CRYPTOGRAPHIC ANALYSIS OF RANDOM SEQUENCES

By

SANDHYA RANGINENI

Bachelor of Technology in Computer Science and Engineering

Jawaharlal Nehru Technological University

Anantapur, India

2009

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2011

CRYPTOGRAPHIC ANALYSIS OF RANDOM

SEQUENCES

Thesis Approved:

Dr. Subhash Kak

Thesis Adviser

Dr. Johnson Thomas

Dr. Sanjay Kapil

Dr. Mark E. Payton

Dean of the Graduate College

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

1.1 Random Numbers and Random Sequences:

A random number is a number generated by some process so that the outcome is unpredictable. It is one number that can be drawn from a set of possible numbers, each of which is equally probable (in case of single numbers).

When each number drawn from a set of numbers is statistically independent of the others then that sequence is said to be random sequence. Sometimes it is difficult to find whether the sequence is random or not. Many investigators have proposed definitions to characterize these random number sequences for overcoming the difficulties arising from the same. According to Knuth[1], "Random sequence is a sequence of independent numbers with a specified distribution and a specified probability of falling in any given range of values". Scheneier[2] defined that "A sequence of random numbers is a sequence that has the same statistical properties as random bits, is unpredictable and cannot be reliably reproduced".

There are several types of random sequences of which algorithmically random and pseudo random sequences are most important for our purposes. *Algorithmically random sequence*[3] is a sequence of an infinite sequence of binary digits which appear random to any algorithm. *Pseudo random sequences*[3] are those that typically exhibit statistical randomness in spite of being generated by an entirely deterministic casual process. This process not only makes it easier to produce but also helps in testing and fixation of software.

Some examples for binary random number sequences are as follows:

- 0000000000000000000000

- 01101010000010011110011
- 11011110011101011111011

On the examination of the above sequences, the second and third sequences appear random while the first one does not. From a frequency point of view each of these sequences is equally random, while from an algorithmic point of view the first one is clearly non-random. Random numbers are used for various purposes like generating data encryption and decryption keys, e-commerce, simulating and modeling of complex phenomena and selecting random samples from larger data sets. They also have applications in the arts including literature, music, games and even gambling.

Random numbers that are generated by computers are pseudo-random numbers[3]. True random numbers[3] are numbers which are completely independent of the other numbers in the sequence and where the output is inherently unpredictable.

In a recent article, Anthes [4] summarizes results on generating true random numbers. In particular, he mentions the work at Intel Corp. to use thermal noise on the central processing unit of the computer as random number generator (RNG) [5]. This is not unlike the RNGs based on quantum processes that have been proposed elsewhere [6], but quantum processes come with their own uncertainty [7]-[10].

Randomness is generally measured in terms of probability or of complexity. From the lens of probability, all binary sequences of length *n* are equivalent. From the point of view of complexity, it depends on the algorithm that has been used to generate the sequence. Ritter has provided a summary of several measures of algorithmic complexity [11].

1.2 Types of Random Number Generators:

Given their many applications the need for a fast generation of random sequences using a computer program is essential. Two approaches have been developed for generating random sequences using a computer program: Pseudo-Random Number Generators (PRNGs) and True Random Number Generators (TRNGs)[14].

PRNGs[14] are generators which uses algorithms that have mathematical formulae or just pre-calculated tables to produce random number sequences. Linear congruential method is one of the good example for PRNGs. PRNGs are efficient when they can produce many numbers in a short time. Modern PRNGs have a period that is so long that it can be ignored for most practical purposes.

TRNGs[14] are used to extract the randomness from the physical phenomena and introduce it into the computer. The best way to generate a TRNG is using quantum processes. Another method is to use is a radioactive source that is fed into the computer avoiding any buffering mechanisms in the operating system.

1.3 Random Number Generation Methods:

Traditionally, the first method used to generating random numbers by computer was Von Neumann's mid-square method[15]. The idea behind this method was to take the square of previous random number and extract the middle digits. But the mid-square method was slow and unsatisfactory. To overcome with this drawback, congruential methods were introduced which give better results.

Generally, an acceptable sequence of pseudo random numbers should satisfy the following properties:

1. Uniformly distributed
2. Statistically independent
3. Reproducible and

4. Non-repeating for any desired length

Composite generators[15] combine two separate conguential generators. By combining two separated generators, we can achieve better statistical behavior than separate generators. Here the composite generators uses the second congruential generator to shuffle the output of first congruential generator. The first generator is used to fill a vector with the first k random numbers of size n where as the second generator is used to generate a random integer r which is uniformly distributed over the numbers 1, 2, …, k. The random number which is stored in the r^{th} position is returned as the first random number and then replaces the random number in r^{th} position with a new random number. This process repeats and generates a random sequence.

Tausworthe generators[15] are additive congruential generators which are obtained when the modulus m is equal to 2.

The formula for these generators is:

$$x_i = (a_1x_{i-1} + a_2x_{i-2} + a_nx_{i-n}) \pmod 2$$

where x_i can be either 0 or 1. These generators are independent of the computer used and its word size and also have very large cycles and even produce a stream of bits. Two or more Tausworthe generators can be combined in order to obtain statistically good output.

This is a newer algorithm[15] whose output has very excellent statistical properties when compared to other generators and its period is very long i.e., 2^{19937}-1. This algorithm used 19937 bits long seed value on a very large linear feedback shift register.

1.4 Applications of Random Numbers:

Random numbers has many applications in several fields:

- Monte Carlo simulation method
- Spectrum spreading telecommunication systems
- Generation of primes
- Cryptography
- Computer games and Gambling

A Monte Carlo Method[16] is a technique which uses random numbers and probability to solve problems. It is mainly used when the model is complex, non linear, or has more uncertain parameters. Typically a simulation involves over 10,000 evaluations of a model.

In spread-spectrum[17] random numbers are used in order to generate the signals to spread in the frequency domain.

Random numbers are also used for generation of prime numbers. Random primes are used in RSA encryption. These numbers are uniformly distributed only on the primes in the range but not on the entire range because primes are not uniformly distributed over ranges of positive integers.

In cryptography, random numbers are used for generation of the following[17]:

- ***Nonce*** is an abbreviation of number used once. It is often a random or pseudo-random number generated from an authentication protocol which is used to avoid replay attacks.
- ***Key*** is a piece of information or a parameter which determines the functional output of an algorithm. In encryption, a key transforms plain text into cipher text and cipher text into plain text in decryption. Keys are also used in other cryptographic algorithms such

as digital signature schemes and message authentication codes. Without a key, the algorithm would not produce useful result.

- ***Challenge*** is a protocol where in one party presents a question (challenge) and another party must provide a valid answer (response) to be authenticated. An example for challenge-response protocol is password authentication where the challenge is asking for the password and the valid response is the correct password.
- ***Initialization vector*** is a fixed size input to a cryptographic primitive that should be random or pseudorandom. Some cryptographic primitives require initialization vector only to be non-repeating where the required randomness is derived internally.
- ***Padding byte*** is nothing but padding the input with the padding string of between 1 and 8 bytes to make the total length a multiple of 8 bytes. The value of each byte of the padding string is set to the number of bytes added - i.e. 8 bytes of value 0x08, 7 bytes of value 0x07, 2 bytes of 0x02, or one byte of value 0x01.
- ***Blind Signature*** can be implemented by using a number of public key signing schemes, like RSA and DSA. In blind signature, the message is first blinded by combining it with a random blinding factor. The blinded message is passed to a signer, and then the signer signs it using a standard signing algorithm. The resulting message, along with the blinding factor, can be verified against the signer's public key.

Random numbers plays a vital role in casinos and, in turn, entertainment[18]:

- Role playing games – used to select the roles randomly.
- Card shuffling – used for shuffling cards randomly.
- Lottery tickets – used to obtain random numbers for the tickets.
- Indoor tennis game – used for making new combination of players.
- Dungeons and Dragons – based on the random rolls of dice.

Random graphs[19] are one of the mainstays of modern discrete mathematics. They have been employed extensively as models of real world networks of various types and in epidemiology.

1.5 Statistical tests for Random Number Generators:

The Diehard tests[20] are the statistical tests which are used for measuring the quality of a random number generator. These tests require input as a specially formatted binary file containing 3 million 32-bit integers. After producing this specially formatted binary file, Diehard tests are performed on the resulting file. The tests are:

- ***Overlapping permutations:*** Analyze sequences of five consecutive random numbers. The 120 possible orderings should occur with statistically equal probability.
- ***Birthday Spacings:*** Choose random points on a large interval. The spacings between the points should be asymptotically exponentially distributed. The name is based on the birthday paradox.
- ***Monkey Tests:*** Treat sequences of some number of bits as "words". Count the overlapping words in a stream. The number of "words" that don't appear should follow a known distribution. The name is based on the infinite monkey theorem.
- ***Ranks Of Matrices:*** Select some number of bits from some number of random numbers to form a matrix over {0,1}, then determine the rank of the matrix. Count the ranks.

- ***Parking Lot Test:*** Randomly place unit circles in a 100 x 100 square. If the circle overlaps an existing one, try again. After 12,000 tries, the number of successfully "parked" circles should follow a certain normal distribution.
- ***Count The 1s:*** Count the 1 bits in each of either successive or chosen bytes. Convert the counts to "letters", and count the occurrences of five-letter "words".
- ***Minimum Distance Test:*** Randomly place 8,000 points in a 10,000 x 10,000 square, then find the minimum distance between the pairs. The square of this distance should be exponentially distributed with a certain mean.
- ***Random Spheres Test:*** Randomly choose 4,000 points in a cube of edge 1,000. Center a sphere on each point, whose radius is the minimum distance to another point. The smallest sphere's volume should be exponentially distributed with a certain mean.
- ***The Squeeze Test:*** Multiply 2^{31} by random floats on [0,1) until you reach 1. Repeat this 100,000 times. The number of floats needed to reach 1 should follow a certain distribution.
- ***Runs Test:*** Generate a long sequence of random floats on [0,1). Count ascending and descending runs. The counts should follow a certain distribution.
- ***Overlapping Sums Test:*** Generate a long sequence of random floats on [0,1). Add sequences of 100 consecutive floats. The sums should be normally distributed with characteristic mean and sigma.
- ***The Craps Test:*** Play 200,000 games of craps, counting the wins and the number of throws per game. Each count should follow a certain distribution.

CHAPTER II

COMPARISION OF SOME RANDOM SEQUENCES

2.1 Binary Decimal Sequences:

Binary random sequences[21] are generated by starting with the decimal sequences. A decimal sequence is obtained by representing a number in a decimal form in a base r and it may terminate, repeat or be aperiodic. For a certain class of decimal sequences of 1/q, q prime, the digits spaced half a period apart add up to r-1, where r is the base in which the sequence is expressed. These decimal sequences are periodic and their randomness properties are checked only in one period. Decimal sequences are known to have good auto correlation properties and they can be used in applications involving pseudorandom sequences.

Any periodic sequence can be represented as a generalized d-sequence *m/n*, where *m* and *n* are suitable natural numbers, i.e., positive integers.

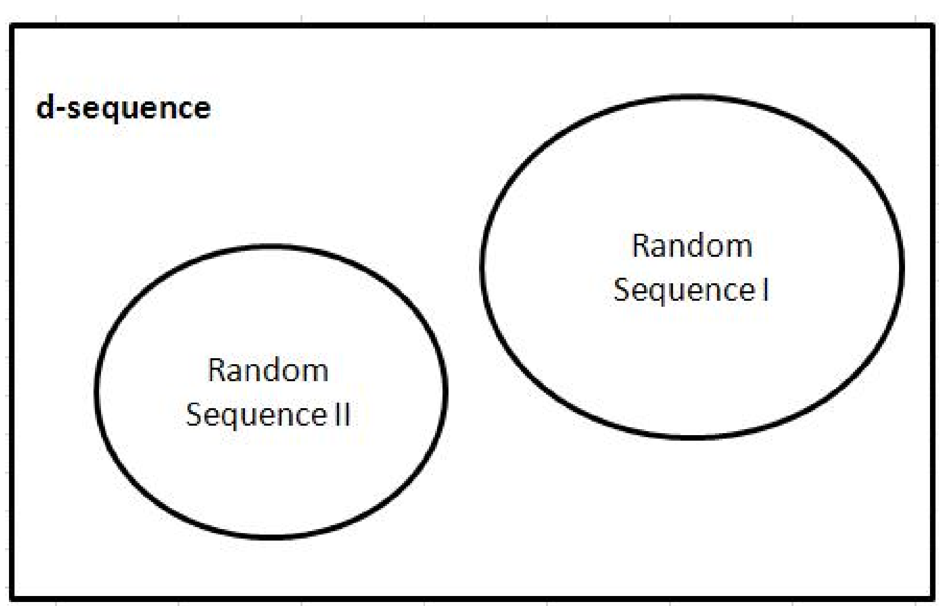

Figure 2.1: General Random sequences as a subset of d-sequences

2.1.1 Properties of Decimal Sequences:

Properties of decimal sequences from [22] to [24], which are summarized below:

Frequency Characteristics:

Theorem 1: Any positive number x may be expressed as a decimal in the scale of r

$$A_1A_2...A_{s+1}.a1.a2\ldots$$

Where $0 \leq A_i < r$, $0 \leq a_i < r$, not all A and a are zero, and an infinity of the a_i are less than (r-1). There is a one to one correspondence between the numbers and the decimals, and

$$X = A_1r^s + A_2r^{s-1} + \ldots + A_{s+1} + a_1/r + a_2/r^2 + \ldots..$$

For example, ¼ can be represented as 0.25 in the scale of 10 and 0.01 in the scale of 2. The decimal sequences of rational and irrational numbers may be possibly used to generate pseudorandom sequences and this is suggested by the following properties of decimals of real numbers.

Theorem 2: Almost all decimals, in any scale, contain all possible digits which mean that the property applies everywhere except to a set of measure zero.

Theorem 3: Almost all decimals, in any base, contain all possible sequences of any number of digits.

Theorem 2 and 3 guarantee that a decimal sequence missing any digit is exceptional to know the behavior of the digits for any particular decimal sequence.

For example: A number x is said to be simply normal in base r if in the decimal of x each of the r possible digits occur with a frequency 1/r, i.e.,

$$\lim n_b / n \rightarrow 1/r \text{ where } n \rightarrow \infty$$

For all b, where the digit b occurs n_b times in the first n places.

Theorem 4: Almost all numbers are normal in any base.

It may be noted, however, that while finite periodic decimal sequences may be simply normal in a given scale, they will not be simply normal in all scales. For example, consider

$$x = 0.0123456789$$

which is simply normal in the scale of 10. In the scale of 1010 the same number is x = .b where b is 123456789, which is not simply normal, 1010-1 digits being missing. So, a normal number cannot be rational. Theorem 4 guarantees the existence of an uncountably infinity of irrational numbers, whose decimal representation would perfectly exhibit all randomness properties. Generating a periodic sequence from its rational number representation is computationally less complex than generating it from an irrational number.

Theorem 5: The decimal for a rational number p/q between 0 and 1 is terminating or recurring, and any terminating and recurring decimal in the scale of 10 is equal to a rational number. If $(p,q)=1$, $q=2^{\alpha}5^{\beta}$, and $\max(\alpha,\beta) = \mu$, then the decimal terminates after μ digits. If $p,q)=1$, $q=2^{\alpha}5^{\beta}Q$, where $Q > 1$, $(Q,10) = 1$, and v is the order of 10(mod q), then the decimal contains μ non-recurring and v recurring digits.

Theorem 6: Suppose $0 < x < 1$, $x = p/q, (p,q) = 1$. If $q = s^{\alpha}t^{\beta}...u^{\gamma}$, where s,t…u are the prime factors of r, and $\mu = \max(\alpha,\beta\ldots\gamma)$, then the decimal for x terminates at the μth digit. If q is prime to r and v is the order of r(mod q), then the decimal is pure recurring and has period of v digits. If $q= s^{\alpha}t^{\beta}...u^{\gamma}Q$, $(Q>1)$, Q is prime to r, and v is the order of r(mod Q), then the decimal is mixed recurring, and has μ non-recurring and v recurring digits.

Theorem 7: A maximum length decimal sequence when multiplied by p, $p < q$, is a cyclic permutation of itself.

The remainders 1,2,…q-1 obtained during the division of 1/q have a one-way correspondence with the coefficients 0,1,…r-1. Since p/q starts off with a remainder rp(mod q) instead of r(mod q), there would be a corresponding shift of a decimal sequence.

Theorem 8: if the decimal sequence, in the scale of r, of p/q; $(p,q)=1$, $p < q$, and $(r,p) = 1$is shifted to the left in a cyclic manner, 1 times, the resulting sequence corresponds to the number $p^1/q, (p^1,q)=1, p^1<q$ where $p^1 = r^1 \times p \pmod q$.

Theorem 9: For a maximum length decimal sequence $1/q = a_1a_2 \ldots a_k,$ $k = q-1$, in the scale of r:

$a_i + a_{k/2+I} = r-1$

For example: x = {1/19} in base r =2

$x \leftrightarrow$ 0 0 0 0 1 1 0 1 0 1 1 1 1 0 0 1 0 1

Here, $a_i + a_{k/2+I} = r-1=1$

The extension to the above theorem is stated below.

Theorem 10: If the period k of the decimal sequence of 1/q, q prime, is even in the scale r:

$a_i + a_{k/2+I} = r-1$

Some additional structural properties of the remainder and the decimal sequences digits are presented below.

Theorem 11: For a maximal length decimal sequence the remainder sequence $m_1m_1\ldots m_k$, $k = q-1$, satisfies the relations

$m_im_{q-1} = m_1 (\text{mod } q)$

$m_im_{l-i} - 2\, m_jm_{l-j} + m_l = 0 (\text{mod } q)$ for all I, j, l.

Theorem 12: The decimal sequence of 1/q, where q is of the form $r^N + 1$ when expressed in the scale of r would be N consecutive zeros followed by N consecutive (r-1)'s.

Distance Properties:

Let the ith remainder in the division of 1/q be represented by m_i, where $m_0 = 1$, $m_i = r\, m_{i-1} – qa_i$. So, the following is obtained.

$m_{i+j} = r^{j+1}m_{i-1} – ql_i(j+1)$

where $l_i\,(j+1) = r^ja_i + r^{j-1}a_{i+1} +\ldots + r\, a_{i+j-1} + a_{i+j}$.

Theorem 13: For a binary decimal sequence 1/q, if $2^m > q$, then all l_i (m) are different.

For such a sequence, all subsequences of length m are different.

Theorem 14: The Hamming Distance d_j between the binary maximum length sequence {1/q} and its jth cyclic shift satisfies

$$d_j \geq k/m,\ j \neq 0,\ j < k,$$

where $2^m > q$, $k = q-1$.

At least one of each m consecutive digits is different from Theorem 13. Hence, the minimum distance between each set of m digits is one. For a total of k such group of digits, the distance is k, and since the sequence considered is m times over, the distance is k/m.

Randomness Properties:

The randomness of a periodic binary sequence of +1's and -1's can be checked by comparing the run characteristics of +1's and -1's as well as its autocorrelation function to that obtained for a normal number where the digits are independent.

Autocorrelation Properties:

Let the equation below represent the auto-correlation function of the decimal sequence $a_1 \ldots a_n$.

$$C(k) = \frac{1}{n} \sum_{j=0}^{n} a_j a_{j+k}$$

For a normal number, the autocorrelation function is:

$C_1(\tau) = E(a_n a_{n+\tau})$

Where the nth digit of the sequence a_n □ {0,1,2, … r-1}. The autocorrelation function is two-valued for a binary random sequence.

2.1.2 Generation of Decimal Sequences:

According to the standard method, the binary d-sequence is generated using the algorithm below[16]:

$a(i) = 2^i \bmod p \bmod 2$

where p is a prime number. The maximum length period (p-1) sequences are generated when 2 is a primitive root of p.

This may be rewritten as:

$$a(0)= 2$$

$$b(i+1) = 2b(i) \bmod q$$

$$a(i)=b(i) \bmod 2$$

To generate decimal expansions of 1/p, one may use the following formula[17]:

If prime ends in 1,

$$a(i) = 9 \times 10^i \bmod p \bmod 10$$

If prime ends in 3,

$$a(i) = 3 \times 10^i \bmod p \bmod 10$$

If prime ends in 7,

$$a(i) = 7 \times 10^i \bmod p \bmod 10$$

If prime ends in 9,

$$a(i) = 10^i \bmod p \bmod 10$$

The above formula produces a random sequence of decimal numbers ranging from 0 to 9. These decimals are then converted to an equivalent binary numbers, which results in a sequence of binary numbers.

2.1.3 Examples for Decimal Sequences:

1) For p=1117

The binary sequence equivalent to the above decimal sequence is:

0000000000111010101010111110001110010100101111011100001111110100000101010001010
0111000101110010010111010000110010011010111011011110010011110100110001010001100
0101110000100001111010110101111110010001111111011011010101001001000111000011000
0010010110010110000111011100101101001001100100011000100001011101100000001111100
1010110101000011100111000001001101000000011001101010110011001110010001000100110
0000010110111010110010010011100110000110100010000010001011010110000011110010000
0001010000101011000101100011101100100001001110110101101111100111001111110111100
1111111101001111111111000101010101000001110001101011010000100011110000000101111
1010101110101000111010001101101000101111001101100101000100100001101100000101100
1110101110011010001111011110000101001010000001101110000000010010010101011011011
1000111100111101101001101001111000100011010010110110011011100111011110100010010
1111100000110101001010111100011000111110110010111111100110010101001100110001100
1110111011001111101001000101001101101100011001111001011101111101110100101001110
1100001101111110101111010100111010011100010011011110110001001010010000011000110
000000100001000000001011

The period of binary sequence is 1116

The auto-correlation values of the above binary sequence are in the below graph:

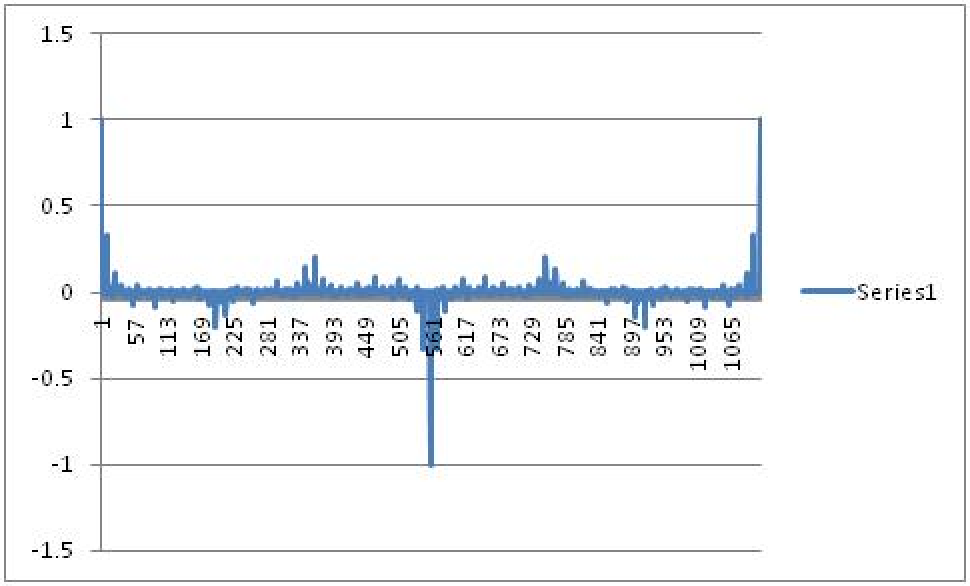

Figure 2.2. Autocorrelation for binary sequence generated using p=1117; period = 1116.

2) For p = 1541

The binary sequence equivalent to the above decimal sequence is:

0000000000101010100001110011100111111010010110100000101001001100110000000000010
1010100001110011100111111010010110100000101001001100110000000000010101010000111
0011100111111010010110100000101001001100110000000000010101010000111001110011111
1010010110100000101001001100110000000000010101010000111001110011111101001011010
0000101001001100110000000000010101010000111001110011111101001011010000010100100
1100110000000000010101010000111001110011111101001011010000010100100110011000000
0000101010100001110011100111111010010110100000101001001100110000000000010101010
0001110011100111111010010110100000101001001100110000000000010101010000111001110
0111111010010110100000101001001100110000000000010101010000111001110011111101001
0110100000101001001100110000000000010101010000111001110011111101001011010000010
1001001100110000000000010101010000111001110011111101001011010000010100100110011
0000000000101010100001110011100111111010010110100000101001001100110000000000010
1010100001110011100111111010010110100000101001001100110000000000010101010000111
0011100111111010010110100000101001001100110000000000010101010000111001110011111
1010010110100000101001001100110000000000010101010000111001110011111101001011010
0000101001001100110000000000010101010000111001110011111101001011010000010100100
1100110000000000010101010000111001110011111101001011010000010100100110011000000
0000101010100001110011100111111010010110100000101001001100110000000000010101010

0001110011100111111010010110100000101001001100110000000000101010100001110011100
01111110100101101000001010010011001100000000001010101000001

The length of decimal sequence is 1541

The period of the binary sequence is 1540

The auto-correlation values of the above binary sequence are in the below graph:

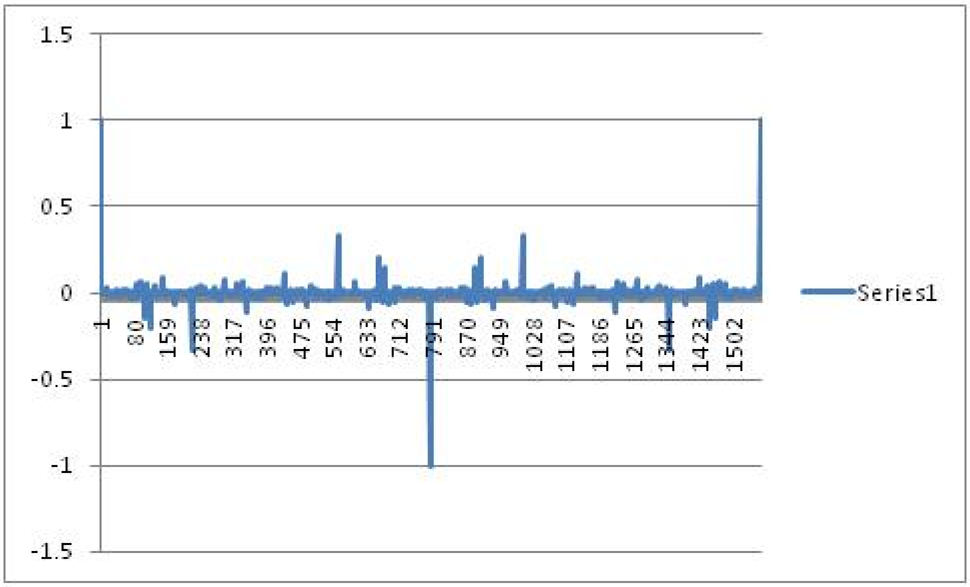

Figure 2.2. Autocorrelation for binary sequence generated using p=1541; period = 1540.

3) For p = 1861

The binary sequence equivalent to the above decimal sequence is:

0000000000100011001101110010100101100111110110111010111100011101010011001110010
0101110011011010011100100010110011011000001111101011101001000001110000000111111
1101001111111011000011000011111000101101011001010001101101011111111100001001011
1110111011110001010001111111000110110001100101110010110111001110110000000110111
0000011000110000101100100100011101000001100111011100100000100110100001000101010
1010010011001100001000111100010000000110000011010111101100011101110110011100
0100001100100001001001101110110111111010110001011100111111011010100101010110010
00000000110100110100101011111000011011110010011000011010101011111100110101100

0001011010001111010101101000011010001000101111000010111011000101010000101111110111101111110001001001001011101010001000001011110101001000011111101000111000111100110011010011110101111101010100010100110001011000100101101100010000101001010001001010010010000101101101010111000100110110010101100001110011100011001111111111011100110010001101011010011000001001000101000011100010101100110001101010001100100101100011011101001100100111110000010100010110111110001111110000000010110000000100111100111100000111010010100110101110010010100000000111101101000001000100001110101110000000111001001110011010001101001000110001001111110010001111100111001111010011011011100010111110011000100011011111011001011110111010101010110110011001111011100001110111111100111110010100001001110001000100110001111011110011011111011011001000100100000010100111010001100000010010101101010100110111111111001011001011010100000111100100001101100111100101010100000011001010011111010010111000010101001011110010111011101000011110100010011101010111101000000100001000000111011011011010001010111011111010000101011011110000000101110001110000110011001011000010100000010101011101011001110100111011010010011101111010110101110110101101101111010010010101000111011001001101010011110001100011100 11

The length of decimal sequence is 1861

The period of the binary sequence is 1860

The auto-correlation values of the above binary sequence are in the below graph:

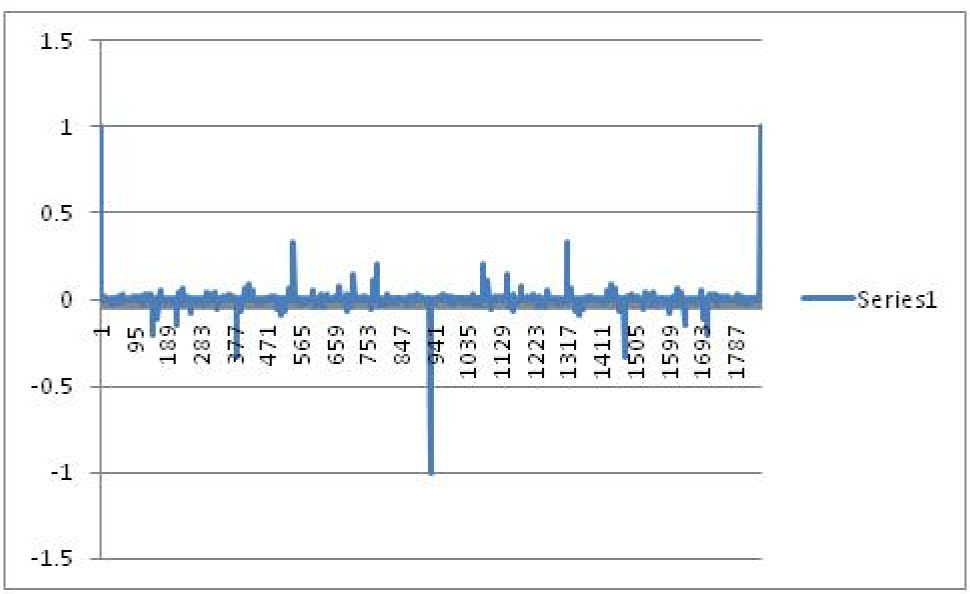

Figure 2.3. Autocorrelation for binary sequence generated using p=1861; period = 1860.

2.1.4 Randomness measured by Autocorrelation Function:

For simplicity, we consider only the autocorrelation function as measure of randomness. Since the randomness measure of discrete-time white noise sequence is 1 whereas that of a constant sequence is 0, this measure conforms to our intuitive expectations. The value of randomness for a binary shift register maximum-length sequence is close to 1, in accord with our expectations. The function for a maximum-length d-sequence is has a negative peak of -1 for half the period because of the anti-symmetry of the sequence.

Randomness may be quantified by measuring how much the autocorrelation departs from the ideal of white noise. Kak defines this randomness measure R(x) [26] by the expression below:

$$R(x) = 1 - \frac{\sum_{k=1}^{n-1} |C(k)|}{n-1}$$

I have calculated the randomness measure for several d-sequences which have the results as follows:

R(1117) = 0.986390678

R(1861) = 0.990382

R(2843) = 0.993393

This shows that d-sequences are quite random from the perspective of this measure.

2.2 Binary sequences from Windows PC

Different random number generators are used for PC applications. Here I have taken an online random number generator and generated random numbers to determine their autocorrelation function.

I downloaded the random number generator from www.cnet.com to my PC where it asks for the type of random sequence to be generated, such as decimal, binary etc, and also the length of the sequence.

The binary random sequence generated for a length of 2000 is shown below:

1000101101010001111001001100110101000011101111011010011110110100000001010111100
0111110010101000111100010011000111100010111101000001110101110001010101011101111
011010011000000101101000111001101010100000101110111100100110110000101100101010
0100010110000111000001011111001000101110010001000110011011100010010011011110010
0011011001011000101110011111100101101010000010100001101110100101100010100000111
1101010110110101101000110011100001111010110000101001001110100010101011011001000
0100101000110111100010001100110100100011011000101000001000000110000100101111000
0111111011110110100101010111110010101110000101001101101110000100010010000101000
1100001001010010011111100100001111101001000110101110111101000101101001010011010
0001001100000101100101101100101000101000101101001011000111010110111111011110010
0101101101010101110110001111101011101101110001101010101111001001011011000010010
1001101000110000110011100010111100011110011000100011100011111001101010001101110
1101110101100110101001111101011100110011100001100110001100101111110111110111010
0011000100000101001110000111111101100100100100111010111010101111100101111010110
1010010001101100100000111111111000010011100100000001001100010111000010001100111
0010101010101010001010000110000100001100110000100000100001000010000000001001010
0110110101001100100001100010000000110110001100100010001010100101001111111001100
1011111100110101001011001010100100100100110001110111100101111001000000111111010
1010010101111010001110010110101100100000100101111000100110010010111001101111111
0100101110101011001110101100001101100000011100000000000101100111010000111110101
1110111011011001001011011110011011111100010101001011100011001111011010001101100
0000100111001000101010000011001111010000000001001001111000000100110001001001010
0011101011110011110111101110101000011000010011011111110000111110000101100010100
0111101101101101011011001100110111101110111101010111001001101011000010101110100
1001010000101100011100000111010011000000000110110101001100101111110111110000100
111100100100000000000100101000100101000111111111001

The auto-correlation values of the above binary sequence are in the below graph:

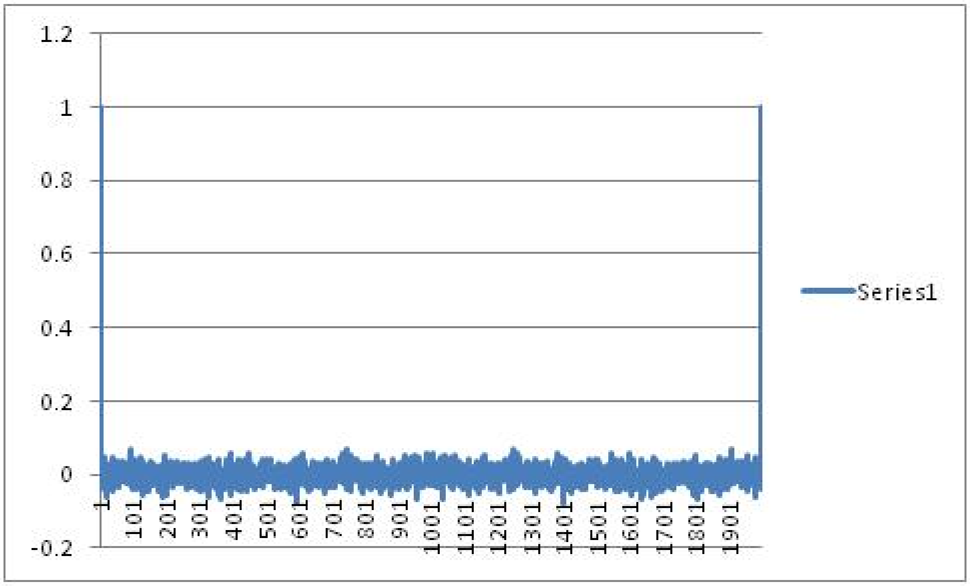

Figure 2.5. Autocorrelation for binary sequence generated from windows PC.

The randomness measure for this binary sequence is:

$$R(x) = 0.981007$$

This value is lower than the values found for d-sequences. But just this fact should not lead to the conclusion that d-sequences are superior owing to their obvious structure in the sense that the second half of the sequence is a complement of the first half.

CHAPTER III

MESH ARRAY SCRAMBLING OF NUMBERS

This chapter presents the use of mesh array for matrix multiplication [27]-[28] for generating random sequences.

3.1 Standard and Mesh Arrays:

In contrast to the standard array that requires 3n-2 steps to complete its computation, the mesh array requires only 2n-1 steps[27].

When we consider the problem of matrix multiplication, the standard array to compute the product of two 3×3 matrices is shown in Figure 3.1. It can be easily seen that the number of steps to solve this problem is (3n-2). The numbers inside the circle are the indices of the product matrix.

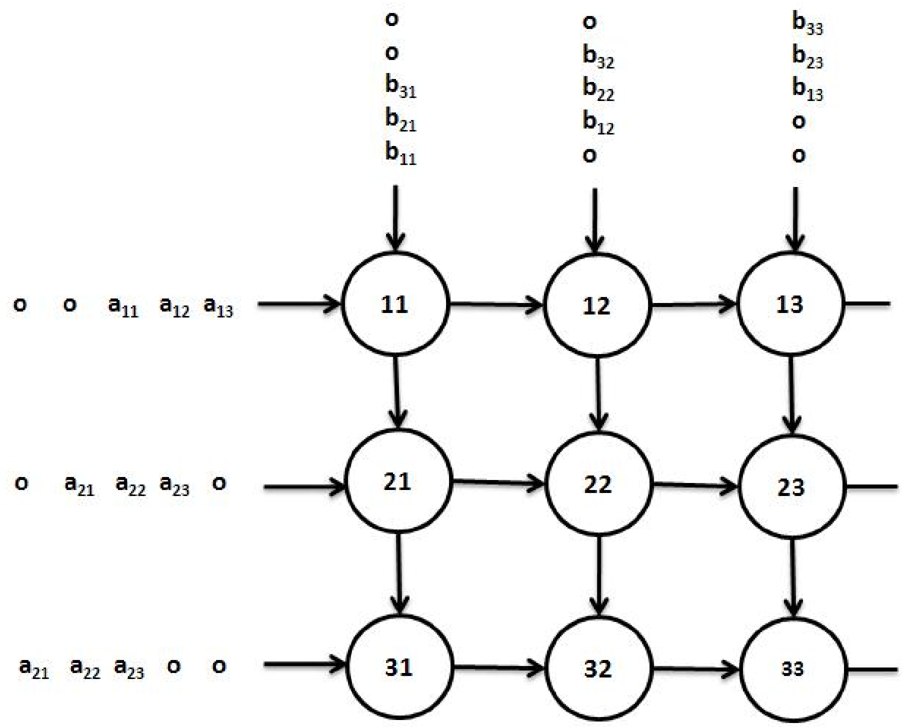

Figure 3.1: Standard Array matrix multiplication of two 3×3 matrices

The mesh array [27] of Figure 3.2 is more efficient than the standard array of Figure 3.1. The time taken to execute matrix multiplication on it is (2n-1).

3.2 Matrix Multiplication on a Mesh Array:

Matrix multiplication[27]-[28] is basic to many computational problems. In signal processing, the signal is usually transformed by a matrix. For an image the signal itself is a matrix, and for a one-dimensional signal, a large data set can be represented as a matrix.

Figure 3.2 presents the mesh array for multiplying two 4 × 4 matrices which takes 7 steps, whereas the standard array requires the same number of steps to multiply two 3 × 3 matrices. The speedup of the mesh array is a consequence of the fact that no zeros are padded in its inputs.

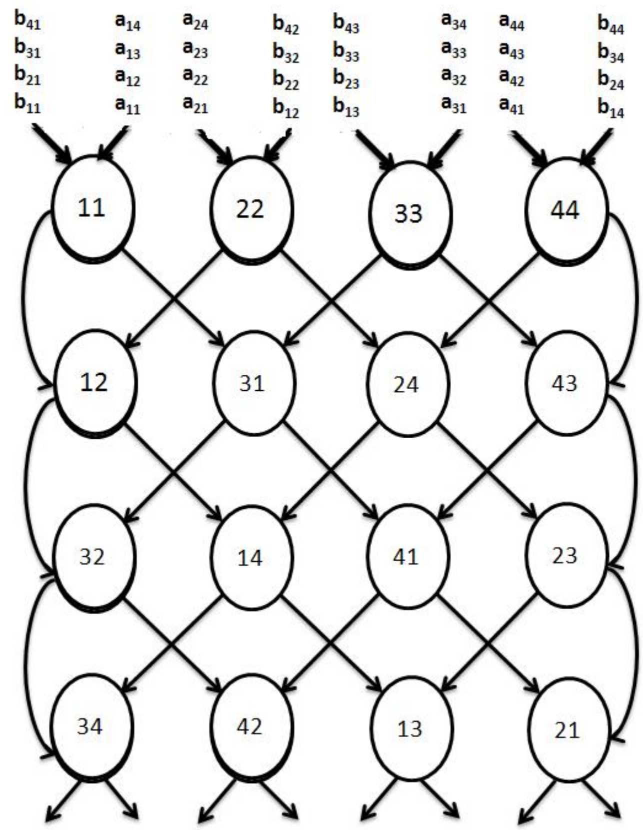

Figure 3.2: Mesh Array matrix multiplication of two 4×4 matrices C = AB

In the mesh array for the matrix multiplication the top layer has the diagonal terms 11, 22, 33, etc as goes from left to right. These numbers are written in the array as below.

11	22	33	44
12	31	24	43
32	14	41	23
34	42	13	21

We can see that the fourth row is the mirror reversed image of the second row and the third row has symmetry within itself.

For the 5×5 matrix, the product components are as follows:

11	22	33	44	55
12	31	24	53	45
32	14	51	25	43
34	52	15	41	23
54	35	42	13	21

3.3 Scrambling Transformations:

We have seen that the product matrix values do not appear in the standard arrangement[28]. The new arrangement when a matrix is multiplied with the identity matrix may be called scrambling transformation S. Given a total of $n \times n = n^2$ items, the total number of permutations is $n^2!$.

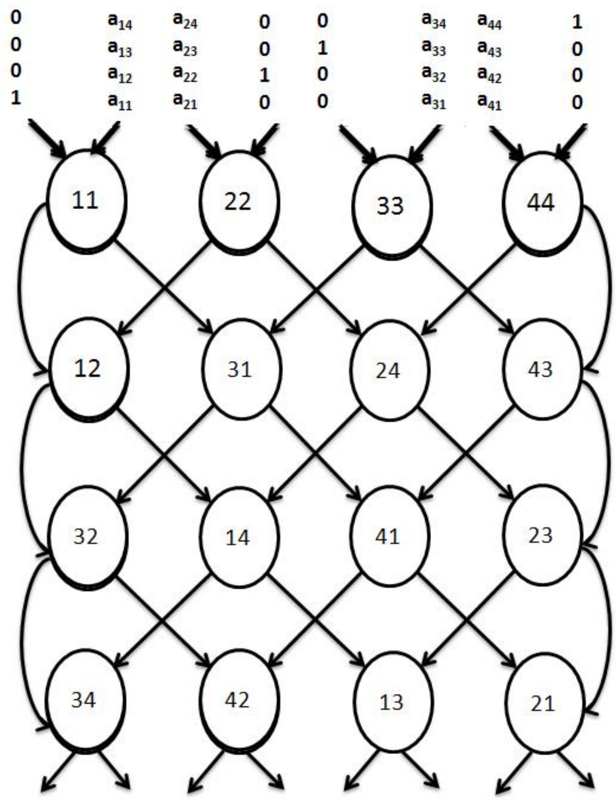

Figure 3.3: Mesh Array matrix multiplication of matrix A with Identity Matrix

Let the scrambling transformation is denoted by S, if the scrambling transformation is applied repeatedly, we obtain the original standard array in some number of steps. For the 4×4 matrix, the original array will be obtained in 7 steps[28].

$$\begin{matrix} 11 & 12 & 13 & 14 \\ 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \\ 41 & 42 & 43 & 44 \end{matrix} \quad \xrightarrow{S} \quad \begin{matrix} 11 & 22 & 33 & 44 \\ 12 & 31 & 24 & 43 \\ 32 & 14 & 41 & 23 \\ 34 & 42 & 13 & 21 \end{matrix} \quad \xrightarrow{S^2} \quad \begin{matrix} 11 & 31 & 41 & 21 \\ 22 & 32 & 43 & 13 \\ 14 & 44 & 34 & 24 \\ 23 & 42 & 33 & 12 \end{matrix} \quad \xrightarrow{S^3}$$

$$\begin{matrix} 11 & 32 & 34 & 12 \\ 31 & 14 & 43 & 13 \\ 44 & 21 & 23 & 43 \\ 24 & 42 & 41 & 22 \end{matrix} \quad \xrightarrow{S^4} \quad \begin{matrix} 11 & 14 & 23 & 22 \\ 12 & 31 & 24 & 42 \\ 21 & 12 & 24 & 13 \\ 43 & 42 & 34 & 31 \end{matrix} \quad \xrightarrow{S^5} \quad \begin{matrix} 11 & 44 & 24 & 31 \\ 14 & 21 & 41 & 34 \\ 12 & 22 & 43 & 33 \\ 13 & 42 & 23 & 32 \end{matrix} \quad \xrightarrow{S^6}$$

$$\begin{matrix} 11 & 21 & 43 & 32 \\ 44 & 12 & 34 & 23 \\ 22 & 31 & 13 & 41 \\ 33 & 42 & 24 & 14 \end{matrix} \quad \xrightarrow{S^7} \quad \begin{matrix} 11 & 12 & 13 & 14 \\ 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \\ 41 & 42 & 43 & 44 \end{matrix}$$

3.3.1 Generating Cycles:

As shown in [28], the items of both standard array and mesh array for the 4×4 case will be written in an array as follows:

$$\begin{pmatrix} 11 & 12 & 13 & 14 & 21 & 22 & 23 & 24 & 31 & 32 & 33 & 34 & 41 & 42 & 43 & 44 \\ 11 & 22 & 33 & 44 & 12 & 31 & 24 & 43 & 32 & 14 & 41 & 23 & 34 & 42 & 13 & 21 \end{pmatrix}$$

By using the above arrangement, we can write the standard and mesh arrays into cycles. The cycles generated from the above arrangement are as follows:

= (11) (42) (12 22 31 32 14 44 21) (13 33 41 34 23 24 43)

By writing the standard array and mesh array into cycles, we get the period of the scrambling transformations. The period is nothing but the maximum of lengths of the cycles since the lengths of the cycles are {1,1,7,7}, the period of the scrambling transformation = 7.

3.3.2 Prime Periods:

Some periods of the cycles associated with matrices of order n×n are prime. Over the range of n from 2 to 1000, the matrices associated with prime periods have the following distribution[29]:

- 101 – 200 ---- 15
- 201 – 300 ---- 10
- 301 – 400 ---- 11
- 401 – 500 ---- 5
- 501 – 600 ---- 3
- 601 – 700 ---- 8
- 701 – 800 ---- 5
- 801 – 900 ---- 4

- 901 – 1000---- 12

3.4 Generating Binary Sequence:

We can create a binary sequence of the periods in terms of 1s and 0s where the even period is represented as 1 and odd period is represented as 0[29]. The binary sequence for the periods of orders 2 to 1000 is as follows:

1110110110000110000001000000000000000100000000100000000000000010000010000000000
0000000000000101000000110000000001000100000000101011000001000000000000000000000
1010000000100000000000000000100000010000000010000011000000000100000000100001000 10
0000000000000000000000001000100000000000000000000000001000000100000000000010000
0000100000000000011000000000000001000000000001000000000000000000000000000000000
0011011100000000000000000000000101000000000000000000010000000000000000001000000
0000100000000000000000000000000000000000000010000000000000000000000000000000000
0000000000000000000000000000000000000000100010000000000000000000010000001100100000
0000000000000000000000000001000000000000000000000000110000100000000000000000000
0000000000000100000000000000000000000000000110000000000000000100000000000000000
0000000000000001000000000000000000000000010000001000000000001000000000010000000
0000000000000000000000000000000000000000100000000100100000000000100000000000000
0000000001100000000000001000000101000000000100000000000000000001000

The autocorrelation function for k ranging from 0 to 1000 is shown in Figure 3.4.

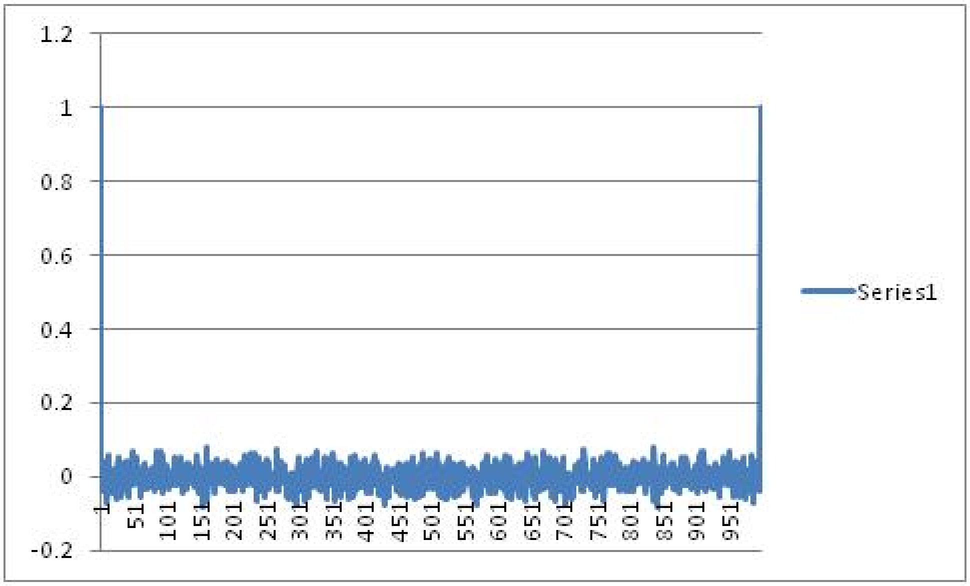

Figure 3.4: Autocorrelation Function of periods of mesh array

The randomness measure for random sequence generated from the Mesh Array is:

$$R(x) = 0.972194$$

This is quite close to but not as high as the values obtained for d-sequences.

CHAPTER IV

CRYPTOGRAPHIC HARDENING OF PSEUDORANDOM SEQUENCES

In this chapter, we investigate results of a method of cryptographic strengthening of RNGs. Basically the idea is to apply a many–to-one mapping to the binary output of the RNG, increasing the complexity of the reverse process. We wish to consider the use of a 3-to-1 and higher mappings where each group of 0s and 1s is replaced by whatever is the majority to see if it improves the autocorrelation function of the resultant sequence. This will be tried both for the Windows based RNGs as well as d-sequences

4.1 The PR(n) Sequence:

The PR(n) sequences[32] emerges by mapping each group of adjacent n bits (n odd) of the PR sequence to 0 or 1 depending on whether it has a majority of 0s or 1s. We have done experiments on many d-sequences and we find that PR(3) provides significant improvement and that there is no significant advantage in taking larger values of n.

Let us take some examples for d-sequences, then the PR(n) sequences will be as follows:

P=1571:

Original Sequence or PR(1):

0000000000101001101101110101001010011110000100001001111100001010111010101111110
1010011111010111100101100110011111110110110111111110010111101101011011000010111

0100110101100111001001100100101101001000110101110001100101000001111111111000001
0110110100000100000100101110011100010001011011111001111110001000000010000111100
1001111001100100000011011011000000100111000110111101110101110100001011111100101
0100011010001111000100101011110101011010000111010000000101110111011100011111001
1110001110100101011001011101100010010000101100111110011010010100010010011010001
1110101101110111110001010101011000110011110100011011100010100000010110001010001
0110001111100011111110001101010001111101110011001101010100100100101010100000111
1110011100111011001001101111001000100110001000011001000110000100011110110011100
0101011111111110101100100100010101101011000011110111101100000111101010001010100
0000101011000001010000110100110011000000010010010000000011010000100101001001111
0100010110010100110001101100110110100101101110010100011100110101111110000000001
1111010010010111110111110110100011000111011101001000001100000011101111111011110
0001101100001100110111111001001001111110110001110010000100010100010111101000000
1101010111001011100001110110101000010101001011110001011111110100010001000111000
0011000011100010110101001101000100111011011110100110000011001011010111011011000
1011000010100100010000011101010101001110011000010111001000111010111111010011100
1011010011100000111000000011100101011100000100011001100101010110110110101010111
1100000011000110001001101100100001101110110011101111001101110011110111000010011
1001110101

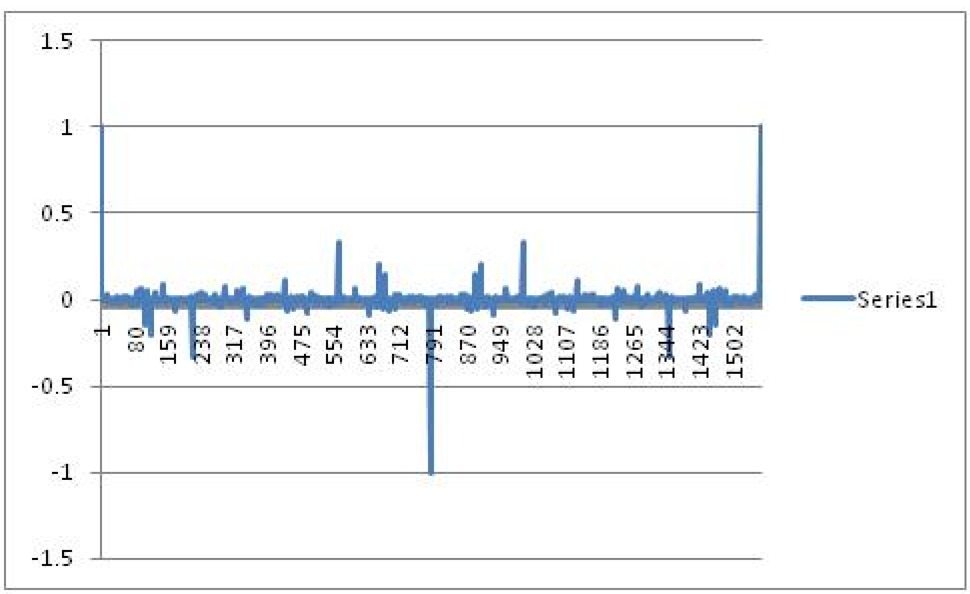

Figure 4.1: Autocorrelation function for PR(1) for 1/1571

3-numbers or PR(3):

0000011100101000011011101110111101101111111011111001010110010010010101000111001100000010100111011000001001010001100010111101001101001010001101101100011110101010010111000011101000001010111101010101001100001010101101101101110011000010011011100110001000000100011000111101000011001111011101000100101010000000001000001001001011110011000110110001100111110001110001001111101101111000111010000001100111011011000001101110000100001010010011110100011111010000000110010001001111101110101000101100001011110111000100110011101101101100000100

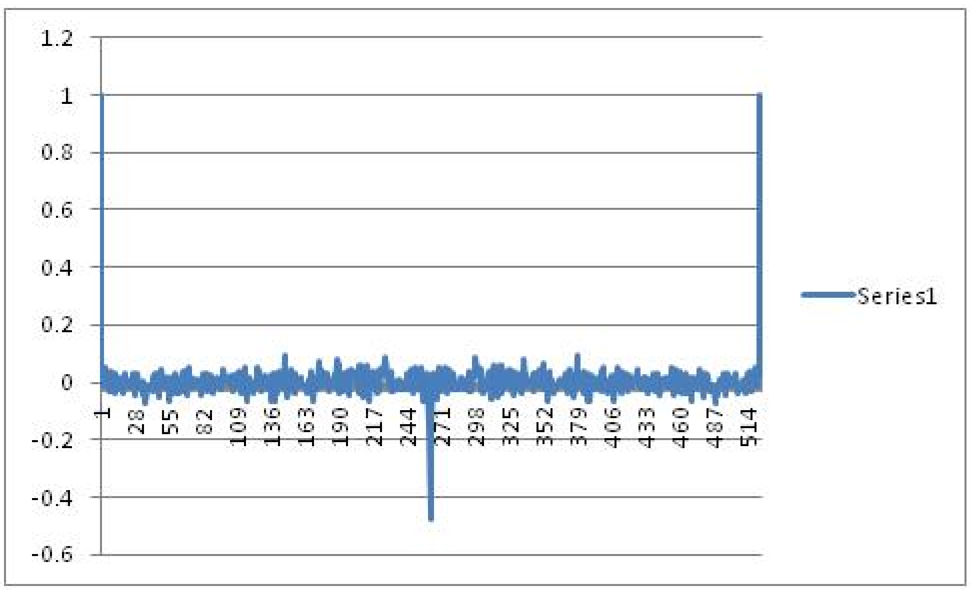

Figure 4.2: Autocorrelation function for PR(3) for 1/1571

The autocorrelation function in Figure 4.2 is better than that in Figure $.1 because the negative peak for half the sequence has been reduced.

5 numbers or PR(5):

0001100100101111111011111111101011100101100011010001100111000101000100011110100
1100101000111110101100011100001111000110100100010100111010010111011000000011001
1100110110100000000100000000010100011010011100101110011000111010111011100001011
0011010111000001010011100011110000111001011011101011000101101000100111111100
11

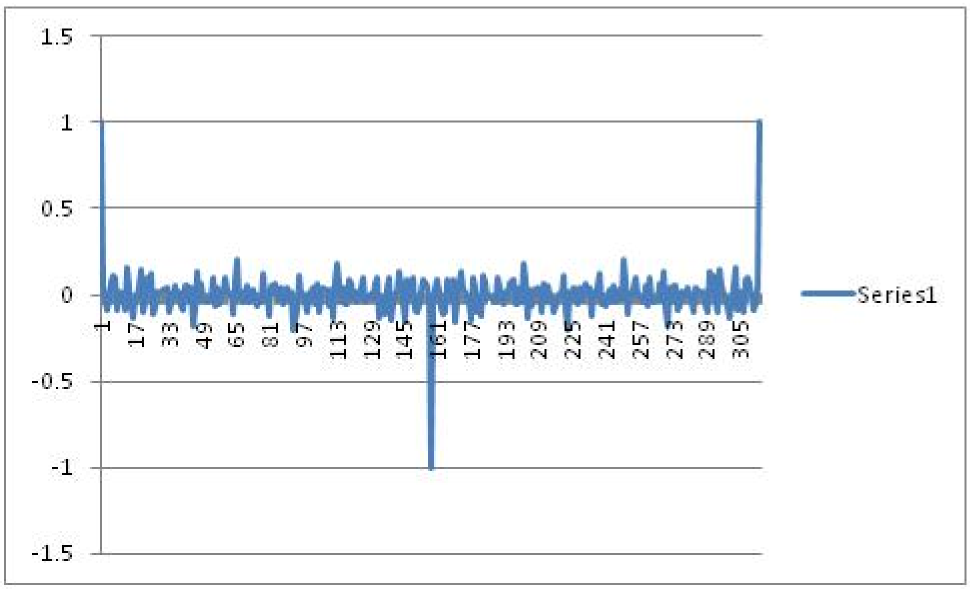

Figure 4.3: Autocorrelation function for PR(5) for 1/1571

The autocorrelation function in Figure 4.3 is unchanged for Figure 4.1 as 5 divides 1570.

7 numbers or PR(7):

0011010101101111111110111000100110000101110001000011101001011000111101100110011
1111010000000110010100011110000011011001010100100000000010011010010011110100011
001101000101101001010100010110011100011010111100010001110000011111010

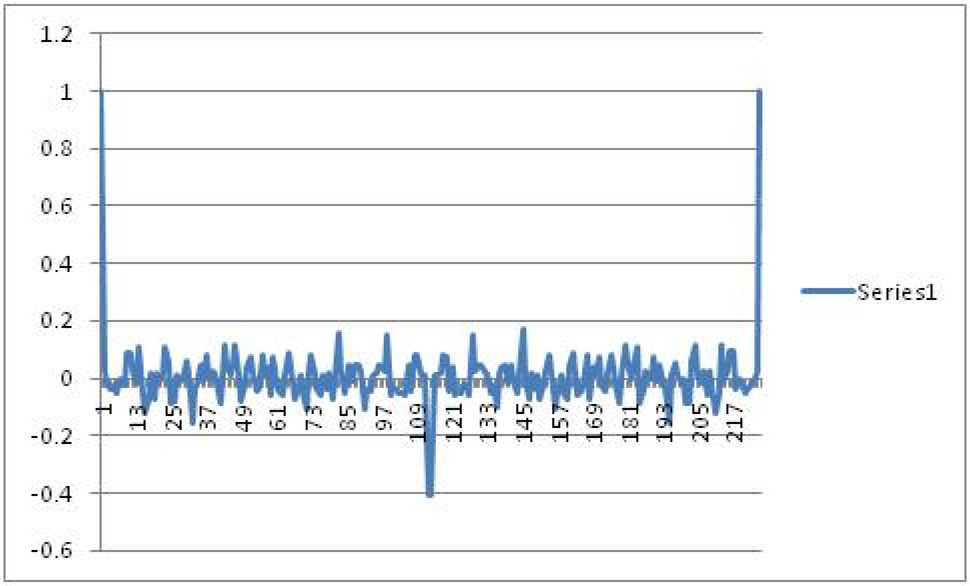

Figure 4.4: Autocorrelation function for PR(7) for 1/1571

9 numbers or PR(9):

0010000111111111011000001010100100110101101010100011101001001111011000110100001110000101101110000000000010001101100110100100101000011100110000011011000110011
10010011100111111010

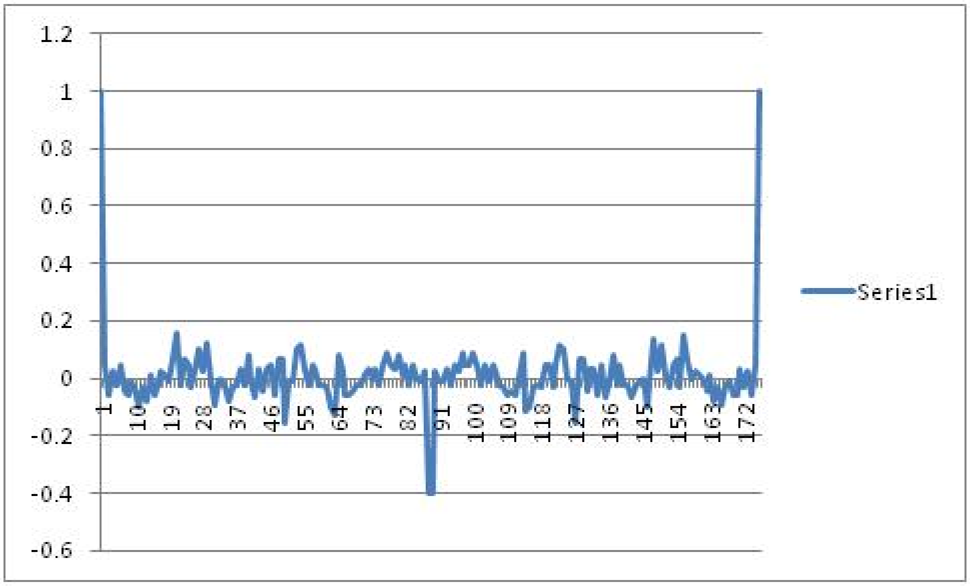

Figure 4.5: Autocorrelation function for PR(9) for 1/1571

11 numbers or PR(11):

0110001111111111001011010110101011101010011110110111011000111001101000011001000010000000110101111001101100101100100010101100101110010011000111 00

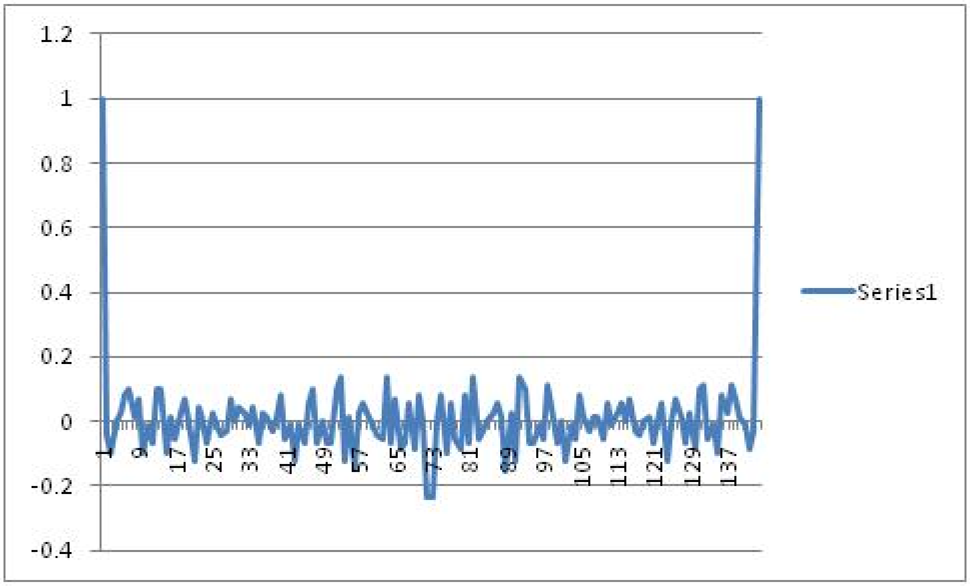

Figure 4.6: Autocorrelation function for PR(11) for 1/1571

From the above graphs, we can say that the negative peak for half the period gets smaller and smaller as we increase n in PR(n) and the improvement in randomness for PR(11) is quite impressive.

The tables below provide the list of the largest values of the autocorrelation function for the given sequences. The off-1 or -1 values are 0.33 and -0.33 for PR(1). We see that these values have reduced to 0.11 and -0.13 for the d-sequence corresponding to 1907[32]. There is corresponding improvement (not necessarily of the same extent) for the other examples given below.

1541:

Table 4.1: Reduction in the largest value Autocorrelation function values for d-sequence 1541

PR(1)	PR(3)	PR(5)	PR(7)	PR(9)	PR(11)
1.0	1.0	1.0	1.0	1.0	1.0
-1.0	-0.26	-0.28	-0.19	-0.22	-0.31
0.33	0.94	0.74	0.65	0.86	-0.94
-0.33	-0.24	-0.26	-0.15	-0.17	-0.27
0.19	0.71	0.47	0.51	0.74	-0.88
-0.19	-0.21	-0.21	-0.14	-0.13	-0.24
0.14	0.57	0.31	0.42	0.49	-0.77
-0.14	-0.19	-0.19	-0.12	-0.08	-0.21

1907:

Table 4.2: Reduction in the largest value Autocorrelation function values for d-sequence 1907

PR(1)	PR(3)	PR(5)	PR(7)	PR(9)	PR(11)
1.0	1.0	1.0	1.0	1.0	1.0
-1.0	-0.50	-0.37	-0.46	-0.81	-0.38
0.33	0.11	0.11	0.18	0.19	0.26
-0.33	-0.13	-0.12	-0.13	-0.21	-0.17
0.20	0.08	0.10	0.12	0.18	0.18
-0.20	-0.10	-0.11	-0.12	-0.20	-0.16
0.14	0.07	0.09	0.10	0.17	0.17
-0.14	-0.07	-0.10	-0.10	-0.18	-0.13

2243:

Table 4.3: Reduction in the largest value Autocorrelation function values for d-sequence 2243

PR(1)	PR(3)	PR(5)	PR(7)	PR(9)	PR(11)
1.0	1.0	1.0	1.0	1.0	1.0
-1.0	-0.49	-0.38	-0.42	-0.39	-0.80
0.33	0.19	0.12	0.11	0.12	0.2
-0.33	-0.12	-0.09	-0.19	-0.21	-0.2
0.19	0.09	0.11	0.10	0.11	0.19
-0.19	-0.08	-0.07	-0.11	-0.20	-0.19
0.14	0.08	0.10	0.09	0.10	0.14

-0.14	-0.07	-0.06	-0.09	-0.19	-0.14

2333:

Table 4.4: Reduction in the largest value Autocorrelation function values for d-sequence 2333

PR(1)	PR(3)	PR(5)	PR(7)	PR(9)	PR(11)
1.0	1.0	1.0	1.0	1.0	1.0
-1.0	-0.52	-0.40	-0.39	-0.35	-1.0
0.33	0.09	1.13	0.12	0.15	0.21
-0.33	-0.10	-0.11	-0.17	-0.21	-0.21
0.19	0.06	0.08	0.11	0.14	0.19
-0.19	-0.09	-0.09	-0.14	-0.18	-0.19
0.14	0.05	0.07	0.10	0.12	0.18
-0.14	-0.08	-0.08	-0.13	-0.15	-0.18

2843:

Table 4.5: Reduction in the largest value Autocorrelation function values for d-sequence 2843

PR(1)	PR(3)	PR(5)	PR(7)	PR(9)	PR(11)
1.0	1.0	1.0	1.0	1.0	1.0
-1.0	-0.44	-0.4	-1.0	-0.74	-0.4
0.33	0.17	0.15	0.17	0.14	0.10
-0.33	-0.16	-0.10	-0.17	-0.12	-0.19
0.19	0.07	0.10	0.16	0.11	0.09
-0.19	-0.10	-0.08	-0.16	-0.10	-0.18
0.14	0.06	0.09	0.15	0.10	0.08
-0.14	-0.07	-0.07	-0.15	-0.09	-0.13

It should also be noted that the performance of PR(n) for a larger value of n does not always imply improved results as far as the autocorrelation function is concerned.

4.2 PR(n) sequences from Windows PC:

For binary sequences generated by random number generators in windows PC the performance of many-to-one mapping on the quality of the autocorrelation function is given in the table below:

The results for the windows PC RNG are shown in the table below:

PR(1)	PR(3)	PR(5)	PR(7)	PR(9)	PR(11)
1.0	1.0	1.0	1.0	1.0	1.0
-0.07	-0.12	-0.15	-0.19	-0.20	-0.16
0.06	0.10	0.14	0.18	0.16	0.19
-0.06	-0.10	-0.14	-0.15	-0.14	-0.15
0.05	0.09	0.12	0.16	0.14	0.18
-0.05	-0.09	-0.13	-0.14	-0.13	-0.12
0.04	0.08	0.11	0.15	0.13	0.13
-0.04	-0.08	-0.10	-0.13	-0.12	-0.11

Table 4.6: The largest value autocorrelation values for Windows RNG

We see that the use of applying the many-to-one mapping does not improve the autocorrelation function of this RNG.

4.3 Mesh PR(n) Sequence:

The results of autocorrelation function for PR(n) sequence are given in the table below:

Table 4.7: The largest value autocorrelation values for the mesh array sequence

PR(1)	PR(3)	PR(5)	PR(7)	PR(9)	PR(11)
1.0	1.0	1.0	1.0	1.0	1.0
-0.08	-0.14	-0.22	-0.18	-0.19	-0.23
0.08	0.12	0.22	0.19	0.19	0.17
-0.07	-0.13	-0.16	-0.16	-0.17	-0.15
0.07	0.11	0.18	0.18	0.18	0.15
-0.06	-0.11	-0.15	-0.15	-0.16	-0.13
0.06	0.10	0.14	0.15	0.15	0.13
-0.05	-0.10	-0.12	-0.13	-0.14	-0.10

The performance of the mesh array sequence to the many-to-one mapping is similar to that for the windows RNG.

4.4 Nested PR(n) Sequences:

Nested PR(n) sequences are nothing but taking the PR(n) sequences for the PR(n) sequences. Below table shows the nested PR(n) sequences for Windows PC. It is significant that the performance of nested PR(n) sequences for the windows PRNG is not very good as given by the results in the table below:

Table 4.8: Nested values for windows RNG

PR(1)	PR(3)	PR3(3)	PR3(PR3(3))
1.0	1.0	1.0	1.0
-0.07	-0.12	-0.12	-0.21
0.06	0.10	0.15	0.18
-0.06	-0.10	-0.10	-0.18
0.05	0.09	0.14	0.16
-0.05	-0.09	-0.09	-0.13
0.04	0.08	0.13	0.13
-0.04	-0.08	-0.08	-0.10

The randomness measure for the random sequences and PR(n) sequences are calculated as follows:

Table 4.9: PR(n) values for all the sequences

	P=1117	P=1861	P=2843	Win PC	Mesh Array
PR(1)	0.986390678	0.990382	0.993393	0.981007	0.972194
PR(3)	0.951072	0.964047	0.977966	0.966357	0.950941
PR(5)	0.943512	0.945826	0.968162	0.954581	0.940452
PR(7)	0.932415	0.943862	0.949445	0.947845	0.923323
PR(9)	0.896452	0.93822	0.949137	0.935778	0.896104
PR(11)	0.892918	0.935363	0.943303	0.930733	0.895849

From the above examples, we can see that the majority of d-sequences have the highest value of randomness measure for the PR(3) sequence.

For Windows PC, the randomness measure value is high for the PR(11) sequence and is very low for PR(9) sequence. For Mesh Array, the randomness measure value is high for the PR(7) sequence and is very low for PR(11) sequence. As mentioned before, for Windows PC and Mesh Array, the use of applying the many-to-one mapping does not improve the randomness measure value.

CHAPTER V

CONCLUSION

From the analysis of the above random sequences, we conclude that d-sequences have an excellent randomness measure. However, they are not cryptographically strong when compared to the random sequences generated from Windows PC and the mesh array random sequences because of the linear structure behind their generation. We have used many-to-one mappings to improve the cryptographic strength of d-sequences.

For the autocorrelation function of d-sequences, we can see that in many cases the negative peak for half period gets progressively smaller as we increase n in PR(n). The best results are obtained when n=3. However, the use of applying the many-to-one mapping does not improve the autocorrelation function for random sequences from Windows PC and Mesh random sequences.

REFERENCES

[1] D. Knuth, The Art of Computer Programming, Vol. 2, 2nd ed., Addison-Wesley, Reading, (1981).

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

[3] S. Kak, Classification of random binary sequences using Walsh-Fourier analysis. Proceedings of Applications of Walsh Functions. Pp. 74-77. Washington, 1971.

[4] G. Anthes, The quest for randomness. Communications of the ACM, 54: 13-15, 2011.

[5] S. Srinivasan, Rapid Abstract Control Model for Signal Processing Implementation. et al. 2010 IEEE Symposium on VLSI Circuits, June 16-18, 2010.

[6] M.A.Nielsen and I.L. Chuang, Quantum Computation and Quantum Information. Cambridge University Press, 2000.

[7] R. Landauer, The physical nature of information. Phys. Lett. A 217, 188-193, 1996.

[8] S. Kak, Information, physics and computation. Foundations of Physics 26, 127-137, 1996.

[9] S. Kak, The initialization problem in quantum computing. Foundations of Physics 29, pp. 267-279, 1999.

[10] S. Kak, Quantum information and entropy. International Journal of Theoretical Physics 46, 860-876, 2007.

[11] T. Ritter, Randomness tests: a literature survey, http://www.ciphersbyritter.com/RES/RANDTEST.HTM

[12] A. Kolmogorov, Three approaches to the quantitative definition of information. Problems of Information Transmission. 1, 1-17, 1965.

[13] G.H. Hardy and E.M. Wright, An Introduction to the Theory of Numbers. Oxford University Press, 1954.

[14] T. E. Hull and A. R. Dobell, Random number generators. SIAM Review. Vol. 4, 230-254, 1962.

[15] P. Harry, Computer Simulation Techniques. Addison-Wesley, 25-46, 2008.

[16] P.D. Coddington, Analysis of random number generators using Monte Carlo simulation, Northeast Parallel Architecture Center. Paper 14, 1994.

[17] S. Martin and F. Viktor, Random Numbers in Cryptography. http://www.ipam.ucla.edu/publications/scws4/scws4_6568.pdf, 2006.

[18] Random.org, Games and Gambling. http://www.random.org/testimonials/games

[19] M.E.J. Newman, Random graphs with arbitrary degree distributions. arXiv: 64.026118, 2007.

[20] M. George , Some difficult-to-pass tests of randomness. Journal of Statistical Software, Vol. 7, 2002.

[21] A.Parakh, A d-sequence based recursive random number generator. arXiv:cs/0603029v2, 2006.

[22] S. Kak and A. Chatterjee, On decimal sequences. IEEE Transactions on Information Theory, IT-27: 647 – 652, 1981.

[23] S. Kak, Encryption and error-correction coding using D sequences. IEEE Transactions on Computers, C-34: 803-809, 1985.

[24] S. Kak, New results on d-sequences. Electronics Letters, 23: 617, 1987.

[25] S.B. Thippireddy, Binary random sequences obtained from decimal sequences, arXiv: 0809.0676, 2009.

[26] S.K.R. Gangasani, Testing d-sequences for their randomness. arXiv:0710.3779

[27] S. Kak, A two-layered mesh array for matrix multiplication. Parallel Computing 6, 383-385, 1988.

[28] S. Kak, On the mesh array for matrix multiplication. 2010. arXiv:1010.5421.

[29] S. Rangineni, New results on scrambling using the mesh array. arXiv: 1102.4579

[30] S.W. Golomb, Shift Register Sequences. Aegean Park Press, 1982.

[31] S. Rangineni, Cryptographic Hardening of d-Sequences. arXiv: 1106.3574

VITA

Sandhya Rangineni

Candidate for the Degree of

Master of Science

Thesis: CRYPTOGRAPHIC ANALYSIS OF RANDOM SEQUENCES

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the Master of Science in Computer Science at Oklahoma State University, Stillwater, Oklahoma in July, 2011.

Completed the requirements for the Bachelor of technology in Computer Science at Jawaharlal Nehru Technological University, Anantapur, Andhra Pradesh, India in 2009

Experience:

Web Developer at College of Arts and Science Technical Services, Oklahoma State University, Stillwater, OK

Name: Sandhya Rangineni Date of Degree: July, 2011

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: CRYPTOGRAPHIC ANALYSIS OF RANDOM SEQUENCES

Pages in Study: 47 Candidate for the Degree of Master of Science

Major Field: Computer Science

Scope and Method of Study:

Cryptographically strong random sequences are essential in cryptography, digital signatures, challenge-response systems, and in Monte Carlo simulation. This thesis examines techniques for cryptographic hardening of random sequences that are not cryptographically strong. Specific random sequences that are considered include d-sequences, that is sequences that are reciprocals of primes, and a new sequence obtained by the use of a specific two-dimensional mesh array.

Findings and Conclusions:

It is shown that the use of many-to-one mapping on blocks of the raw sequence improves the quality of autocorrelation function. Various types of many-to-one mappings are used and their effect on the autocorrelation function is compared. Sequences are also compared using another measure of randomness.

ADVISER'S APPROVAL: Dr. Subhash Kak