentially, pseudorandom number generators(PRNG) are algorithms that use mathematical formulae or simply pre calculated tables to produce sequences of numbers that appear random. A good deal of research has gone into pseudorandom number theory, and modern algorithms for generating pseudorandom numbers are so good that the numbers look exactly like they were really random.

3.2 Characteristics of PRNGs

Because of the their practical application in information security systems, random and pseudorandom sequence generators should fulfill the following requirements (Kao and Wong 1998, Knuth 1997).

3.2.1 The PRNG must be reproducible

The PRNG must be deterministic, that is, given sequence of numbers can be reproduced at any time if the starting point in the sequence is known. If the results of simulation studies or estimation procedures are to be verified the PRNG must be reproducible. For cryptography this is less important. PRNGs are not suitable for applications where it is important that the numbers are really unpredictable, such as data encryption and gambling.

3.2.2 The PRNG must have a long period

The period of the PRNG is the number of times the algorithm can be run before the sequence of bits repeats. The period should be long enough to ensure that the PRNG does not cycle in practice. A period of 2^{32} or less is too short, but a period above 2^{60} is sufficient. While periodicity is hardly ever a desirable characteristic, modern PRNGs have a period that is so long that it can be ignored for most practical purposes.

3.2.3 The sequence must be uniform and independent

The generated sequence should be uniform and independent. There are many tests available to ensure independence and uniformity of random numbers such as, TestU01, DIEHARD, SPRNG, and NIST Tests. Most statistical tests compute a p-value which should be uniform over [0, 1].

3.2.4 The PRNG must be efficient

Efficiency is a nice characteristic if the application needs many numbers, and determinism is handy if there is need to replay the same sequence of numbers again at

a later stage. This can be measured by seeing how many numbers can be produced in a fixed amount of time. Integer and bit shift operations will produce the fastest PRNG's.
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. Each bit which is independently generated must attain values of 0 or 1 with equal probability. The sequence being generated must be unpredictable in both forward and reverse directions: this implies that it is impossible to predict the value of the bits, both following and preceding a sample capture, with a probability above 0.5.
The statistical tests may be useful as a first step in determining whether or not a generator is suitable for a particular cryptographic application. However, no set of statistical tests can absolutely certify a generator as appropriate for usage in a particular application, i.e., statistical testing cannot serve as a substitute for cryptanalysis.

3.2.5 The algorithm must be portable

In order a PRNG to be generally useful, it should be portable. This means that it should be relatively easy to implement on a wide range of hardware, operating systems, and programming environments.
In the following sections we will discuss some of the PRNGs in detail.

3.3 Linear Congruential Generator

One of the most popular, extensively used and the oldest pseudorandom number generator is Linear Congruential Generator (LCG)(Lehmer 1951). A list of parameters for linear congruential generators of different sizes is given by L'Ecuyer,(L'Ecuyer 1999). The theory behind them is simple to understand, and they are easily implemented as well as fast. The generator is defined by a recurrence relation:
X_{n+1} = (a X_n + c) (mod m), where X_{n+1} is the sequence of pseudorandom values, and
m, 0 $<$ m , is the modulus

a, $0 < a <$ m , the multiplier

c, $0 \leq c < m$, the increment

$X_0, 0 \leq X_0 < m$, the seed or initial value, are integer constants that specify the generator.

Example 3.1 When m = 10 and X_0 = a = c = 7, the sequence is

6,9,0,7,6,9,0,7,6,9, $\cdots$.

If we take $X_0 = 0$, a = c = 7, the sequence is

7, 6, 9,0,7,6, 9, 0, 7, 6, $\cdots$.

This example shows that the sequence is not always random for all choices of m,a,c and X_0. The period of a general LCG is at most m, and for some choices of a much less than m.

Knuth,(Knuth 1997) gives the period of LCG. If c is nonzero, the LCG will have a full period for all seed values if and only if

- c and m are relatively prime,
- a -1 is divisible by all prime factors of m ,
- a -1 is a multiple of 4 if m is a multiple of 4.

While LCGs are capable of producing decent pseudorandom numbers, they are extremely sensitive to the choice of the coefficients c, m, and a. The most efficient LCGs have an m equal to a power of 2, most often m = 2^{32} or m = 2^{64}, because this allows the modulus operation to be computed by merely truncating all but the rightmost 32 or 64 bits. The Lehmer random number generator sometimes also referred to as the Park-Miller random number generator (Park and Miller 1998) is a variant of linear congruential generator that operates in the multiplicative group of integers modulo n.

A general formula of a random number generator of this type is:

X_{n+1} = (aX_n) mod m where the modulus m is a prime number or a power of a prime number, the multiplier a is an element of high multiplicative order modulo m(e.g.,a primitive root modulo m),and the seed X_0 is co-prime to m. A list of parameters for multiplicative congruential generators of different sizes is given by L'Ecuyer,(L'Ecuyer 1999).

3.4 Linear Feedback Shift Registers (LFSR)

Random numbers are required in a wide variety of applications. As digital systems become faster and denser, it is necessary, to implement random number generators directly in hardware. Feedback shift registers (FSR) are most useful in designing and generating pseudorandom or pseudonoise sequences. This is due to their simplicity of defining rules and their capability of generating sequences with much longer period. Approaches using FSR sequences solve the following two basic problems in most applications: cryptographic secrecy and ease of reproducing the same sequence (Song 2003).

A linear feedback shift register (LFSR) is a shift register whose input bit is a linear function of its previous state. A shift register is a device whose identifying function is to shift its contents into adjacent positions within the register or, in the case of the position on the end, out of the register. The position on the other end is left empty unless some new content is shifted into the register. Feedback shift registers are the building blocks for constructing key stream generators.
A feedback shift register is made up of two parts:

- A shift register
- Feedback Function

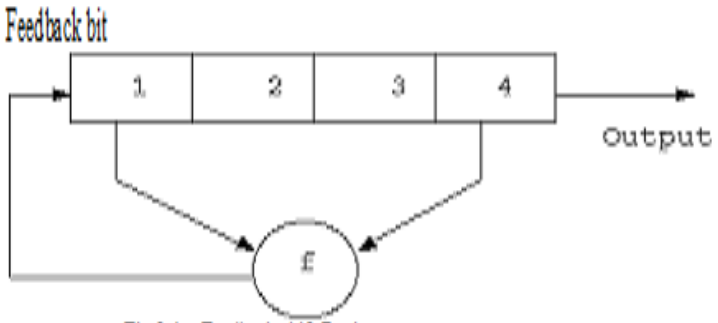


Fig 3.1 Feedback shift Register

An n stage shift register stores a sequence of n bits. At each clock pulse all the bits in the shift register are shifted one bit to the right. The new left most bit is

computed as a function of the other bits in the register. If the function is linear then the shift register is called Linear Feedback Shift Register (LFSR). Otherwise Non Linear Feedback Shift Register (NLFSR). The initial value of the LFSR is called the seed, and because the operation of the register is deterministic, the sequence of values produced by the register is completely determined by its current (or previous) state. Likewise, because the register has a finite number of possible states, it must eventually enter a repeating cycle. However, a LFSR with a well-chosen feedback function can produce a sequence of bits which appears random and which has a very long cycle. The outputs that influence the input are called taps .A maximal LFSR produces an n-sequence (i.e. cycles through all possible 2^n- 1 states within the shift register except the state where all bits are zero), unless it contains all zeros, in which case it will never change.

The tap sequence of an LFSR can be represented as a polynomial mod 2. This means that the coefficients of the polynomial must be 1's or 0's. This is called the feedback polynomial or characteristic polynomial. For example, if the taps are at the 16th, 14th, 13th and 11th bits , the resulting LFSR polynomial is $x^{11} + x^{13} + x^{14} + x^{16} + 1$. The 'one' in the polynomial does not correspond to a tap. The powers of the terms represent the tapped bits, counting from the left.

- LFSR is maximal only if the polynomial is primitive.
- The LFSR will only be maximal if the number of taps is even.
- The tap values in a maximal LFSR will be relatively prime.
- There can be more than one maximal tap sequence for a given LFSR length.

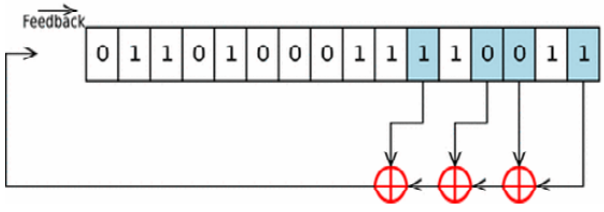


Fig 3.2 LFSR with taps at 16th, 14th, 13th and 11th bits

LFSR can be implemented in hardware and this makes them useful in applications that require fast generation of a pseudorandom sequence.LFSRs have been used as pseudorandom number generators in stream ciphers, especially in military Cryptography, due to the ease of construction from simple electronic circuits, long periods and very uniformly distributed outputs. But due to the linearity of output, cryptanalysis is easier.

3.5 Blum Blum Shub Pseudorandom Number Generator (BBS)

Blum Blum Shub pseudorandom number generator was proposed in 1986 by Lenore Blum, Manuel Blum and Michael Shub, (Blum et al. 1986). It is described by

$$x_{n+1} = x_n^2 \bmod \mathrm{M}$$

where M = pq is the product of two large primes p and q. The two primes, p and q, should both be congruent to 3 (mod 4) and gcd($\phi(p-1)$, $\phi(q-1)$) should be small (this makes the cycle length large). The output is commonly the bit parity of x_{n+1} or one or more of the least significant bits of x_{n+1}.

Example 3.2 Let p = 47, q = 67 and $x_0 = 6$, the seed. We can expect to get a large cycle length for those small numbers, because
gcd($\phi(p-1)$, $\phi(q-1)$)=2. The generator creates the sequence,
36,1296,1199,1657,2870,2265,504,2096,361,1212,1510,224,2941,2327,.....
The output is,
Least Significant bit: 0,0,1,1,0,1,0,0,1,0,0,0,1,1,.....
Even parity bit : 0,1,1,1,1,0,0,1,1,0,1,1,1,0,.....
The generator is not appropriate for use in simulations, because it is not very fast. However, it has an unusually strong security proof, which relates the quality of the generator to the computational difficulty of integer factorization. If integer factorization is difficult then BBS with large M will have an output free from any nonrandom patterns that can be discovered with any reasonable amount of calculation. This makes it as secure as other encryption technologies tied to the factorization problem, such as RSA encryption.

3.6 Mersenne Twister

The Mersenne twister is a PRNG developed by Matsumoto and Nishimura (Matsumoto and Nishimura 1998). It provides for fast generation of very high-quality pseudorandom numbers, having been designed specifically to rectify many of the flaws found in older algorithms. Its name derives from the fact that period length is chosen to be a Mersenne prime.

There are at least two common variants of the algorithm, differing only in the size of the Mersenne primes used. The newer and more commonly used one is the Mersenne Twister MT19937, with 32-bit word length. There is also a variant with 64-bit word length, MT19937-64, which generates a different sequence. It has an incredibly long period, $2^{19937} - 1$(more than 106001). And it has 623 dimensions of equidistribution, meaning that all sequences up to 623 numbers long are equally probable. The generator is also fast. For a k-bit word length, the Mersenne Twister generates numbers with an almost uniform distribution in the range [0,2^k- 1].

3.7 PRNG using Elliptic Curves

Elliptic curve sequences with strong cryptographic properties have been studied in the literature. Gong and Lam (Gong and Lam 2002) have introduced linear feedback shift register sequences (LFSR) over the group of the elliptic curve points, and a construction of binary sequences obtained from these LFSR sequences. Kaliski,(Kaliski 1987) presented a new pseudo-random bit generator based on elliptic curves. Chen and Xiao,(Chen and Xiao n.d.) constructed some families of binary sequences using elliptic curves. Beelen and Doumen,(Beelen and Doumen 2002) produced pseudorandom sequences using both additive and multiplicative characters on elliptic curves.

3.7.1 PRNG using Elliptic Curves and Linear Feedback Shift Registers

This is an elliptic curve based pseudorandom number generator. Lehmer's LCG is applied to points on elliptic curve E defined over a finite field F_p, where p is a prime,

to generate random sequence of points on E. This sequence of points is combined with the random integer sequence generated using LFSR defined over $GF_{(p)}$ .The detailed flow chart is given in fig 3.4.

3.7.2 Algorithm:

1. Choose a Galois field over $GF_{(p)}$ and an elliptic curve E of order O_E.
2. Find all O_E -1 points on elliptic curve and find orders of each of the points.
3. Select the point P with highest order say N.
4. Choose an integer a $<O_E$.
5. Compute P_{i+1} = [a] P_i + B, where seed is the initial point $P_0 \in$ E and B is a random point on E.
6. If [a] Pi = - B, then P_{i+1} = O, the point at infinity,then
7. Replace P_{i+1} by $P_m \in$ E. (P_m is any arbitrary point except P_0)
8. Let K_i, sequence generated from LFSR.
9. Generate key sequence K_i $P_{i+1} = (x_i, y_i)$.

The x coordinate or the y coordinate or even XOR of these two can be taken as random number.

3.7.3 Block Diagram:

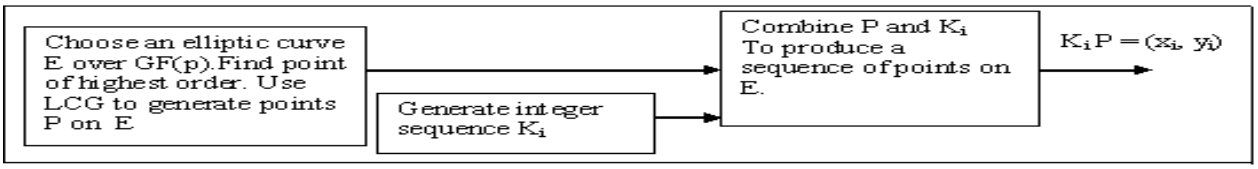


Fig 3.3 block diagram

3.7.4 Flow chart

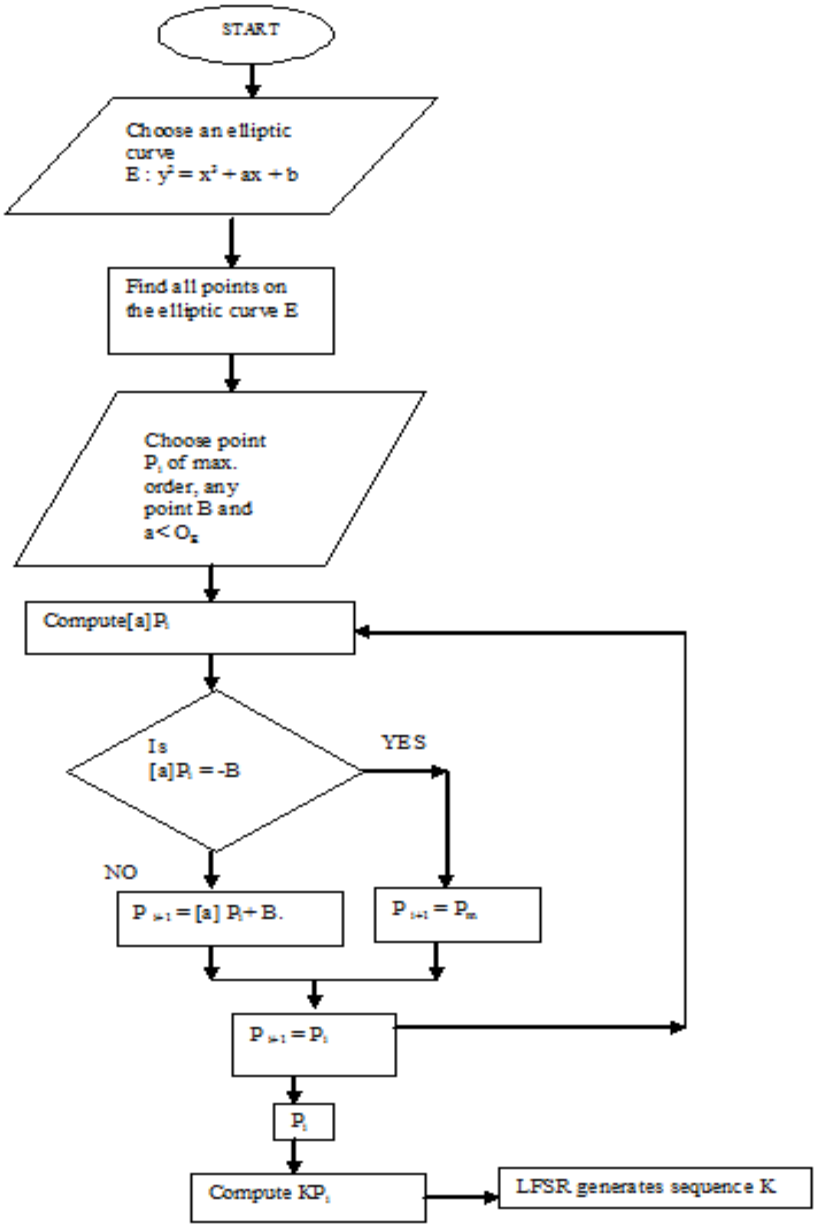


Fig 3.4 Detailed flowchart

Example 3.3 E: $y^2 = x^3 + x + 1$ p=881

A sequence of 20000 bits generated is tested using FIPS test suite. The following table shows the result.

Test	sequence(x_i)		sequence(y_i)		sequence($x_i \bigoplus y_i$)	
Mono bit	9947		9766		9872	
Long run	0		0		0	
Poker	0		1		0	
Run Tests	Run 0	Run 1	Run 0	Run 1	Run 0	Run 1
1	2654	2571	2427	2372	2598	2486
2	1288	1368	1348	1197	1358	1293
3	592	584	518	619	685	701
4	256	351	371	286	319	297
5	176	183	208	194	156	162
$\geq$6	165	102	183	129	205	196

Table 3.1 FIPS test results for the sequence of bits generated on E(F_{881})

Example 3.4 E: $y^2 = x^3 + x + 1$, p=4909

Test	sequence(x_i)		sequence(y_i)		sequence($x_i \bigoplus y_i$)	
Mono bit	9958		9750		9842	
Long run	0		0		0	
Poker	1		0		0	
Run Tests	Run 0	Run 1	Run 0	Run 1	Run 0	Run 1
1	2573	2639	2096	2371	2482	2679
2	1352	1298	1306	1336	1299	1262
3	571	597	693	718	721	674
4	379	283	317	315	298	321
5	198	179	201	176	201	182
$\geq$6	188	193	198	165	194	169

Table 3.2 FIPS test results for the sequence of bits generated on E(F_{4909})

Example 3.5 E:$y^2 = x^3 + 4x + 20$, p= 2383

Test	sequence(x_i)		sequence(y_i)		sequence($x_i \bigoplus y_i$)	
Mono bit	9843		9792		9957	
Long run	0		0		0	
Poker	2		0		1	
Run Tests	Run 0	Run 1	Run 0	Run 1	Run 0	Run 1
1	2456	2367	2592	2275	2638	2469
2	1297	1187	1316	1290	1327	1264
3	585	628	591	617	692	710
4	247	285	319	296	347	339
5	119	126	193	178	128	195
$\geq$6	107	109	188	197	153	173

Table 3.2 FIPS test results for the sequence of bits generated on E(F_{4909})

3.8 Remarks

The generator produced sequences of long period.In all the examples, sequences (x_i),(y_i) and sequence($x_i \bigoplus y_i$) have satisfied the bounds specified in the FIPS test except for Poker's test. In Poker's test only one or two patterns were greater than 428.So we can use this generator as a source of random numbers.

Chapter 4

Fibonacci Sequence

4.1 Definition

The sequence 0,1,1,2,3,5,8,13,21,34,55,89,... is the well known Fibonacci Sequence. It is defined by the recurrence relation

$F_n = F_{n-1} + F_{n-2}$, with seed values $F_0 = 0$ and $F_1 = 1$.

A Lagged Fibonacci generator (LFG),(Mascagni et al. 1995*b*, Mascagni et al. 1995*c*) is an example of a pseudorandom number generator. This class of random number generators is aimed at being an improvement on the 'standard' linear congruential generator. These are based on a generalisation of the Fibonacci sequence. LFG produce very high quality random sequences and have remarkably long periods. LFGs are particularly suitable for generating many streams of independent random numbers in parallel.

A family of pseudo-random number generators of the form:

$S_n = S_{n-j} + S_{n-k} (\text{mod m})$, where j $>$ k $>$ 0. Instead of two initial values, j initial values, S_0,S_1,S_2 ... , S_{j-1}, are needed in order to compute the next element of the sequence. Here, j and k are called "lags".

4.2 A New Class of Generators

We will discuss a new class of random number generators (Marsaglia 1992).
Consider the classical Fibonacci sequence

0,1,1,2,3,5,8,13,21,34,55,89,... Reducing each term mod 10, we get the sequence

0,1,1,2,3,5,8,3,1,4,5,9,... This is an example of lagged Fibonacci sequence with lags j = 2 and k = 1 and the binary operation is

$u \diamond v$ =(u + v) mod 10.

The sequence consists of all 1 x 2 vectors x = (x_1, x_2) with an iterating function f defined by f (x_1, x_2) = (x_2, (x_1 +x_2) mod 10).

Example 4.1 The following table gives the sequence produced by the generator of period 60, with starting values (0,1).

(0,1)	(1,1)	(1,2)	(2,3)	(3,5)	(5,8)	(8,3)	(3,1)	(1,4)	(4,5)
(5,9)	(9,4)	(4,3)	(3,7)	(7,0)	(0,7)	(7,7)	(7,4)	(4,1)	(1,5)
(5,6)	(6,1)	(1,7)	(7,8)	(8,5)	(5,3)	(3,8)	(8,1)	(1,9)	(9,0)
(0,9)	(9,9)	(9,8)	(8,7)	(7,5)	(5,2)	(2,7)	(7,9)	(9,6)	(6,5)
(5,1)	(1,6)	(6,7)	(7,3)	(3,0)	(0,3)	(3,3)	(3,6)	(6,9)	(9,5)
(5,4)	(4,9)	(9,3)	(3,2)	(2,5)	(5,7)	(7,2)	(2,9)	(9,1)	(1,0)

Table 4.1 Sequence generated by f (x_1, x_2) = (x_2, (x_1 +x_2) mod 10 with seed(0,1).

Example 4.2 The table gives the sequence produced by the generator with seed (0,2), having period 20.

(0,2)	(2,2)	(2,4)	(4,6)	(6,0)	(0,6)	(6,6)	(6,2)	(2,8)	(8,0)
(0,8)	(8,8)	(8,6)	(6,4)	(4,0)	(0,4)	(4,4)	(4,8)	(8,2)	(2,0)

Table 4.2 Sequence generated by f (x_1, x_2) = (x_2, (x_1 +x_2) mod 10 with seed(0,2).

Example 4.3 The table gives the sequence produced by the generator with seed (2,3),having period 60.

(2,3)	(3,5)	(5,8)	(8,3)	(3,1)	(1,4)	(4,5)	(5,9)	(9,4)	(4,3)
(3,7)	(7,0)	(0,7)	(7,7)	(7,4)	(4,1)	(1,5)	(5,6)	(6,1)	(1,7)
(7,8)	(8,5)	(5,3)	(3,8)	(8,1)	(1,9)	(9,0)	(0,9)	(9,9)	(9,8)
(8,7)	(7,5)	(5,2)	(2,7)	(7,9)	(9,6)	(6,5)	(5,1)	(1,6)	(6,7)
(7,3)	(3.0)	(0,3)	(3,3)	(3,6)	(6,9)	(9,5)	(5,4)	(4,9)	(9,3)
(3,2)	(2,5)	(5,7)	(7,2)	(2,9)	(9,1)	(1,0)	(0,1)	(1,1)	(1,2)

Table 4.3 Sequence generated by f $(x_1, x_2) = (x_2, (x_1 + x_2) \bmod 10$ with seed(2,3).

We observe from the above examples that the period of the sequence depends on the seed value. In the next section we discuss how to make use of this concept to develop new PRNGs.

4.3 Fibonacci Sequence and Elliptic Curves

Using the concept of lagged Fibonacci sequence, PRNG is developed using elliptic curves overthe finite fields. The points on elliptic curve are used as seed values.

4.4 Lagged (2, 1) Generators

Let $E : y^2 = x^3 + ax + b$ be an elliptic curve over finite field F_p, where p is any fixed prime.
Let P_1, $P_2 \in$ E. Define f$(P_1, P_2) = (P_2, P_1 \circ P_2)$, where the operation '$\circ$' is elliptic point addition.

4.4.1 Algorithm:

1. Select an elliptic curve E over the Galois field GF(p).
2. Choose two points $P_1(x_1,y_1)$, $P_2(x_2,y_2)$ on E.
3. Apply f$(P_1, P_2) = (P_2, P_1 \circ P_2)$.

4. If $x_1 = x_2$, the resulting point is point at infinity. Go To step 2.

This algorithm generates a sequence of points on the elliptic curve with long period. FIPS Test is applied to the generated sequences.

Example 4.6 E: $y^2 = x^3 + x + 1$, p = 881

Test	sequence(x_i)		sequence(y_i)		sequence($x_i \bigoplus y_i$)	
Mono bit	9865		9902		10167	
Long run	0		0		0	
Poker	1		0		2	
Run Tests	Run 0	Run 1	Run 0	Run 1	Run 0	Run 1
1	2432	2398	2613	2590	2614	2481
2	1188	1216	1317	1295	1194	1217
3	548	590	623	674	682	593
4	274	289	317	334	274	341
5	127	194	169	173	190	158
≥6	201	174	183	192	199	152

Table 4.6 FIPS test results for the sequence of bits generated on E(F_{881})

Example 4.7 E: $y^2 = x^3 + x + 1$, p = 1979

Test	sequence(x_i)		sequence(y_i)		sequence($x_i \bigoplus y_i$)	
Mono bit	9735		9910		10027	
Long run	0		0		0	
Poker	0		0		1	
Run Tests	Run 0	Run 1	Run 0	Run 1	Run 0	Run 1
1	2363	2517	2269	2491	2416	2583
2	1253	1178	1204	1319	1295	1349
3	542	617	538	591	579	710
4	251	276	293	316	282	317
5	115	174	196	179	169	145
≥6	147	120	182	167	193	139

Table 4.7 FIPS test results for the sequence of bits generated on E(F_{1979})

Example 4.8 E: $y^2 = x^3 + x + 1$, p = 3407

Test	sequence(x_i)		sequence(y_i)		sequence($x_i \bigoplus y_i$)	
Mono bit	9826		9919		9793	
Long run	0		0		0	
Poker	1		2		1	
Run Tests	Run 0	Run 1	Run 0	Run 1	Run 0	Run 1
1	2517	2539	2269	2319	2594	2489
2	1261	1193	1219	1326	1315	1279
3	585	562	627	693	596	614
4	247	319	275	382	294	329
5	116	183	174	127	150	147
≥6	173	104	169	195	168	126

Table 4.8 FIPS test results for the sequence of bits generated on E(F_{3407})

4.5 Lagged (3,1) Generators

Let $E : y^2 = x^3 + ax + b$ be an elliptic curve over finite field F_p, where p is any fixed prime.
Let P_1, P_2,$P_3 \in$ E. Define f(P_1, P_2,P_3) = (P_2, P_3, $P_1 \circ P_3$), where the operation $\circ$ is elliptic point addition.

4.5.1 Algorithm:

1. Select an elliptic curve E over the Galois field G$F_{(p)}$.
2. Choose three points $P_1(x_1,y_1)$, $P_2(x_2,y_2)$, $P_3(x_3,y_3)$ on E.
3. Apply f(P_1, P_2,P_3) = (P_2, P_3, $P_1 \circ P_3$).
4. If $x_1 = x_3$, the resulting point is point at infinity. Go To step 2.

The algorithm produced a sequence of points on the elliptic curve with long period.The sequence is tested using FIPS test suite for randomness.

Example 4.9 E: $y^2 = x^3 + 4x + 20$, p = 2383

Test	sequence(x_i)		sequence(y_i)		sequence($x_i \bigoplus y_i$)	
Mono bit	9806		9939		9751	
Long run	0		0		0	
Poker	0		1		1	
Run Tests	Run 0	Run 1	Run 0	Run 1	Run 0	Run 1
1	2341	2437	2472	2556	2650	2488
2	1243	1257	1315	1329	1179	1326
3	543	557	618	677	590	681
4	256	304	368	297	250	264
5	118	142	129	116	204	170
≥6	192	144	172	159	168	154

Table 4.9 FIPS test results for the sequence of bits generated on E(F_{2383})

Example 4.10 E: $y^2 = x^3 + 4x + 20$, p = 3407

Test	sequence(x_i)		sequence(y_i)		sequence($x_i \bigoplus y_i$)	
Mono bit	9765		9893		9781	
Long run	0		0		0	
Poker	0		0		1	
Run Tests	Run 0	Run 1	Run 0	Run 1	Run 0	Run 1
1	2374	2612	2445	2489	2517	2560
2	1319	1256	1229	1340	1264	1187
3	539	573	518	627	688	581
4	256	315	344	278	290	323
5	144	162	174	198	180	157
≥6	200	132	154	166	173	185

Table 4.10 FIPS test results for the sequence of bits generated on E(F_{3407})

Example 4.11 E: $y^2 = x^3 + 4x + 20$, p = 4909

Test	sequence(x_i)		sequence(y_i)		sequence($x_i \bigoplus y_i$)	
Mono bit	10124		9828		9943	
Long run	0		0		0	
Poker	0		0		0	
Run Tests	Run 0	Run 1	Run 0	Run 1	Run 0	Run 1
1	2431	2582	2388	2475	2449	2516
2	1287	1249	1316	1304	1259	1193
3	693	547	588	635	586	729
4	354	319	302	260	251	283
5	165	124	118	180	153	149
≥6	133	151	187	155	174	190

Table 4.11 FIPS test results for the sequence of bits generated on E(F_{3407})

4.6 Remarks

In all the examples the sequences (x_i), (y_i) and the sequence($x_i \bigoplus y_i$) have satisfied the bounds specified in the FIPS test except for the Poker's test. In Poker's test, very few 4-bit patters were exceeded 428.

Chapter 5

Multiplicative Congruential Generator

A Linear Congruential Generator (LCG) is defined by a recurrence relation:
X_{n+1} = (a X_n + c) (mod m), where X_{n+1} is the sequence of pseudorandom values, and
m, 0 $<$ m , is the modulus
a, 0 $<$ a $<$ m , the multiplier
c, 0 $\leq c < m$, the increment
$X_0, 0 \leq X_0 < m$, the seed or initial value, are integer constants that specify the generator.
If c = 0, the generator is often called a Multiplicative Congruential Generator (MCG). If m is prime, a is a primitive element modulo m, and $x_0 \neq$, 0 then the generated sequence will have period length = m - 1, and the generator is called a full period MCG.
As computations become faster, increasingly long sequences of random numbers are desirable in applications. Period lengths of the random sequences is a vital issue. But a large period length is not the onlycriterion. The quality of pseudorandom numbers is of central importance in any application requiring their use. It is therefore important to choose a suitable pseudorandom number generator which will generate sequences of large period lengths and also demonstrating good statistical and randomness properties.

One alternative is the class of multiple recursive generators based on the higher order recursion(L'Ecuyer et al. 1993):

$X_n = a_1\ X_{n-1} + ... + a_k X_{n-k}$

L'Ecuyer,(L'Ecuyer 1990) demonstrated that for a given prime modulus, such a generator may produce a sequence with period length of m^k–1. It is possible to get sequences with much better structural properties than simple MLCGs with the same modulus and almost as fast and easy to implement.

Multiple recursive generators for modulus 2 was proposed by Tausworthe, (Tausworthe 1965) and later by Knuth, (Knuth 1997) for prime modulus.

Combining different streams of pseudorandom numbers into a new stream yields an easy way to achieve long periods while keeping the computational costs of generating the numbers low by choosing suitable parameters for each underlying generator(L'Ecuyer 1994).

Fishman,(Fishman 1990) presented the results of a search to find optimal maximal period multipliers for multiplicative congruential random number generators with moduli 2^{32} and 2^{48}.

Before determining the conditions on the multiplier so that x_i has a period length as large as possible, we require the notion of a primitive root modulo m, which is defined in terms of Euler's ϕ -function.

Definition 5.1 Let m be a positive integer.

1. The function $\phi : Z_+ \longrightarrow Z_+$ defined by

   $\phi(m) = \{$t $\in Z_+ : 1 \leq$ t $\leq$ m; (t,m) = 1 $\}$, is called the Euler ϕ-function.

2. If a is an integer relatively prime to m, then the smallest positive integer n such $a^n \equiv 1$(mod m) is called the order of a modulo m, denoted $ord_m(a)$. Furthermore, if $ord_m(a) = \phi$(m), then a is called a primitive root modulo m.

Theorem 5.1 For a pure multiplicative congruential sequence $\{x_i\}$, determined by (x_0, a, m), the maximum possible period length is m, which is attained if:

1. x_0 is relatively prime to m
2. a is a primitive root modulo m.

The multiplicative congruential generator can be generalized to get the extended congruential generator, in which each new integer is a linear combination of previous n-1 integers(Marsaglia 1992).
$X_n = a_1\ X_1 + a_2\ X_2 + a_3 X_3 + ... + a_{n-1}\ X_{n-1}$ (mod p),
where a_1, a_2, a_3, a_4, $\cdots$ a_{n-1} are constants.
This concept is used to develop a pseudorandom number generator using elliptic curves. The construction is explained as follows:

5.1 Algorithm:

1. Select an elliptic curve say, E: $y^2 = x^3 + x + 1$ over a field F_p, where p is some fixed prime.

2. Choose n-1 points say, P_1, P_2, $\cdots$ P_{n-1} on E.

3. Select n-1 integers a_1, a_2, $\cdots$ a_{n-1} such that $a_i <$ order of P_i
   for all i, $1 \leq$ i $\leq$ n-1.

4. Generate sequence of points

   $$P_n = a_1\ P_1 + a_2 P_2 + a_3 P_3 + \cdots + a_{n-1}\ P_{n-1} \text{ (mod } p) \text{ on E.}$$

The extended multiplicative congruential generator seems to produce sequences with fairly long period from simple multiplicative congruential generators with short periods.
For example, consider the elliptic curve E: $y^2 = x^3 + x + 1$; p=101.
With seed (27, 4) and multiplier 2, the random sequence generated has the period 12 and with seed as (28, 8) and multiplier 3, the sequence has period 2. But the combination of these two seeds with respective multipliers, 2 * (27, 4) + 3*(28, 8) resulted in a sequence with a relatively long period of 144. Similarly, 2*(93, 17) generated a sequence with period 12, and 95*(95, 48) also generated a sequence with period 12, but the combination, 2*(93, 17) +95*(95, 48) could generate a sequence with period 144.

Example 5.1 $E : y^2 = x^3 + x + 1$, p =101, n=2

Sl.No	First Generator with period l_1	Second Generator with period l_2	Period of the combination
1	2*(82,30) $l_1 = 12$	3*(83,3) l_2 =4	144
2	85*(30,8) $l_1 = 4$	17*(32,28) $l_2 = 4$	210
3	2*(93,17) l_1 =12	95*(95,46) $l_2 = 12$	144
4	98*(99,30) $l_1 = 6$	59*(100,4) $l_2 = 4$	410

Table 5.1 Generators with respective periods(n=2)

Example 5.2 $E : y^2 = x^3 + x + 1$, p =101, n=3
7*(10,1) has period 2, 11*(11,38) has the period 6 and 11*(12,23) has the period 6, while 7*(10,1) +11*(11,38) + 11*(12,23) resulted in a sequence with period 124.

Sl.No	First Generator with period l_1	Second Generator with period l_2	Third Generator with period l_3	Period of the combination
1	15*(32,28) $l_1 = 12$	5*(35,17) l_2 =6	10*(36,43) l_3 =4	195
2	15*(83,3) $l_1 = 4$	5*(84,11) $l_2 = 6$	2*(85,38) $l_3 = 12$	176
3	7*(86,34) $l_1 = 6$	11*(87,24) $l_2 = 2$	65*(88,35) $l_3 = 2$	135
4	19*(25,20) $l_1 = 4$	23*(27,4) $l_2 = 12$	41*(28,8) $l_3 = 6$	195

Table 5.2 Generators with respective periods(n=3)

Example 5.3 $E : y^2 = x^3 + x + 1$, p =101, n=4

Sl.No	First Generator with period l_1	Second Generator with period l_2	Third Generator with period l_3	Fourth Generator with period l_4	Period of the combination
1	2*(23,24) $l_1 = 12$	2*(25,20) $l_2 = 12$	2*(27, 4) $l_3 = 12$	28*(28,8) $l_4 = 6$	150
2	2*(74,17) $l_1 = 6$	2*(76,39) $l_2 = 6$	2*(79,21) l_3=12	99*(82.30) l_4= 6	205
3	5*(8,4) $l_1 = 6$	8*(10,1) $l_2 = 4$	12*(11,38) l_3= 12	75*(12,23) $l_4 = 3$	296
4	5*(60,27) $l_1 = 2$	8*(61,46) $l_2 = 4$	12*(62,43) $l_3 = 12$	40*(64,65) $l_4 = 12$	190

Table 5.3 Generators with respective periods(n=4)

Example 5.4 $E : y^2 = x^3 + x + 1$, p =101, n=5

Sl.No	First Generator with period l_1	Second Generator with period l_2	Third Generator with period l_3	Fourth Generator with period l_4	Fifth Generator with period l_5	Period of the combination
1	2*(59,45) $l_1 = 12$	2* (60,27) $l_2 = 4$	2*(61,46) $l_3 = 12$	2*(62,43) $l_4 = 12$	27*(64,35) $l_5 = 2$	275
2	2*(92,24) $l_1 = 6$	2*(93,17) $l_2 = 12$	2*(95,48) $l_3 = 12$	2*(99,30) $l_4 = 12$	29*(100,10) $l_5 = 2$	290
3	12*(74,17) l_1 =6	17*(76,39) $l_2 = 2$	19*(79,21) $l_3 = 2$	26*(82,30) l_4= 2	87 *(83,3) $l_5 = 3$	320
4	12*(79,21) $l_1 = 12$	17*(82,30) $l_2 = 4$	19 * (83,3) l_3 =2	26*(84,11) $l_4 = 2$	29*(85,38) $l_5 = 2$	350

Table 5.4 Generators with respective periods(n=5)

Example 5.5 $E : y^2 = x^3 + x + 1$, p =101, n=6

Sl.No	First Generator with period l_1	Second Generator with period l_2	Third Generator with period l_3	Fourth Generator with period l_4	Fifth Generator with period l_5	Sixth Generator with period l_6	Period of the combination
1	2*(84,11) l_1= 12	2*(85,38) l_2= 12	2*(86,34) l_3= 4	2*(87,24) l_4 = 4	2*(88,35) l_5= 12	14*(89,33) l_6= 12	260
2	8*(66,4) l_1= 4	12*(68,47) l_2= 12	62*(72,5) l_3 = 2	25*(74,17) l_4= 6	42*(76,39) l_5= 2	38*(79,21) l_6= 3	205
3	19*(15,19) l_1= 2	15*(21,30) l_2= 12	91*(23,24) l_3= 4	73*(25,20) l_4= 3	42*(27,4) l_5= 4	45*(28,8) l_6= 2	212
4	19*(25,20) l_1= 2	15*(27,4) l_2=12	62*(28,8) l_3= 2	73*(29,49) l_4 = 3	42*(30,8) l_5=4	5*(32,28) l_6= 6	360

Table 5.5 Generators with respective periods(n=6)

In the next section we discuss the randomness properties of the sequences.

5.2 Generator Sequence Properties

A good pseudorandom sequence generator worthy of consideration for encryption purposes is characterized by the following properties (Blake et al. 2005).

- Long period: The generator should have long enough periods in order to avoid the recurrence of the sequence after a short length of time.
- High linear complexity: Sequences with low linear complexity are easily predictable and vulnerable to attack based on their linear structure.
- Reproducibility: Being able to reproduce exactly the same sequence can be necessary and very useful for practical applications.
- Statistical Properties: The generator must pass a battery of statistical tests to validate its randomness attributes.

5.3 Analysis

The sequence produced by the generator was analyzed and tested for conformance to the above characteristics. This analysis is important to check the merit of the sequence produced and thus ascertain confidence in its use for key stream generation.

5.3.1 Long Period

The generated sequences have relatively long periods as desired. The results are illustrated in tables 5.1 to 5.5.

5.3.2 Linear Complexity

One measure of the strength of a random sequence is its linear complexity, as studied by various authors (Chan et al. 1982, Fredricksen 1982, Massey 1986, Massey 1969). The linear complexity of a binary sequence is defined as the length of the shortest linear feedback shift register (LFSR) that generates it. If a sequence has small linear complexity,then the synthesis of a linear equivalent of the sequence generator becomes computationally feasible. The linear complexity of a finite sequence is determined using Massey-Berlekamp algorithm(Massey 1969).It is desirable that a random sequence which can be used as key sequence in stream cipher systems should have a large linear complexity. The linear complexity is an important concept in the analysis of stream ciphers. Any sequence produced over a finite field has a finite linear complexity (Menezes et al. 1997).

Definition 5.1 The linear complexity of an infinite binary sequence s, denoted L(s), is defined as follows:

- if s is the zero sequence s = 0,0, 0,..., then L(s) = 0
- if no LFSR generates s, then L(s) =∞
- otherwise, L(s) is the length of the shortest LFSR that generates s.

Definition 5.2 The linear complexity of a finite binary sequence s^n, denoted L(s^n), is the length of the shortest LFSR that generates a sequence having s^n as its first n terms.

5.3.3 Properties of Linear Complexity

Let s and t be binary sequences.

- For any n $\geq$1, the linear complexity of the subsequence s^n satisfies 0 $\leq$ L(s^n) $\leq$ n.
- L(s^n) = 0 if and only if s^n is the zero sequence of length n.
- L(s^n) = n if and only if s^n = 0, 0, 0,...,0, 1.
- If s is periodic with period N, then L(s) $\leq$ N.
- L(s $\bigoplus$ t)$\leq$ L(s)+L(t),where s $\bigoplus$ t denotes the bitwise XOR of s and t

The Berlekamp-Massey algorithm is an efficient algorithm for determining the linear complexity of a finite binary sequence s^n of length n.

5.3.4 Massey-Berlekamp Algorithm

Given a sequence S^n = s_0,s_1,s_2,s_3,...,s_{n-1}, of length n,the algorithm generates the linear complexity L(S^n) of S^n, 0 $\leq$ L(S^n) $\leq$ n.

1. Initialization: C(D) =1, L=0, m=-1, B(D)=1, N=0.
2. While (N < n) do the following:

   Compute the next discrepancy d: d = (S_n + $\sum_{i=1}^{L} C_i\ S_{N-i}$)mod 2.

   If d = 1 then do the following:

   T (D)= C(D), C(D) =C(D) + B(D).D^{N-m}.

   If L $\leq N/2$ then L= N+1 -L, m = N, B(D) = T(D).

   N = N+1.
3. Return L.

The table of Linear Complexity of binary sequences generated using the extended multiplicative congruential generator for various values of primes and different values of n using the elliptic curve $y^2 = x^3 + x + 1$ is given below,table 5.6.

p \ n	1	2	3	4	5	6
229	417	457	452	478	496	446
337	426	480	494	614	458	485
563	479	494	410	445	424	414
453	413	426	428	489	412	468
881	458	493	468	482	480	414

Table 5.6 Linear complexities of the generated sequences using extended MCG.

5.3.5 Statistical Properties

The NIST test suite was applied to pseudorandom sequences produced by the generator. This test suite is used as a bench mark by NIST in the evaluation of possible candidate generators for the Advanced Encryption Standard (Murphy 2000). The suite conducts a comprehensive battery of statistical tests, in which there are 16 core test strategies. For the battery of tests the level of significance α is set to 0.01. An α of 0.01 means that 1 sequence in 100 sequences is to be rejected. An evaluated p-value $> \alpha$ signifies that the sequence is random with a confidence of 99% (NIST 2001). The NIST suite also has a routine to aid the analysis of results. It generates a report file containing the proportion of the p-values that passed each distinct test. In Figure 5.1 the proportion of the p-values that passed each test is plotted against the test ID number.

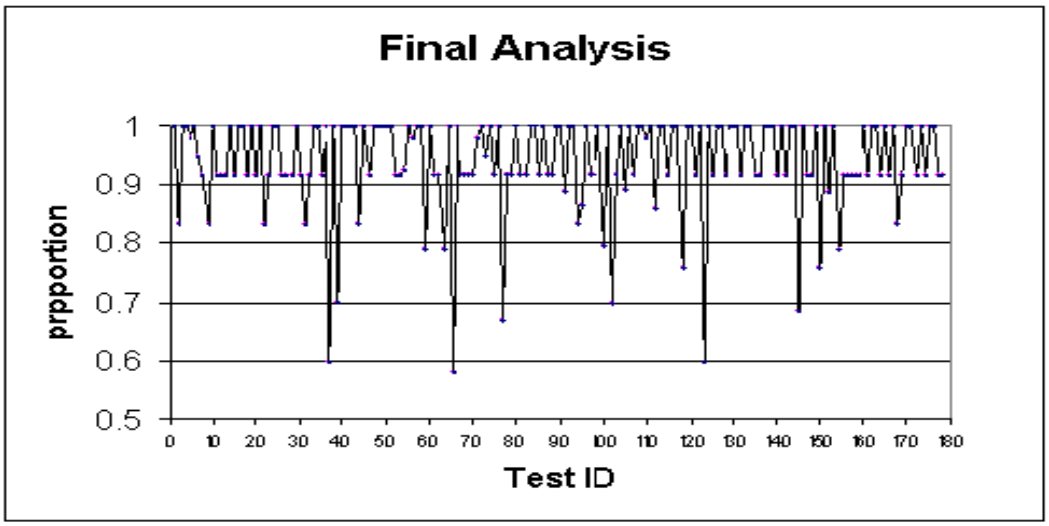


Fig 5.1 NIST Test results for p = 101.

5.4 Remarks

The extended multiplicative congruential method is simple and easy to implement. Experimental results show that the periods of such generators are much longer than those that use the congruential method. Thus, in any Elliptic Curve Cryptosystem, pseudorandom numbers can be easily generated without much overhead.

Chapter 6

Palindromes and Lychrel Numbers

6.1 Introduction

An interesting property of the positive integers in decimal representation is that most of them become palindromes quickly after repeated digit reversal and addition. All 1 digit and 2 digit numbers eventually become palindromes after repeated reversal and addition. About 80 % of all numbers under 10,000 resolve into a palindrome in 4 or fewer steps. About 90% resolve in 7 steps or less.

6.2 Lychrel Numbers

A Lychrel number is a natural number which cannot form a palindrome through the iterative process of repeatedly reversing its base 10 digits and adding the resulting numbers. This process is called the *196-algorithm*.
John Walker began the 196 Palindrome Quest in August 1987. He stopped after 2,415,836 iterations without reaching a palindrome in 1990. In 1995, Tim Irvin used a supercomputer and reached the two million digit mark in only three months without finding a palindrome. Jason Doucette then followed and reached 12.5 million digits in 2000. Since June 2000, Wade VanLandingham has been carrying the task using programs written by various enthusiasts. In 2006, VanLandingham had reached the 300 million digits. A palindrome is yet to be found. *196* is the smallest Lychrel

number.

6.3 Secret Keys

This interesting nature of the number 196 is used to generate keys with the aid of points on elliptic curves over finite fields. The key generation algorithm is very simple as the operation involved is simple reversal and addition of decimal numbers and is achieved as follows: Take any point P(x,y) on an elliptic curve E over the finite field F_p. This pair is the private key.Apply reverse and add digits procedure on *196*, x times to generate key k_1 and y times to generate key k_2 .

6.3.1 Algorithm to Generate Secret Keys

The keys are generated using the following algorithm.

1. The smallest Lychrel number 196 is the Public Key.
2. Generate a point (x, y) on an elliptic curve E. This pair is the private key. Apply reverse and add procedure on *196*, x times to generate key k_1 and y times to generate key k_2 .

Let us generate few keys and analyse them.
Example 6.1 E: $y^2 = x^3 + x + 1$, p = 523. (519, 478) is a point on E. Apply the above algorithm on '519' to generate the key K_1, Similarly, generate the key K_2 using the y coordinate, '478'.
K_1**:**
60859260096454481997330032583646819345580465161711061552189879053658
33840026419038388340495086363456301511399417861499321520354446279059
40328848309136309493286525107809016507110617245408554392954737533003
368929444369116294805.

K2:

153722769557679744601422732137575636758745008755633281659902392532 2
912109278184712921597169553293396317882260428235607069513030649178 2
911120836393198957183226557909447858626585741226225195447976756077227241.

Let us test the randomness of the keys generated by determining the frequency of occurrence of digits 0,1,2,3,4,5,6,7,8 and 9 in these keys and apply Pearson's chi-squared test for independence.

6.4 Pearson's Chi-squared Test for Independence

The chi-square test of independence is used to determine whether there is a significant difference between the expected frequencies and the observed frequencies (Hogg et al. 2004).

A chi-square test is a statistical test commonly used for testing independence and goodness of fit. Testing independence determines whether two or more observations across two populations are dependent on each other (that is, whether one variable helps to estimate the other). Testing for goodness of fit determines if an observed frequency distribution matches a theoretical frequency distribution

Chi-Square Test for Independence: The test is applied when you have two categorical variables from a single population. It is used to determine whether there is a significant association between the two variables. For example, in a clinical trial of a new drug, the alternative hypothesis might be that the new drug has a different effect, on average, compared to that of the current drug. This approach consists of four steps:

- State the hypotheses
- Formulate a plan of analysis
- Analyze sample data
- Interpret the results

The Hypotheses: We use two hypotheses , null hypothesis H_0 and alternative Hypothesis H_a.

A null hypothesis H_0 is a statement that proposes that no statistical significance exists in a set of given observations. The null hypothesis attempts to show that no variation exists between variables, or that a single variable is no different than zero. It is presumed to be true until statistical evidence nullifies it for an alternative hypothesis.
The alternative hypothesis, H_a, is the hypothesis used in hypothesis testing that is contrary to the null hypothesis.
Suppose that we have to test the uniformity of digits in a sequence of N numbers.
Formulate the hypotheses as below:
H_0 : Digits in the sequence are equally distributed .
H_a: Digits in the sequence are not equally distributed.
Plan of Analysis :
The plan of analysis describes how to use sample data to accept or reject the null hypothesis. The plan should specify the following elements:

- Significance level: Often, researchers choose significance levels equal to 0.01, 0.05, or 0.10.
- Test method: Use the chi-square test for independence to determine whether there is a significant relationship between two categorical variables.

Analysis of the Sample Data: Using sample data, find the degrees of freedom, expected frequencies, test statistic, and the P-value associated with the test statistic.

- Degrees of freedom: The degrees of freedom (df) is equal to,

  df = n-1 ,where n is the number of classes of the variable.

- Expected frequencies: The expected frequency counts are computed separately for each class .

We assume,
H_0: All frequencies are equal.
The value of the test-statistic is

$$X^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

where

χ^2 = Pearson's cumulative test statistic;

O_i = an observed frequency;

E_i = an expected (theoretical) frequency, asserted by the null hypothesis;

n = the number of cells in the table.

Interpretation of results: If the calculated value is $\leq$ critical value for certain level of significance, there is no difference between the frequencies. Otherwise we reject H_0.

Example 6.2 E: $y^2 = x^3 + x + 1$, p = 881

First we generate few points on E. Apply the algorithm to generate keys. Then, find the frequencies of digits in the keys. Now apply Chisquare test for each keys separetely. The follwing tables illustrates the results of the test.

Point(x,y)	Frequency of Digits										Accepted/Rejected			
	0	1	2	3	4	5	6	7	8	9	Testvalue	0.1%	1%	5%
(367, 232)	14	14	16	16	19	20	21	12	16	15	4.54	A	A	A
	13	9	8	13	8	11	13	15	5	7	9.37	A	A	A
(381, 738)	20	16	16	20	15	19	14	21	15	13	4.31	A	A	A
	30	32	28	34	32	35	29	33	36	33	1.85	A	A	A
(430,483)	20	22	20	18	21	22	15	16	15	21	3.68	A	A	A
	23	20	19	19	23	20	23	19	25	21	1.96	A	A	A
(519,103)	20	23	19	22	27	25	22	20	22	25	2.60	A	A	A
	5	5	5	4	3	2	6	6	5	7	4.08	A	A	A
(763,501)	38	33	39	30	30	39	32	29	33	28	4.74	A	A	A
	22	19	23	18	26	19	17	27	25	22	5.03	A	A	A

Table 6.1 Chi Square results for the keys generated using points on $E(F_{881})$

Example 6.3 E: $y^2 = x^3 + x + 1$, p = 1979

Point(x,y)	Frequency of Digits										Accepted/Rejected			
	0	1	2	3	4	5	6	7	8	9	Testvalue	0.1%	1%	5%
(124,766)	9	4	3	7	3	3	6	9	9	3	11.86	A	A	A
	40	23	41	31	30	44	24	35	32	29	13.64	A	A	A
(234,1057)	8	10	8	14	12	13	7	14	8	9	6.42	A	A	A
	40	39	43	44	48	45	44	39	47	45	2.08	A	A	A
(316,440)	15	14	15	14	13	15	15	16	13	12	0.96	A	A	A
	19	21	14	20	18	17	22	23	21	22	3.46	A	A	A
(449,1903)	17	18	20	21	22	16	22	19	22	20	2.14	A	A	A
	82	74	81	85	80	89	74	73	77	76	3.15	A	A	A
(874,1383)	42	38	35	33	34	33	35	39	36	40	2.37	A	A	A
	62	62	51	66	55	50	55	56	62	52	4.81	A	A	A

Table 6.2 Chi Square results for the keys generated using points on $E(F_{1979})$

Example 6.4 E: $y^2 = x^3 + x + 1$, p = 4909

Point(x,y)	Frequency of Digits										Accepted/Rejected			
	0	1	2	3	4	5	6	7	8	9	value	0.1%	1%	5%
(899,683)	9	20	9	16	11	6	14	19	16	12	4.95	A	A	A
	4	6	6	5	2	4	3	2	2	2	7.61	A	A	A
(464,1879)	22	16	22	20	30	22	17	13	20	21	9.16	A	A	A
	63	77	76	51	72	70	90	90	97	95	25.56	R	R	A
(937,1960)	38	32	37	39	42	37	39	37	47	38	3.48	A	A	A
	86	84	78	82	80	78	78	81	85	85	1.10	A	A	A
((948,1856)	6	10	9	8	13	4	9	16	14	10	9.36	A	A	A
	9	16	16	17	10	17	9	7	9	19	4.11	A	A	A
(502,1274)	23	15	20	41	23	14	21	18	21	21	22.9	R	A	A
	65	49	46	53	49	42	47	63	37	73	21.6	R	A	A

Table 6.3 Chi Square results for the keys generated using points on $E(F_{4909})$

6.5 Palindromes and Secret Keys

In the previous section we have generated secret keys by employing the non palindromic nature of the decimal number 196. Now we make use of the palindromic nature of decimal numbers to generate the key streams.The non-palindrome number yields a palindrome on repeated reverse and add procedure. Let us see how this simple procedure is deployed to generate complex key streams using points on elliptic curves. This property of integers can be used to generate secret keys. Points on the elliptic curve are taken as seeds to generate secret keys of variable length.
Here, we consider any point P(x, y) on an elliptic curve E over finite feild F_p. Reverse and add digits of x coordinate of the point repeatedly till a palindrome is obtained. The intermediate numbers obtained are concatenated to get a sequence of decimal digits.

6.5.1 Algorithm to Generate Secret Keys

1. Generate a point P(x,y) on an elliptic curve E. Take x-coordinate as the seed.
2. Key = " "
3. Test the number for palindrome. If yes, go to step 5. Otherwise reverse the number and add to the original number to get the sum.
4. Concatenate this sum to the Key. Go to 3.
5. Concatenate this number to the key.

For example, suppose $E : y^2 = x^3 + x + 1$; $p = 523$.
A point on E is (519, 478)
Key = " "
519 + 915 = 1434. Key = 1434: 1434 + 4341 = 5775,a palindrome. Therefore 14345775 is the required key to be used for encryption.
Key = " ". 478 + 874 = 1352. Key = 1352.
1352 + 2531 = 3883. Key = 13523883.
Again we apply chi-square test to check for independence.

Example 6.5 E: $y^2 = x^3 + x + 1$, p = 881 Again we generate points and find the keys using different algorithm and test for independence.

Point(x,y)	Frequency of Digits										Accepted/Rejected			
	0	1	2	3	4	5	6	7	8	9	Testvalue	0.1%	1%	5%
(290,390)	1	2	4	2	0	3	2	0	1	0	12.33	A	A	A
	1	1	0	2	1	1	4	1	2	2	7.0	A	A	A
(297,79)	6	3	2	1	2	0	1	2	2	6	14.6	A	A	A
	4	3	0	1	7	3	2	3	1	1	14.6	A	A	A
(395,380)	4	7	2	0	0	2	1	5	10	3	27.17	R	R	A
	2	4	3	3	3	5	2	2	1	0	7.4	A	A	A
(671,286)	4	2	0	1	7	3	1	2	1	1	17.1	R	A	A
	23	28	14	19	17	14	14	28	25	13	16.75	A	A	A
(692,182)	4	7	2	0	0	2	1	5	10	3	27.18	R	A	A
	2	4	3	3	3	5	2	2	1	0	7.4	A	A	A

Table 6.4 Chi Square results for the keys generated using points on $E(F_{881})$

Example 6.6 E: $y^2 = x^3 + x + 1$, p =1979

Point(x,y)	Frequency of Digits										Accepted/Rejected			
	0	1	2	3	4	5	6	7	8	9	Testvalue	0.1%	1%	5%
(391,938)	5	10	6	5	1	5	3	2	2	1	17.5	R	A	A
	4	2	0	0	1	1	0	4	6	1	20.47	R	A	A
(594,799)	6	3	2	1	2	0	1	2	2	6	14.6	A	A	A
	3	5	0	2	6	1	4	4	2	4	9.97	A	A	A
(698,778)	4	3	5	0	2	3	1	0	0	1	15.2	A	A	A
	1	3	1	3	3	2	3	3	0	0	7.84	A	A	A
(829,562)	9	9	4	3	1	9	2	10	10	4	19.16	R	A	A
	2	4	3	2	2	5	1	2	1	0	8.9	A	A	A
(1962,671)	1	2	1	2	2	3	4	1	1	2	4.68	A	A	A
	4	2	0	1	7	3	1	2	1	1	17.09	R	A	A

Table 6.5 Chi Square results for the keys generated using points on $E(F_{1979})$

Example 6.7 E: $y^2 = x^3 + x + 1$, p =3407

Point(x,y)	Frequency of Digits										Accepted/Rejected			
	0	1	2	3	4	5	6	7	8	9	Testvalue	0.1%	1%	5%
(281,847)	2	4	3	3	3	5	2	2	1	0	7.4	A	A	A
	4	2	0	1	6	3	1	1	0	1	17.3	R	A	A
(375,1509)	3	10	9	5	14	4	9	16	21	11	28	R	R	A
	4	4	2	3	0	3	4	1	0	0	12.8	A	A	A
(880,1484)	23	28	14	19	17	14	14	28	25	13	16.7	A	A	A
	8	11	3	6	1	5	3	1	1	1	27	R	R	A
(463,1574)	2	4	3	2	2	5	1	2	1	0	8.9	A	A	A
	8	9	3	6	2	5	3	2	1	1	18	R	A	A
(3088,1577)	4	9	5	4	3	3	0	1	2	3	16	A	A	A
	9	8	4	3	1	8	2	8	0	4	16.5	A	A	A

Table 6.6 Chi Square results for the keys generated using points on $E(F_{3407})$

6.6 Remarks

Once again we see that that keys of variable length can be generated using this algorithm. Chi-square test at 0.1%, 1%, 5% levels approves the independence of digits in the keys except in few cases. Thus we generated keys of unpredictable lengths using two simple algorithms. In the next section we see the application of these keys.

Chapter 7

One Time Pad

7.1 Introduction

One time pad or OTP, also called Vernam-cipher or the perfect cipher, is a crypto algorithm where the plaintext is combined with a random key. A one-time pad is a very simple yet completely unbreakable symmetric cipher where the same key is used for encryption and decryption of a message. It is the only known method to perform mathematically unbreakable encryption (Shannon 1949). One-time pad was developed by Gilbert Vernam in 1917. A pad is used only once and discarded, hence the name one-time pad.
Encryption: c_i=$m_i \bigoplus k_i$, i = 1,2,3,4,5.....
m_i: plain-text bits.
k_i : key -stream bits.
c_i : cipher-text bits.
Decryption : $m_i = c_i \bigoplus k_i$, i = 1,2,3,4,5.....
One-time pad is secure if four important rules are followed. If these rules are applied correctly, the one-time pad can be proven unbreakable.

- The key is as long as the plaintext.
- The key is truly random.
- There should only be two copies of the key: one for the sender and one for the

receiver.

- The keys are used only once, and both sender and receiver must destroy their key after use.

Although one-time pad is a perfect cipher, it has two major disadvantages. The first problem is the generation of a large quantity of random numbers or letters. The second problem is the key distribution. Since each key can only be used once and has to have the same length as the message, we will need a large number of different keys, physically distributed to both sender and receiver. The costs of secure production, distribution, custody and destruction of one-time pad keys can only be supported by government departments such as military, intelligence services and embassies.
One-time pad encryption is only possible if both sender and receiver are in possession of the same key. Therefore, the keys must be exchanged physically and securely beforehand, by both parties personally or through a trusted courier. This means that the secure communications are expected and planned within a specific time frame. Enough key material must be available for all required communications until a new exchange of keys is possible. Depending upon the situation, a large volume of keys could be required for a short time period, or little key material could be sufficient for a very long time period, up to several years.
Modern computer algorithms such as symmetric block ciphers and asymmetric public key algorithms replaced one-time pads because of practical considerations and solution to key distribution problems. However, although the current crypto algorithms are secure, they could become useless because of new hardware developments, a breakthrough in mathematics such as a faster factoring of numbers, new types of attacks or new quantum computing solutions that speed up brute force attack. Modern crypto algorithms provide practical security and privacy, essential to everyday life. However, sometimes we need everlasting absolute security and privacy, and that's only possible with the one-time pad.
We use the keys generated in the previous chapter as one time pads.

7.2 Encryption and Decryption

Both the algorithms discussed in chapter 5 could generate keys of variable lengths. If the size of the plaintext is greater than that of key, one can easily generate another key and concatenate it with the previous key and get key length same as that of plain text.

Although one-time pad is the only known perfect cipher, it has the major disadvantage of the generation of a large quantity of random numbers. The algorithms described in the previous chapters can generate large number of points easily. We can generate two keys for every point. The key generation algorithms are very simple as well as fast. The algorithms generate variable size keys depending upon the points and the number of iterations used. The key K_1, generated using the x-coordinate is first used to encrypt the plaintext to get intermediate cipher. Then the key K_2, generated using y-coordinate of the point is used to encrypt the intermediate cipher to get the ciphertext. By this we can achieve security of a higher level. If the plaintext size is greater than the key size, another key is generated and concatenated to this key. On the receiver end, first K_2 is used to decrypt the ciphertext and then K_1 to get the original plaintext.

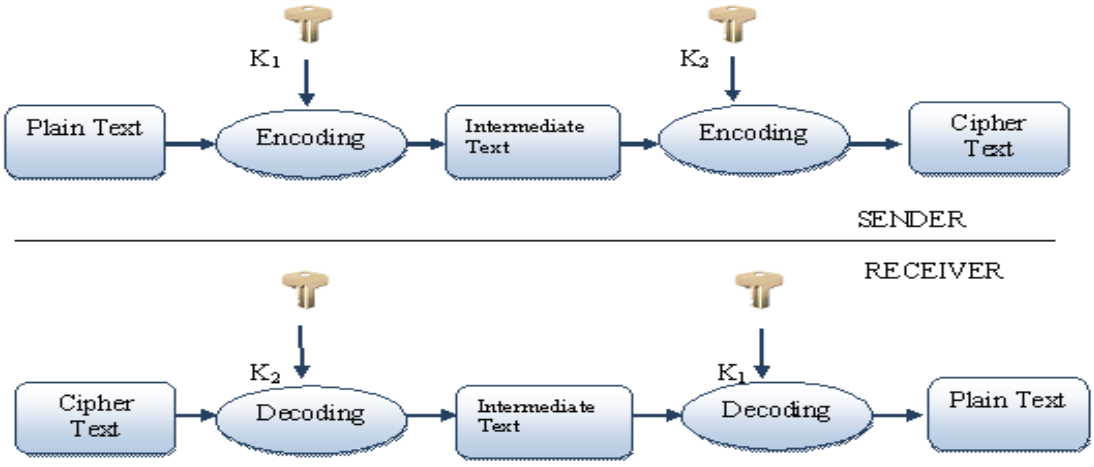


Fig 7.1 Encryption and Decryption

7.3 Steganography

Since the advent of the Internet, the security and integrity of information have been the most important factors of information technology and communication. Cryptography is a technique for securing the secrecy of communication and many different methods have been developed to encrypt and decrypt data . But it is not enough to keep the contents of a message secret, it may also be necessary to keep the existence of the message secret. The technique used to implement this is called steganography.

Steganography is the art and science of writing hidden messages in such a way that no one, apart from the sender and intended recipient, suspects the existence of the message. The word steganography is of Greek origin and means "concealed writing" from the Greek words "steganos" meaning "covered or protected", and "graphei" meaning "writing". Cryptography protects the contents of a message, steganography can be said to protect both messages and communicating parties.

Steganography differs from cryptography in the sense that where cryptography focuses on keeping the contents of a message secret, steganography focuses on keeping the existence of a message secret (Wang and Wang 2004). Steganography and cryptography are ways to protect information from unwanted parties but neither technology alone is perfect and can be compromised. Once the presence of hidden information is revealed or even suspected, the purpose of steganography is partly defeated (Wang and Wang 2004) . The strength of steganography can thus be amplified by combining it with cryptography. While cryptography provides privacy, steganography is intended to provide secrecy. Steganography takes cryptography a step farther by hiding an encrypted message so that no one suspects it exists.

Almost all digital file formats can be used for steganography, but the formats that are more suitable are those with a high degree of redundancy. Redundancy can be defined as the bits of an object that provide accuracy far greater than necessary for the object's use and display (Currie and Irvine 1996). Essentially, the information-hiding process in a steganographic system starts by identifying a cover medium's redundant

bits. The embedding process creates a stego medium by replacing these redundant bits with data from the hidden message.
The most popular data formats used are .bmp, .doc, .gif, .jpeg, .mp3, .txt and .wav.This is primarily due their popularity on the Internet and the ease of use of the steganographic tools that use these data formats. These formats are also popular because of the relative ease by which redundant or noisy data can be removed from them and replaced with a hidden message.
Embedding secret messages in digital images are by far the most widely used of all methods in the digital world of today. Almost any plaintext, cipher text, image and any other media that can be encoded into a bit stream can be hidden in a digital image.

The most well-known steganographic technique in the data hiding field is least-significant-bits (LSBs) substitution. This method embeds the fixed-length secret bits in the same fixed length LSBs of pixels(Wang et al. 2001, Wang and Wang 2004)

7.3.1 Least Significant Bit Technique

Least significant bit (LSB) insertion is a common, simple approach to embedding information in a cover image . The least significant bit (in other words, the 8th bit) of some or all of the bytes inside an image is changed to a bit of the secret message. When using a 24-bit image, a bit of each of the red, green and blue colour components can be used, since they are each represented by a byte. In other words, one can store 3 bits in each pixel. An 800 x 600 pixel image can thus store a total amount of 1,440,000 bits or 180,000 bytes of embedded data. For example a grid for 3 pixels of a 24-bit image can be as follows:

```
(00101101   00011100   11011100)
(10100110   11000100   00001100)
(11010010   10101101   01100011)
```

When the number 200, whose binary representation is 11001000, is embedded into

the least significant bits of this part of the image, the resulting grid is as follows:

(00101101 00011101 11011100)
(10100110 11000101 00001100)
(11010010 10101100 01100011)

Although the number was embedded into the first 8 bytes of the grid, only the 3 underlined bits needed to be changed according to the embedded message. On average, only half of the bits in an image will need to be modified to hide a secret message using the maximum cover size(Johnson and Menezes 2001). Since there are 256 possible intensities of each primary colour, changing the LSB of a pixel results in small changes in the intensity of the colours. These changes cannot be perceived by the human eye - thus the message is successfully hidden. With a well-chosen image, one can even hide the message in the least as well as second to least significant bit and still not see the difference(Johnson and Menezes 2001) This approach is very easy to detect. A slightly more secure system is for the sender and receiver to share a secret key that specifies only certain pixels to be changed.
We can distinguish two kinds of keys: steganographic keys and cryptographic keys (Krenn 2004). A steganographic key controls the embedding and extracting process. For example, it can scatter the message to be embedded over a subset of all suitable places in the carrier medium. Without the key, this subset is unknown, and each sample used to detect embedding by a statistical attack is a mixture of used and unused places which spoils the result. A cryptographic key, however, is used to encrypt the message before it is embedded. For both applications the "secret", which conceals the message, is detached from the actual algorithm in the form of a parameter- the key. If the key is confidential, the steganographic algorithm can be public . It is possible to decide whether the bits read are in fact an encoded message of a potential steganogram only if one has the appropriate decryption key. Encryption is also advisable in addition to steganographic utilities which do not implicitly encrypt.

7.3.2 Block Diagram

The following diagram describes the process of embedding the message in the cover image and extracting.

The data to be transmitted is embedded in an image called, cover-image using the

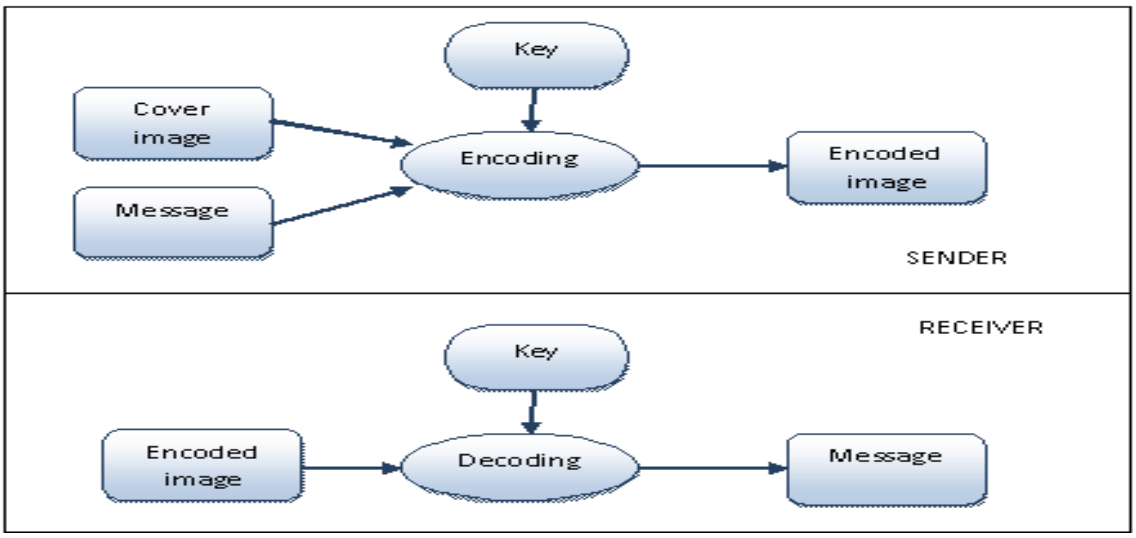


Fig 7.2 Embedding and extracting message

key, called stego-key.The embedded image is called the stego-image.

The process can be described as an expression as follows:

$cover - image$ + message + $stego - key = stego - image$.

The $cover - image$ and $stego - images$ are images in the same format. The data is first encrypted using the key K_1. This encrypted image is embedded into the cover-image at the locations determined using the key K_2.As the keys are random it is very hard to retrieve the data for anybody other than the intended receiver. We take the .bmp files as $cover_image$ and message as .bmp or .jpg images.

Example 7.1 E: $y^2 = x^3 + x + 1$, $p = 1979$. $p = (124{,}766)$

$K_1 = 870338483685310067298611968778702059037600025873847341 67.$

$K_2 = 718737273455553245762780857745790834860100022394426598 42$
$797521589916589102351295595920420806843438173002421263456418217857122$
$515830673077085616446088800267220629314040359226729978915347254907802670286$
$143306598117246543620342093807453475190240294955393053102085609$
$985225787348065155922300009595270985388579872764532555633816477 17.$

We use the key K_1 to encrypt the message and K_2 to embed the encrypted message in the cover image. Next we analyse the stego image to trace the hidden message.

Fig 7.3 a

Fig 7.3 b

Fig 7.3c

Fig 7.3 d

Fig 7.3e

Fig 7.3 f

Fig 7.3 g

Fig 7.3 Cover image, message and stego images

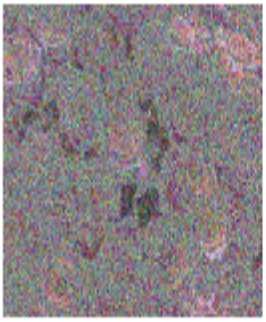


Fig 7.4a Enhanced LSB of Cover image

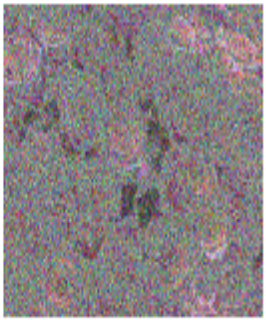


Fig 7.4b Enhanced LSB of Stego image 1

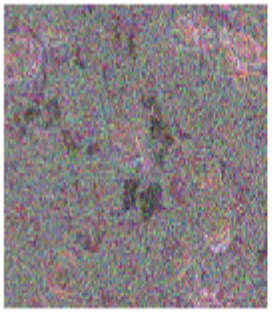


Fig 7.4c Enhanced LSB of Stego image 2

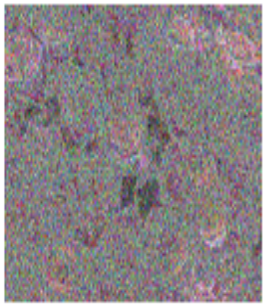


Fig 7.4d Enhanced LSB of Stego image 3

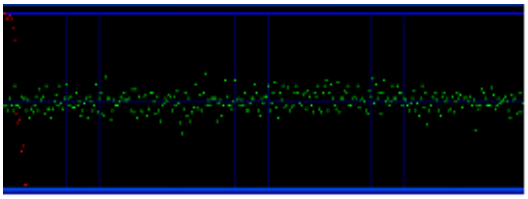
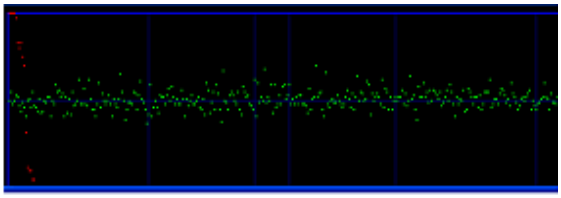


Fig 7.5a Chi square out put of cover image



7.5b Chi square out put of stego image1

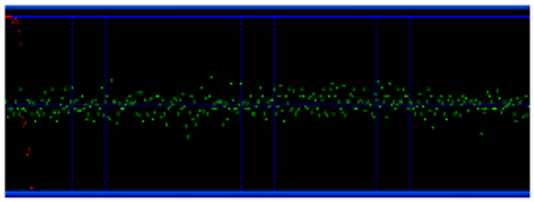
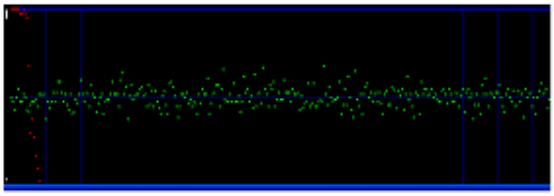
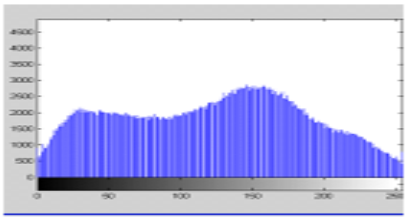


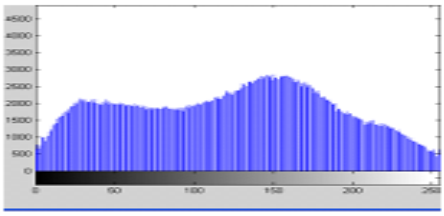
Fig 7.5c Chi square out put of stego image 2



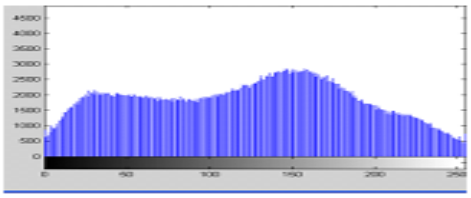
7.5d Chi square out put of stego image 3



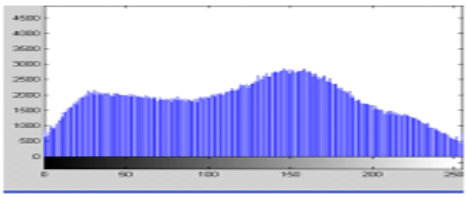
7.6a Histogram of cover image



7.6b Histogram of stego image 1



7.6 c Histogram of stego image 2



7.6d Histogram of stego image

Fig 7.6 Histograms of cover image, message and stego images

7.7a

7.7 b

7.7 c

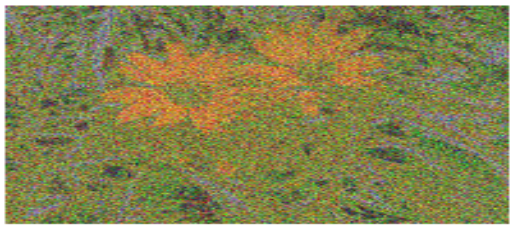
7.7 d

7.7 e

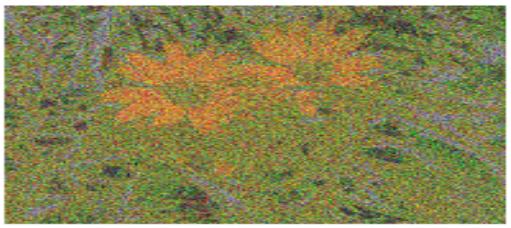
7.7 f

7.7 g

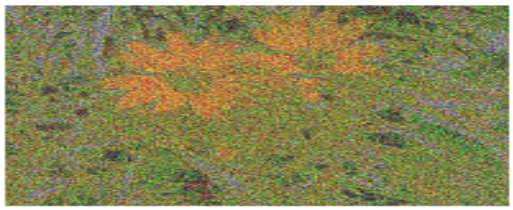
Fig 7.7 Cover image, message and stego images



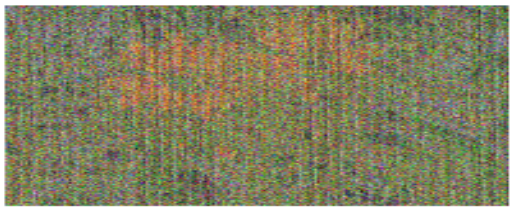
7.8a Enhanced LSB of Cover image



7.8b Enhanced LSB of stego image 1

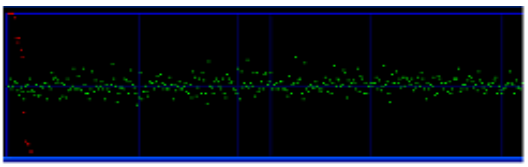


7.8c Enhanced LSB of stego image 2

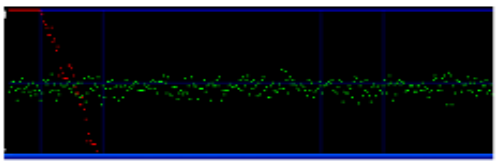


7.8d Enhanced LSB of stego image 3

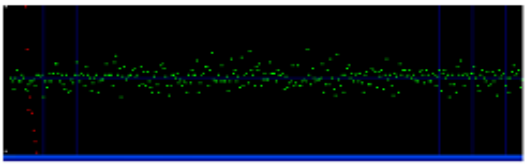
Fig 7.8 Enhanced LSBs of cover image, message and stego images



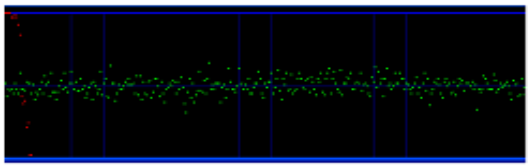
7.9a Chi square output of Cover image



7.9b Chi square output of stego image 1

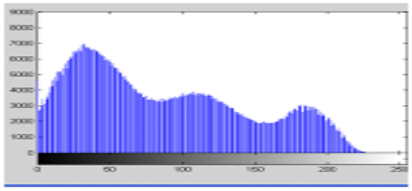


7.9c Enhanced LSB of stego image 2

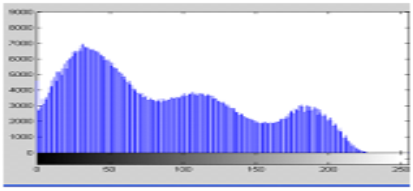


7.9d Enhanced LSB of stego image 3

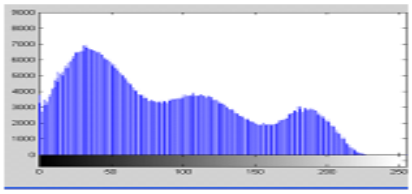
Fig 7.9 Chi square output of cover image, message and stego images



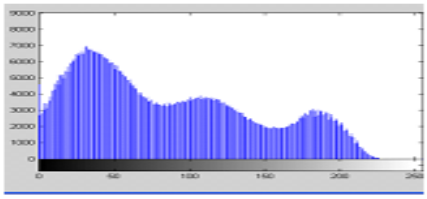
7.10 a Histogram of cover image



7.10 b Histogram of stego image 1



7.10c Histogram of stego image 2



7.10 d Histogram of stego image 3

Fig 7.10 Histograms of cover image, message and stego images

7.11

7.11 b

7.11 c

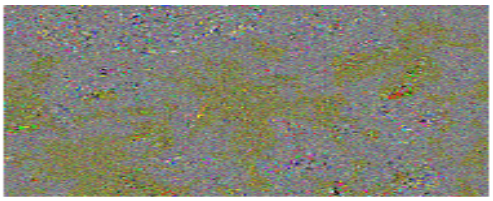
7.11 d

7.11 e

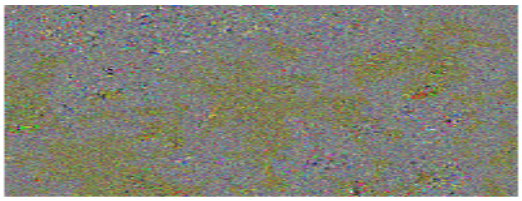
7.11 f

7.11 g

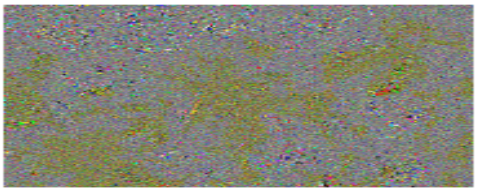
Fig 7.11 cover image, message and stego images



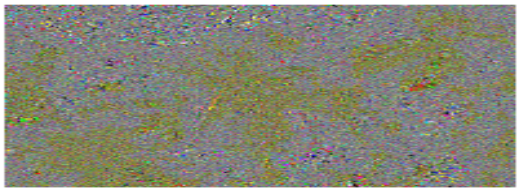
7.12a Enhanced LSB of Cover image



7.12b Enhanced LSB of stego image

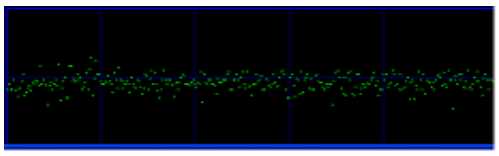


7.12c Enhanced LSB of stego image 2

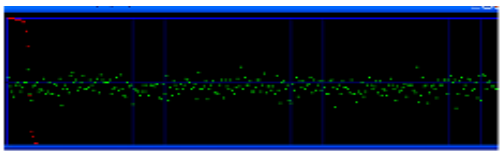


7.12d Enhanced LSB of stego image 3

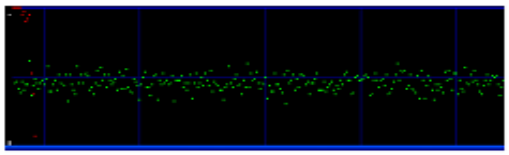
Fig 7.12 Enhanced LSBs of cover image, message



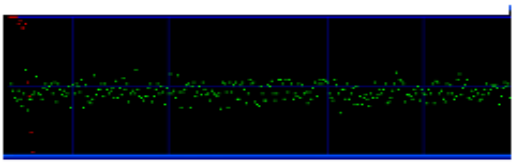
7.13a Chi square output of Cover image



7.13b Chi square output of stego image 1

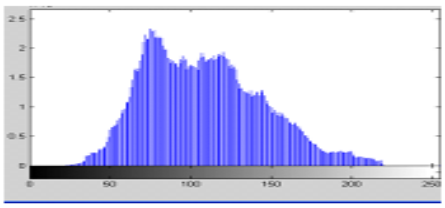


7.13c Enhanced LSB of stego image 2

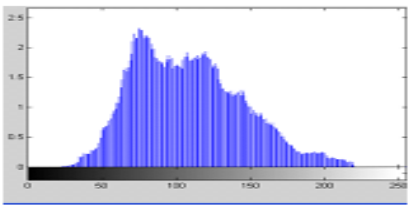


7.13d Enhanced LSB of stego image 3

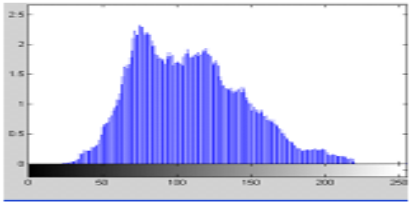
Fig 7.13Chi square output of cover image, message and stego images



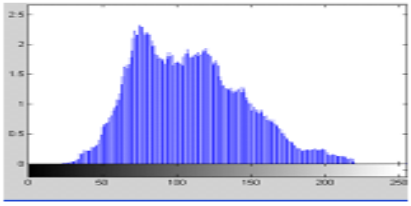
7.14 a Histogram of cover image



7.14 b Histogram of stego image 1



7.14 c Histogram of stego image 2



7.14d Histogram of stego image 3

Fig 7.14 Histograms of cover image, message and stego images

Next we tested the algorithm using the PSNR (Peak Signal-to-Noise Ratio). PSNR is a standard measurement used in steganography technique in order to test the quality of the stego images. The higher the value of PSNR, the more quality the stego image will have.

Stego image / Cover image	Message 1	Message 2	message 3
Coverimage1	56.38	58.46	62.45
Coverimage2	65.74	64.32	72.17
Coverimage3	68.43	74.88	77.42

Table 7.1 PSNR values.

7.3.3 Remarks

Here we have made use of the amalgamation of cryptography and steganography. The cover image and stego images in figures 7.3a, 7.3c, 7.3e and 7.3g look identical. There is no likelihood that human eye can suspect the hidden message inspite of having both cover and stego images together. Therefore a higher level of security is achieved. Even the enhanced LSB of cover image 7.4a and stego images, figures 7.4b, 7.4c and

7.4d look relatively similar, there by hiding the message properly. But, there is some difference between the chi square outputs of figures 7.5a ,7.5b and,7.5d. At the same time, the histograms of the cover image and stego images in figures 7.6a, 7.6b, 7.6c and 7.6d look identical.

Again there are no noticeable differences between the cover image and stego images in figures 7.7a, 7.7c, 7.7e and 7.7g. But the enhanced LSB 0f stego image3 , figure 7.9g failed to hide the message. Even the chi square images are not successful in hiding the message as seen in figures 7.9a, 7.9c, 7.9e and 7.9g.

In the last example also it is possible to detect the presence of message .This due to the large size of message compared to the size of the cover image. There fore by selecting messages of proper size, we can communicate securely in an absolutely imperceptible manner.

It is observed that the stego images have a higher PSNR value. Hence we conclude that the keys are helpful in hiding the messages efficiently.

Chapter 8

Conclusion

The study has given an account of and the reasons for the widespread utility of pseudorandom numbers. The effort explained the central importance of PRNGs in various fields that lead to the development of new generators. Due to the simple addition operation, the PRNGs developed are very efficient and can produce very long sequence of points on the curve with ease. Moreover the sequences generated exhibit randomness properties in accordance with NIST and FIPS statistical test suites. The sequences showed fairly high linear complexity. The lagged Fibonacci generator is more efficient compared to other two generators as it involves only addition of points on the elliptic curve. Thus, in any Elliptic Curve Cryptosystem, pseudorandom numbers can be easily generated without much overhead.

The key generation schemes are very fast and able to generate keys of varied size. The keys generated are fairly random and maybe used for password generation. The keys are suitable for implementation as One Time Pads and also be used in steganography.

Further research may be undertaken in implementing the algorithms on Elliptic curves over other fields. Investigation of PRNGs on different conics over finite fields may also be done.

APPENDIX A

Hasse-Weil Theorem

Let E be an elliptic curve defined over F_q. Then $q+1-2\sqrt{q} \leq \#E(F_q) \leq q+1+2\sqrt{q}$. The interval $[q+1-2\sqrt{q},\ q+1+2\sqrt{q}]$ is called the Hasse interval.An alternate formulation of Hasse's theorem is the following:
if E is defined over Fq, then $\#E(Fq) = q+1-t$ where $|t| \leq 2\sqrt{q}$;
t is called the trace of E over F_q. Since $2\sqrt{q}$ is small relative to q,we have # E(Fq) $\approx q$.

Proof: Let E be an elliptic curve defined over F_q. For each of the q possible values of x, there will be at most 2 y's.
Therefore there will be $2q+1$ points on E. But only half of the elements in F_q have square roots, we expect around half of that number of points.
Let χ be the Legendre symbol, the function
$\chi : F_q \Rightarrow \{0,1\}$, $\chi(0) = 0$.
Thus the number of solutions to $y^2 =$ u is $1 + \chi$(u). Therefore the number of points on $y^2 = x^3 + ax + b$ is
$1 + \sum_{x \epsilon F_q} (1 + \chi(y^2 = x^3 + ax + b)) = 1 + q + \sum_{x \epsilon F_q} \chi(y^2 = x^3 + ax + b)$

This sum is similar to "random walk", where we toss a coin q times and move one step forward if head turns up and one step back ward if tail turns up. D_q be the net distance travelled after q steps. D_q may be positive or negative. So we consider D_q^2.
$D_1^2 = 1$, as after one step ,whether forward or backward movement, square is +1.

The expected distance after q steps for $q > 1$, can be obtained from D_{q-1}.
Thus, $D_q = D_{q-1} + 1$ or $D_{q-1} - 1$) so that ($D_q^2 = D_q^2 + D_q + 1$) or ($D_q^2 = D_q^2 - D_q 1$)
Taking average, $D_q^2 = D_{q-1}^2 + 1$
Since $D_1^2 = 1$, $D_q^2 = q - 1$. Therefore $D_q^2 = q - 1 + 1 =$q;
Hence $D_q = \pm\sqrt{q}$.
Therefore $\sum_{x \epsilon F_q} (1 + \chi(y^2 = x^3 + ax + b)) = \sqrt{q}$.More precisely the sum is bounded by 2 $\sqrt{q}$
Thus the number of points on E over Fq is $q+1-2\sqrt{q} \leq \#E(F_q) \leq q+1+2\sqrt{q}$.

APPENDIX B

Discrete Logarithm Problem

Let F_q be a finite field of q elements so that $q = p^n$ for some prime p and integer n. It is well known that the multiplicative group of nonzero elements of F_q , denoted by F_q^* is a cyclic group of order q.

Thus if α is a generator of this multiplicative group, then every nonzero element β in F_q is given by $\beta = \alpha^x$ for some integer x; in fact for each β there is a unique integer in the range $0 \leq x \leq q-1$ with this property. For a given x and α , the power α^x can be quickly computed by the square and multiply method.

For example, p =1999, α is a primitive root. 3789 can be found to be 1452 (mod 1999) easily.

The inverse problem, i.e., the problem of finding, for a given α and β , the x in the range $0 \leq x \leq q-1$ satisfying $\beta = \alpha^x$, is the discrete logarithm problem.

Given 3, it is not that easy to find x in the rang 0 to 1997 satisfying the equation 3x = 1452 (mod1999).

Elliptic Curve Discrete Logarithm Problem

The discrete logarithm problem is the basis for the security of many crypto systems including the Elliptic Curve Cryptosystem. More specifically, the ECC relies upon the difficulty of the Elliptic Curve Discrete Logarithm Problem (ECDLP).

Let E be an elliptic curve over a finite field Fq. Given any point P on E and an integer k, one can easily find kP=Q(mod q) using repeated doubling and addition.

For example, 100P can be computed as 100P = 2(2(P + 2(2(2(P + 2P))))), performing 6 doublings and 2 additions of points on the curve.

For two points P and Q on an elliptic curve E over a finite field Fq, finding an integer k such that kP=Q(mod q) is very hard .This is called elliptic curve discrete log problem. k is the discrete logarithm of Q to the base.

$y^2 = x^3 + 9x + 17$ over F_{23}, What is the discrete logarithm k of Q = (4,5) to the base P = (16,5)?

One way to find k is to compute multiples of P until Q is found. The first few multiples of P are:
P = (16,5) 2P = (20,20) 3P = (14,14) 4P = (19,20) 5P = (13,10) 6P = (7,3) 7P = (8,7) 8P = (12,17) 9P = (4,5)
Since 9P = (4,5) = Q, the discrete logarithm of Q to the base P is k = 9.
In a real application, k would be large enough such that it would be infeasible to determine k in this manner.

APPENDIX C

The knapsack problem.
Given a set (v_i) of k positive integers and an integer V, find a k-bit integer
n = $(c_{k-l}\ c_{k-2}\ c_{k-3}\ \cdots\ c_1\ c_0)$ where the $c_i \in \{O, 1\}$ and

$$\sum_{i=0}^{k-1} c_i v_i = V$$

if such an n exists.
Note that there may be no solution or many solutions, or there might be a unique solution, depending on the k-tuple (v_i) and the integer V.
A special case of the knapsack problem is the superincreusang knapsack problem.It is known that the general knapsack problem is in a very difficult class of problems, called "NP-complet" problems. This is the case when the v_i, arranged in increasing order, have the property that each one is greater than the sum of all of the earlier v_i.

For example, the 5-tuple (2,3,7,15,31) is a superincreasing sequence.
However, the superincreasing knapsack problem is much, much easier to solve. We look down the v_i, starting with the largest, until we get to the first one that is $\leq$ V. We include the corresponding i in our subset I , replace V by V - v_i, and then continue down the list of v_i until we find one that is less than or equal to this difference.
Continuing in this way, we eventually either obtain a subset of (v_i) which sums to V, or else we exhaust all of (v_i) in which case there is no solution.

If V=24, C_4=0, c_3=1, sothat V=24-15=9 C_2=1 and V=9-7=2, c_1=0, c_0=1. Thus, n=(01101)2 =13.

The multiplicative congruential generator
Let E be an elliptic curve over a finite field F_q.
Let P_1, P_2 ,P_3, $\cdots$, P_{n-1} be n-1 points on E and a_1 , a_2, a_3, $\cdots$,a_{n-1} be n-1 integers.
$P_n = a_1\ P_1 + a_2 P_2 + a_3 P_3 + \cdots + a_{n-1}\ P_{n-1}$ (mod q).

On the right side we have n-1 scalar multiples of points on elliptic curve.Finding each a_iP_i takes $0(a_ilog^3q)$operations.Moreover ECDLP itself is difficult problem. So knapsack version of the multiplicative congruential generator is still harder.

Bibliography

Beelen, P. and Doumen, J. (2002), Pseudorandom sequences from elliptic curves, *in* '6th International Conference on Finite Fields with Applications to Coding Theory, Cryptography and Related Areas, Oaxaca, Mexico', Springer-Verlag, Berlin.

Blake, I.F. Seroussi, G. et al. (2005), *Advances in Elliptic Curve Cryptography*, Cambridge University Press, Cambridge,New York.

Blum, L. Blum, M. et al. (1986), 'A simple unpredictable pseudo random number generator', *SIAM J. Comput.* **15**(2), 364–383.

Chan, A.H. Games, R. et al. (1982), 'On the complexity of debruijn sequences', *Journal of Combinatorial Theory* **33**(3), 233–46.

Chen, Z. and Xiao, G. (n.d.), 'Good ' pseudo-random binary sequences from elliptic curves'.

Crandall, R. and Pomerance, C. (2005), *Prime Numbers: A Computational Perspective*, second edn, Springer-Verlag, New York.

Currie, D. L. and Irvine, C. E. (1996), Surmounting the effects of lossy compression on steganography, 19th National Information Systems Security Conference.

Enge, A. (1999), *Elliptic Curves and Their Applications to Cryptography: an Introduction*, Kluwer Academic Publishers, Dordrecht.

Fishman, G. S. (1990), 'Multiplicative congruential random number generators with modulus 2^{β}: an exhaustive analysis for $\beta = 32$ and a partial analysis for $\beta = 48$', *Math. Comp* **54**(189), 331–344.

Fredricksen, H. (1982), 'A survey of full length nonlinear shift register cycle algorithms', *SIAM Review* **24**, 195–221.

Goldwasser, S. and Kilian, J. (1999), 'Primality testing based on elliptic curves', *Journal of the ACM* **46**(4), 450–472.

Gong, G. and Lam, C. C. Y. (2002), Linear recursive sequences over elliptic curves, Proc. Intern. Conf. on Sequences and Their Applications, Springer-Verlag, pp. 182–196.

Hankerson, D. Menezes, A. et al. (2004), *Guide to Elliptic Curve Cryptography*, Springer-Verlag, New York,Inc.

Hasegawa, A. and Umeno, K. (2002), 'Ip core of statistical test suite of fips 140-2', http://www.design-reuse.com/articles/7946/ip-core-of-statistical-test-suite-of-fips-140-2.html.

Hogg, R V. Craig, A. T. et al. (2004), *Introduction to Mathematical Statistics*, seventh edn, Pearson Education, United States.

Johnson, D. and Menezes, A. (2001), 'The elliptic curve digital signature algorithm', *Intern. J. of Information Security* **1**, 36–63.

Kaliski, B. S. J. (1987), Elliptic curves and cryptography : a pseudorandom bit generator and other tools, PhD thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science.

Kao, C. and Wong, J. (1998), 'Random number generators with long period and sound statistical properties', *Comp. Math. Appl.* **36**(3), 113–121.

Knuth, D. E. (1997), *Art of Computer Programming, Volume 2: Seminumerical Algorithms*, third edn, Addison-Wesley, Reading.

Koblitz, N. (1994), *A Course in Number Theory and Cryptography. Graduate Texts in Mathematics*, second edn, Springer-Verlag.

Krenn, R. (2004), 'Steganography and steganalysis', http://www.krenn.nl/univ/cry/steg/article.pdf.

Lauter, K. (2004), 'The advantages of elliptic curve cryptography for wireless security', *IEEE Wireless Communications* **11**(1), 62–67.

L'Ecuyer, P. (1990), 'Random numbers for simulation', *Communications of the ACM* **33**, 85–98.

L'Ecuyer, P. (1994), 'Uniform random number generation', *Ann. Oper. Res* **53**, 77–120.

L'Ecuyer, P. (1999), 'Tables of linear congruential generators of different sizes and good lattice structure', *Math. Comp.* **68**(225), 249–260.

L'Ecuyer, P. Blouin, F. et al. (1993), 'A search for good multiple recursive random generators', *ACM Trans. Model. Comput. Simul.* **3**, 87–98.

Lehmer, D. H. (1951), 'Mathematical methods in large-scale computing units', *Annals of the Computation Laboratory of Harvard University.*

Lenstra, H. W. (1987), 'Factoring integers with elliptic curves', *Annals of Mathematics* **126**, 649–673.

Marsaglia, G. (1992), The mathematics of random number generators, *in* S. A. Burr, ed., 'The Unreasonable Effectiveness of Number Theory', American Mathematical Soc., pp. 73–90.

Mascagni, M. Cuccaro, S. et al. (1995*a*), 'A fast, high quality and reproducible lagged fibonacci pseudorandom number generator', *Journal of Computational Physics* **119**(2), 211–219.

Mascagni, M. Cuccaro, S. et al. (1995*b*), 'A fast, high quality and reproducible lagged fibonacci pseudorandom number generator', *Journal of Computational Physics* **119**(2), 211–219.

Mascagni, M. Robinson, M. et al. (1995*c*), 'Parallel pseudorandom number generation using additive lagged-fibonacci recursions', *Springer Lecture Notes in Statistics* **106**, 263–277.

Massey, J. (1969), 'Shift-register synthesis and bch decoding', *IEEE Transaction on Information. Theory* **15**, 122–27.

Massey, J. (1986), 'Cryptography and system theory', *Proc. Allerton Conference on Communication Control, Computing* **1**, 1–3.

Matsumoto, M. and Nishimura, T. (1998), 'Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator', *ACM Trans. on Modeling and Computer Simulation* **8**(1), 3–30.

Menezes, A. J. Oorschot, P. et al. (1997), *Handbook of Applied Cryptography*, CRC Press.

Miller, V. S. (1986), 'Use of elliptic curves in cryptography', *Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85* **218**, 417–426.

Murphy, S. (2000), 'The power of nist's statistical testing of aes candidates', http://csrc.nist.gov/archive/aes/round2/conf3/papers/09-smurphy.pdf.

NIST (2001), 'NIST:A statistical test suite for random and pseudorandom number generators for cryptographic applications', http://csrc.nist.gov/groups/ST/toolkit/rng/documents/SP800-22b.pdf.

Park, S. K. and Miller, K. (1998), 'Random number generators: Good ones are hard to find', *Comm.of the ACM* **31**(10), 1192–1201.

Schneier, B. (1996), *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley and Sons, New York.

Schroeppel, R. Orman, H. et al. (1995), 'Fast key exchange with elliptic curve systems', *Advances in Cryptology,Crypto' 95, Lecture Notes in Computer Science,Springer-Verlag* pp. 43–56.

Shannon, C. (1949), 'Communication theory of secrecy systems', *Bell System Technical Journal* **28**(4), 656–715.

Silverman, J. (2009), *The Arithmetic of Elliptic Curves*, Springer-Verlag.

Song, H.-Y. (2003), *Feedback shift register sequences*, 2, Wiley Encyclopedia of Telecommunications,John Wiley and Sons Publication.

Tausworthe, R. C. (1965), 'Random numbers generated by linear recurrence modulo two', *Math. Comp.* **19**, 201–209.

Vancak, O. (2009), Implementation of the statistical randomness tests for cryptographic applications in fpga devices, Master's thesis, Technical University of Kosice, Park Komenskeho,Slovak.

Walker, J. (1990), 'Three years of computing: Final report on the palindrome quest', http://www.fourmilab.ch/documents/threeyears/threeyears.html.

Wang, H. and Wang, S. (2004), 'Cyber warfare: Steganography vs.steganalysis', *Communications of the ACM* **47**(10), 193–196.

Wang, R.Z., L. C. F. et al. (2001), 'Image hiding by optimal lsb substitution and genetic algorithm', *Communications of the ACM* **34**(3), 671–683.

Westfeld, A. and Pfitzmann, A. (2000), 'Attacks on steganographic systems', *Lecture Notes in Computer Science* **1768**, 61–75.

Woodbury, A.F. Bailey, D. V. et al. (2000), Elliptic curve cryptography on smart cards without coprocessors, *in* 'In IFIP CARDIS 2000, Fourth Smart Card Research and Advanced Application Conference', Kluwer, pp. 71–92.

Bio-Data

Karuna Kamath K
Associate professor
Department of Computer Applications
NMAM Institute of Technology
Nitte-574 110,
email: karunapandit@gmail.com
Phone: +91 99640 71440

Educational Qualification:
M. Sc. (Mathematics), Mangalore University.
M. Tech. (Systems Analysis and Computer Applications),NITK,Surathkal.

List of publications:
International Journals

1. Karuna Kamath K and Shankar B R: Elliptic Curves and Multiplicative Congruential Generators, International Journal of Computer Technology and Electronics Engineering, Vol 2, No 4, pp 84-89, 2012. Journal ISSN: 2249-6343.

2. Karuna Kamath K and Shankar B R: One Time Pad via Lychrel number and Elliptic Curve,International Journal of.Computation and Applied Mathematics,Vol 5, No. 2, pp 157-161, 2010. Journal ISSN: 1819-4966. MR2610857(2011f:94118)

3. Karuna Kamath K and Shankar B R: Elliptic Curves and Fibonacci Sequences,International Journal Of. Math. Comput,Vol,4,2009,20-25.
Journal ISSN:0974-5718. MR2596416(2011b:11110)

Conferences

1. Karuna Kamath K and Shankar B R: (2,1) Lagged Fibonacci Generators using Elliptic Curves over finite fields, International Conference on Computer Engineering and Technology, (ICCET-2009), sponsored by IACSIT and IEEE Computer Society, Singapore, Vol II, pp 456-457.

2. Karuna Kamath K and Shankar B R: Pseudo Random Numbers using Elliptic Curves and Linear Feedback Shift Registers, National Workshop on Cryptology-2008, 2008, University of Hyderabad, Hyderabad, pp 165-169.