Project Report On

An Evolutionary Approach for Generation of Optimized Seed for Non-linear Equation

"A report submitted to the Department of Computer Science & Engineering, University of Kalyani in partial fulfillment of the requirements for the two years fulltime postgraduate degree of Master of Technology in Computer Science & Engineering"

Submitted by

RAMEN PAL
ROLL: 90/CSE No: 160014
Registration No: 100197 of 2016-2017

Under the guidance of
Dr. Jyotsna Kumar Mandal
Professor
Dept. of Computer Science & Engineering
University of Kalyani

Department of Computer Science and Engineering
University of Kalyani
Kalyani, Nadia, West Bengal
June,2018

Project Report On

An Evolutionary Approach for Generation of Optimized Seed for Non-linear Equation

Submitted by

RAMEN PAL
ROLL: 90/CSE No: 160014
Registration No: 100197 of 2016-2017

Under the guidance of
Dr. Jyotsna Kumar Mandal
Professor

Dept. of Computer Science & Engineering
University of Kalyani
Kalyani, Nadia, West Bengal

UNIVERSITY OF KALYANI

FACULTY OF ENGINEERING TECHNOLOGY AND MANAGEMEMNT

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Dr. Jyotsna Kumar Mandal
Professor
Dept. of Comp. Sc. & Engg.

Kalyani, Nadia-741235
West Bengal, India
Phone: (033) 25809617 (O)
Mobile: +91 - 9434352214
E-mail: *jkm.cse@gmail.com*

DATE: 03-06-2018

CERTIFICATE OF ORGINALITY

This is to certify that the thesis entitled **"An Evolutionary Approach for Generation of Optimized Seed for Non-linear Equation",** is submitted to the **University of Kalyani** in fulfillment of the requirement for the award of the degree of Master of Technology in Computer Science and Engineering Part II 2nd Semester, is an original work carried out by **Ramen Pal** (Roll. **90/CSE**, No.**160014**, Reg. **100197 of 2016-2017**).

The matter embodied in this project is a genuine work done by the student and has not been submitted whether to this University or to any other University/Institute for the fulfillment of the requirement of any course of study.

Signature of the HOD
Dr. Utpal Biswas
Dept. of Comp. Sc. & Engg.
University of Kalyani
Kalyani, Nadia, India

Signature of the Supervisor
Dr. Jyotsna Kumar Mandal
Dept. of Computer Sc. & Engg.,
University of Kalyani
Kalyani, Nadia, India

External Examiner(s)

Abstract

Chaotic map gained its importance in the field of cryptography, due to its properties like, randomness, unpredictability, sensitivity on initial condition, aperiodicity, which is used to generate pseudorandom bit streams. In this project the optimal values of chaos parameters are generated through Real Coded Genetic Algorithm (RCGA), which is optimal, unpredictable, and optimally sensitive. Here, a non-deterministic RCGA based optimal pseudo-random bit sequence generator based on Chaotic maps, such as Logistic Chaotic map, Skew Tent Chaotic map, Cross Coupled Logistic Chaotic map, Cross Coupled Skew Tent Chaotic map is proposed. A real coded crossover and mutation technique is proposed for RCGA. Seed values for chaotic map has been optimized by using all sub-functions of GA (RCGA). These seed values are used to generate optimal pseudorandom bit stream of finite length. The randomness of the bit stream is tested by NIST statistical test suit.

Declaration

I hereby declare that the project entitled *An Evolutionary Approach for Generation of Optimized Seed for Non-linear Equation* submitted for the M.Tech in Computer Science & Engineering degree, is my original work and the project has not formed the basis for the award of any degree, associate-ship, fellowship or any other similar titles.

Ramen Pal
Dept. of Computer Science
& Engineering
University of Kalyani
June, 2018

Acknowledgements

I am highly indebted to *UNIVERSITY OF KALYANI* and all our teachers without whose moral help and extensive cooperation I would not be able to submit this report.

I would like to express my deep sense of gratitude and profound thanks to the kind acceptance, dynamic encouragement, support and generosity of my project supervisor and mentor *Dr. Jyotsna Kumar Mandal, Professor, Dept. of Computer Science & Engineering, Faculty of Engineering Technology and Management, University of Kalyani*, for his valuable advice, guidance and unending support. During the most difficult time of my career he constantly motivates me and kept faith in me. He was always available with new ideas and suggestions during the difficult phases of the project.

My gratitude goes to *Dr. Somnath Mukhopadhyay, Assistant professor, Dept. of Computer Science & Engineering, Assam University, Silchar, India* for his demand for clarity of expression, useful subject discussions, constant care and kind patience throughout the project duration which greatly helped me to complete this project.

Finally, yet important my sincere thanks goes to all of my friends and respected seniors who have directly and indirectly extended their valuable guidance and advice during the preparation of this report, which will give me the continuous flow of inspiration to complete this report.

Ramen Pal
Dept. of Computer Science
& Engineering
University of Kalyani
June, 2018

Dedicated to my beloved parents, for their love, endless support, encouragement and sacrifices.

Ramen Pal

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The tool is used to generate a random number sequence is called Random Number Generator or RNG. Random Number Generators (RNG's) can be divided in two classes, viz, Pseudo Random Number Generator (PRNG) and Truly Random Number Generator (TRNG)[2]. PRNG algorithm is deterministic in nature[7], the input to this type of algorithm is called seed and the output of this type of algorithm is a pseudorandom bit sequence[1]. The deterministic nature of the aforementioned algorithm is useful to generate same bit sequence for a same seed multiple times. A TRNG algorithm is non deterministic in nature, it uses hardware based architecture or design for the generation of random bit sequences, some TRNG applies some bit level operation on that generated bit sequence to get the output random bit sequence[11].

Chaos and random signals share the property of long term unpredictable irregular behavior[10, 22, 15]. Chaotic maps are used as a tool to use this aforementioned phenomenon of chaos in the field of random number sequence generation. Chaotic maps are evolution functions. In the domain of mathematics a chaotic map can be used to produce chaotic and unpredictable number sequence, for its deterministic simplicity and chaotic behavior. In last two decades it is seen that Chaotic map can be used to design deterministic random numbers generators or PRNG[46, 25]. In 2013, H. Hu et al proposed a Pseudorandom bit sequence generator(PRBG) based on Chen chaotic system[23]. In 2008 Liu et al used a combination of henon map and improved 2D Logistic map in their proposed method[26]. In 2012 Long Bao et al proposed a chaotic system using a combination of three conventional one-dimensional chaotic maps. This system chooses the Sine map or the

Tent map to generate the chaotic sequence according to the output value of the Logistic map. The system shows better chaotic performance by inheriting the high sensitivity to the initial conditions and expanding the range of parameters. They argues that this combination will make the generation of the chaotic sequences more complicated and difficult to be identified by iterates and auto-correlation functions[6]. In 2005 Bergamo et al proposed a cryptosystem, that can be implemented by using any chaotic map, i.e. it enjoys the semi-group property. They had shown that a well designed algorithm can adopt any type of chaotic map to serve a purpose[8]. In 2011 Pawel Dabal et al proposed a logistic chaotic map based PRNG. They had proposed a hardware based design, which is simulated in Virtex 5FXT FPGA device (Xilinx)[16]. In 2015 Martinez and Canton proposed a PRBG based on K-model maps[20]. In 2003 Kocarev et al proposed a PRBG for which security does not rely on a number-theoretical problem, and therefore, does not use modular multiplications. In contrast, its security relies on the large numbers of branches the inverse of a function used in the algorithm has. The generator uses only binary operations[24]. In 2015 Sadkhan and Mohammad proposed a PRBG based on Unified chaotic map[40]. In 2014 A. Akhshani et al proposed a PRBG based on Quantum chaotic map, and shows that the output chaotic sequence is non-periodic[2]. In 2014 M. Francois et al proposed a Secure PRNG three-mix, from where its seen that rather than using a single chaotic map, mixing of several chaotic maps for designing a PRNG can be done[31]. In 2016, L. Liu et al proposed a PRNG based on new multi-delayed Chebyshev map[28]. In Stojanovski et al proposed a PRNG based on a 2-regions PL1D chaotic map, which overcomes the drawbacks of classical PRNGs: reliance on the assumed randomness of a physical process, inability to analyze and optimize the PRNG, inability to compute probabilities and entropy of the PRNG, and inconclusiveness of statistical tests. They exploit the deterministic part of the nature of chaos and the simplicity of the 2-regions PL1D map to extend existing works on chaos-based PRNGs, and to give mathematical analysis of the information generation mechanism[44] and they also extend their works to prove their argument, where they addressed the practical issues of their chaos-based PRNG[45]. From the aforementioned PRNGs, it has been observed that these PRNGs generates random bit sequences by using chaotic map. The main input to those algorithms are the optimal seed values of the chaotic map. But these algorithm does not perform the optimization of the seed values, so it is the responsibility of the user to input optimal seed values for the chaotic map and the optimality should

be check mathematically. This task is difficult to perform for each execution to get a random bit sequence. So, they must rely on a set of optimal seed values. From this discussion it can be included that an predefined optimal seed value or a set of optimal seed values will not only degrade the security, but also it does not able to generate an unique bit sequence in each and every execution of a PRNG.

Genetic algorithms are non deterministic in nature and it provides optimal solution from a set of solutions[5, 12]. In this project it has been established that the seed for chaotic map can be optimized by using evolutionary search algorithms like GA, and this optimized seed will generate an optimal and pseudorandom bit sequence. So, the proposed algorithm is responsible for the optimization of the seed values and that will generate a unique and optimal pseudorandom bit sequence in each execution. These type of random bit sequence can be used for cryptographic applications[13, 41, 42, 50, 51] because of its pseudorandomness property.

GA is a metaheuristic search algorithm. It works on randomly taken initial population of chromosomes. Each chromosome from the population is associated with a fitness value. The fitness value is calculated by using a fitness calculation strategy. GA uses the Darwin principle of natural selection and applies genetic operations, like Elitism, Crossover, Mutation, etc. in an ordered way. It is an iterating process, i.e. aforementioned genetic operations are performed iteratively until a predefined termination condition is achieved[43, 33, 34, 39, 18]. In recent years RCGA gains significant importance over the Binary Coded Genetic Algorithms or BCGA due to its properties like, fast execution rate, efficiency and straightforwardness.

In this thesis a RCGA has been proposed to find the optimal values of Chaos parameters, Chapter 2 of this thesis describes the Chaos Theory and several types of chaotic maps that are used in this project, Chapter 3 describes the fundamental theory of GA, Chapter 4 describes the NIST specified all statistical tests, Chapter 5 considers to obtain optimal seed values of chaotic map using various chaotic equation like Logistic, Skew-tent and Cross-coupled version of each of them. Also this optimal system parameters has been used to generate a pseudorandom bit sequence, and NIST specified tests are performed to conform the randomness of the generated bit sequence. Results are discussed in Chapter 6. Conclusion and Future scope are given in Chapter 7 and the References at the end.

Chapter 2

Chaos Theory

Chaotic maps are evolution functions. In the domain of mathematics a chaotic map is not only simple in terms of interacting parts or rules, but also has pertinent properties of chaos. So, chaotic maps can be used to produce chaotic and unpredictable number sequence, for its deterministic simplicity and chaotic behavior. Each map works on a seed. A seed is combination of μ and X_0, where μ is the system or control parameter and X_0 is the initial population value. By using this seed, chaotic map generate X, where X is the population of pseudo random probabilistic values. This proposed method considers *4* types of chaotic maps and these are discussed in the Section 2.1-2.4.

2.1 Logistic chaotic map

Logistic chaotic map deploys a differential equation to use the time as continuous. In this map population value of current time step is the function of previous time step value[9, 38, 47, 14, 27]. The mathematical model of this map can be defined in Equation 2.1

$$X_{i+1} = F(X_i, \mu) = \mu X_i(1 - X_i) \tag{2.1}$$

The value of X_i where, $0 \leq i \leq n$ should be lies between 0 to 1 and the value of μ should be lies between *0 to 4*. The logistic map shows complex chaotic behavior, if μ lie between *3.5699456 to 4*. Population values in each time step will be different[48, 3, 29]. The optimal seed for this map is optimized by using the proposed method, which is then used to generate the

pseudo random bit sequence. The pseudorandom bit stream is generated by comparing the population values X_i with the arithmetic mean or m of X. This can be represented mathematically in Equation 2.2

$$P(i) = \begin{cases} 0 & \text{if, } X_i \leq mean(X_i) \\ 1 & \text{Otherwise} \end{cases} \quad \text{where, } 0 \leq i \leq n \tag{2.2}$$

Here P is the n bit optimal pseudorandom bit sequence. The arithmetic mean can be calculated by the Equation 2.3.

$$\chi^2 = \frac{\sum_{i=1}^{n} X(i)}{n} \tag{2.3}$$

2.2 Skew tent chaotic map

One dimensional chaotic map is simple but exhibits complex chaotic behavior[28]. Skew tent chaotic map is a one dimensional chaotic map. This map is ergodic. It deploys a uniform invariant density function[35, 37, 52]. In 2011 Adrian Luca et al had shown that a PRNG can generate a random bit sequence with zero redundancy, if the seed value for a tent map is close to 0.5[30]. This map can be defined mathematically in Equation 2.4

$$X_{i+1} = G(X_i, \mu) = \begin{cases} \frac{X_i}{\mu} & \text{where, } 0 \leq X_i \leq \mu \\ \frac{1-X_i}{1-\mu} & \text{where, } \mu \leq X_i \leq 1 \end{cases} \tag{2.4}$$

Here, the value of X_i and μ are lies in the range *0 to 1*. The values of current time step X_{i+1} is depend on μ as well as the previous time step value X_i. In 1997, Hasler and L. Maistrenko proved that that the value of μ should be in the range between *0 to 1*[21]. This map shows its complex chaotic behavior when μ is in the aforementioned range[37]. The optimal seed for this map is optimized by using the proposed method. This project shown that the proposed method is successfully able to generate optimal pseudorandom bit sequence by using this type of chaotic map as an objective function. The pseudo random bit sequence is generated by using the Equation 2.2.

2.3 Cross coupled Logistic map

Two chaotic maps are called cross coupled with each other, if they have the same number time step, i.e. same in population size, and shares the values

of each others population[49]. This project has shown that the proposed method is successfully able to generate optimal pseudorandom bit sequence by using cross coupled logistic map equation. This project considers two piecewise linear logistic map, which are cross coupled with each other. The mathematical model of this map is expressed in Equation 2.5-2.6

$$X_{i+1} = F_1(X_i, \mu_1) \tag{2.5}$$

$$X_{j+1} = F_2(X_j, \mu_2) \tag{2.6}$$

The functionality of $F(X_i, \mu_1)$ is already discussed in section 2.1. Here the optimal seed value for the first chaotic map is optimized by using the proposed method. The optimal seed value for the second chaotic map is fixed, where the value of $\mu = 3.999$ *and* $X_i = 0.66$[19]. The proposed method will generate an optimal pseudorandom bit sequence by comparing the time step values of each logistic chaotic map, by using the Equation 2.7.

$$P(i) = \begin{cases} 0 & \text{if, } X_i < X_j \\ 1 & \text{Otherwise} \end{cases} \quad \text{where, } 0 \leq i \leq n \tag{2.7}$$

2.4 Cross coupled Skew Tent map

Two piecewise linear Skew tent map can also be used for the cross coupling. This type of cross coupling is also shows complex chaotic behavior in output probabilistic values or population, if the optimal seed can be consider for population generation. This project also shown that proposed method is successfully able to optimize the optimal seed for this type of map, and generate optimal pseudo random bit sequence. The mathematical map of this map is expressed in Equation 2.8-2.9.

$$X_{i+1} = G_1(X_i, \mu_1) \quad \text{where, } 0 \leq i \leq n \tag{2.8}$$

$$X_{j+1} = G_2(X_j, \mu_2) \quad \text{where, } 0 \leq j \leq n \tag{2.9}$$

The working principle of the function $G(X_i, \mu_1)$ is already discussed in section 2.2. Here the optimal seed value for the first chaotic map is optimized by using this proposed method. The optimal seed value for the second chaotic

map is fixed, where the value of $\mu = 0.31$ and $X_i = 0.30$[37]. The proposed method is also generates the optimal pseudorandom bit sequence by using the Equation 2.7.

Chapter 3

Genetic Algorithm

Genetic Algorithms or GA is the meta-heuristic search and optimization techniques that mimic the process of natural evolution. It works on randomly taken initial population of chromosomes. Each chromosome from the population is associated with a fitness value. The fitness value is calculated by using a fitness calculation strategy. GA uses the Darwin principle of natural selection and applies genetic operations, like Elitism, Crossover, Mutation, etc. in an ordered way. Its an iterating process, i.e. aforementioned genetic operations are performed iteratively until a predefined termination condition is achieved[43, 39, 18]. The descriptions of the aforementioned genetic operations are described in the Section 3.3 in an ordered way. During the execution it follows the same order. Based on the encoding scheme GA can be divided into two types, viz. Binary Coded GA and Real Coded GA. The following Section 3.1-3.2 considers the Binary coded and Real Coded GA.

3.1 Binary Coded GA

In a genetic algorithm, a population of candidate solutions or individuals or creatures or phenotypes to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties which can be mutated and altered; traditionally, solutions are represented in binary as strings of 0s and 1s. These type of GA is known as Binary Coded GA or BCGA. The binary alphabet offers the maximum number of schemata per bit of information of any coding and consequently the bit string representation of solutions has dominated genetic algorithm research. This coding also

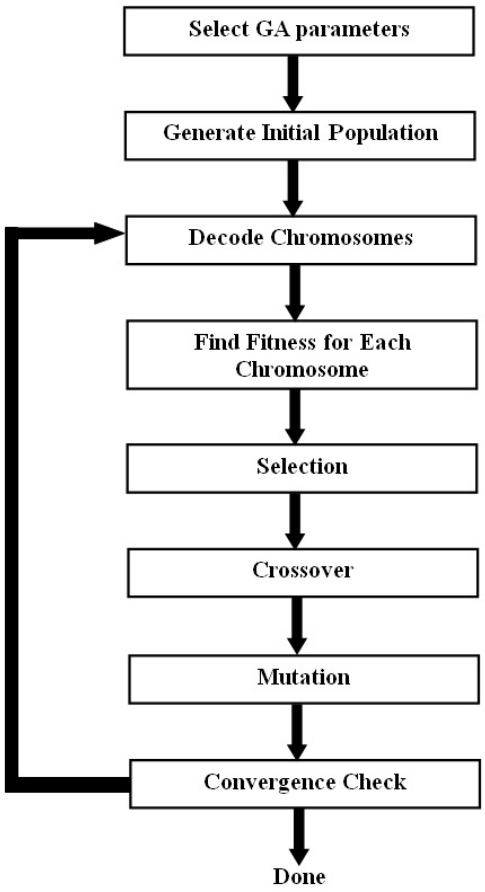

Figure 1: Flowchart of a GA

facilitates theoretical analysis and allows elegent genetic operators. But the *implicit parallelism* result does not depend on using bit strings and it may be worthwhile to experiment with large alphabets and new genetic operators.

BCGA suffers with some major drawbacks, like Hamming cliffs, Difficulty in achieving arbitrary precision, Uneven schema importance. Real Coded GA or RCGA gained its importance due to minimization of these drawbacks and its discussed in Section 3.2

Mating Pool										
Chromosome 1	1	0	0	1	0	1	0	1	0	0
Chromosome 2	0	1	1	1	1	0	1	0	1	1
	....									
Chromosome n	1	0	0	1	0	1	0	1	0	0

Figure 2: A binary coded initial population

3.2 Real Coded GA

The use of Real coded or Floating point genes has a long, if controversial, history in artificial genetic and evolutionary search schemes, and their use as of late seems to be on the rise. This rising usage has been somewhat surprising to researchers familiar with fundamental GA theory, because simple analyses seem to suggest that enhanced schema processing is obtained by using alphabets of low cardinality, a seemingly direct contradiction of empirical findings that real codings have worked well in a number of practical problems.

However, RCGA lgorithm is simple and straightforward. Here the selection operator is also based on the fitness values. Crossover and mutation operators for the RCGA need to be redefined. Section 3.3 consider the basic steps of a GA.

3.3 Basic steps in GA

The following six operation is executed in an ordered way for the optimization process. The Figure 1 represents the basic steps and the flow of operations in a GA.

3.3.1 Initial Encoding

Initial encoding is a process of encoding number of chromosomes randomly to constitute a initial population or mating pool. The rest of the genetic operations are performed iteratively on this initial mating pool. This encoding of chromosomes can be either binary coded or real coded. In this project the Initial population of chromosomes is real coded.

Figure 2 shows a binary coded initial mating pool which is the output of an Initial Encoding operation.

3.3.2 Fitness Evaluation

A fitness function value quantifies the optimality of a solution. The value is used to rank a particular solution against all the other solutions. A fitness value is assigned to each solution depending on how close it is actually to the optimal solution of the problem. In a generation the aim of genetic operations is to finding chromosomes with better fitness values than from the previous generation. The betterment of the fitness value can be assured either by minimizing or maximizing the fitness values. In minimization process the chromosome with the lowest fitness value is the fittest, whereas in maximization process the chromosome with the highest fitness value is the fittest. Thus it stipulates the degree of goodness of the population [23].

3.3.3 Elitism

Elitism is a selection procedure, where some of the fittest chromosomes from the current generation is selected and passes directly to the next generation. Thus is done by comparing the fitness value of the fittest chromosome X from the previous generation with the fitness value of the weakest chromosome Y from the current generation, i.e. if $fitness(X) > fitness(Y)$, then Y is replaced by the X in the output mating pool, otherwise Y survives in that generation.

3.3.4 Selection

Selection is a procedure to create a mating pool from the population. A mating pool consists of a set of chromosomes, which is generated by selecting the fittest chromosomes from the population and neglecting the weakest chromosomes from the population. The sizes of the population and of the mating pool are same. In tournament selection several tournaments are played among a few individuals. The individuals are chosen at random from the population. The winner of each tournament is selected for next generation. Selection pressure can be adjusted by changing the tournament size. Weak individuals have a smaller chance to be selected if tournament size is large. For a given tournament size, t , randomly select t individuals from the

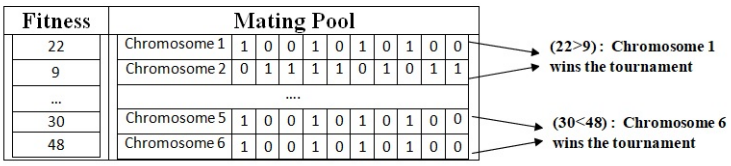

Figure 3: BTS operation

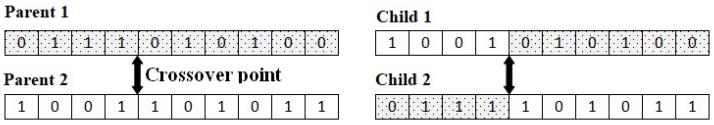

Figure 4: Single point Crossover operation on binary coded chromosomes

population and determine the winner of that tournament as the individual with the largest fitness function value. Tournament selection with the value of t is *2* is called *Binary Tournament Selection* or *BTS*.

BTS operation is described graphically in Figure 3. Here two chromosomes are selected randomly. After comparing fitness values of them, the fittest chromosome is selected and the other is discarded.

3.3.5 Crossover

Crossover is a process to share genetic information of two chromosomes and reproduce them with the intention to make them more fit. Basically it simulates the biological crossover and reproduction process. This process takes a mating pool as an input and performs crossover operation. In a crossover operation, first two chromosomes are randomly selected from the mating pool, these are known as parents. After that they share some genetic information with each other and reproduce two child chromosomes based on a very high crossover probabilistic value P_c. Let $P_c = 0.8$, then 80% of the selected chromosomes will go through the crossover, and 20% of them are copied directly to the output mating pool. This process iterates for $n/2$ times, where n is the size of the mating pool.

A single point crossover operation is described graphically in Figure 4. From this figure it can be clearly observed that the values up to the crossover

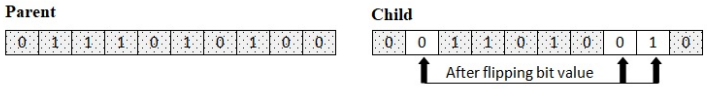

Figure 5: Mutation operation on binary coded chromosome

point of both chromosomes is swapped with each other.

3.3.6 Mutation

Mutation simulates the biological mutation process, where some percentage of genetic information of a chromosome is altered to make them more fit. This process takes the output mating pool from the crossover as an input and performs the mutation operation. In this process, one chromosome from the mating pool is randomly selected first, then this chromosomes is mutated based on a very low mutation probabilistic value P_m. Let, $P_m = 0.1$, then 10% genetic information of a chromosome is altered or mutated for the betterment of its fitness value and 90% of them remains un-mutated or unaltered.

A mutation operation is described graphically in Figure 5. From this figure it can be clearly observed that the values of the randomly chosen bit position is flipped, i.e if the bit value of the Parent is 0, then in child chromosome it will be flipped to 1 and if the bit value of the Parent is 1, then in child chromosome it will be flipped to 0.

Chapter 4

NIST Test Suit

The need for random and pseudorandom numbers arises in many cryptographic applications. For example, common cryptosystems employ keys that must be generated in a random fashion. Many cryptographic protocols also require random or pseudorandom inputs at various points, e.g., for auxiliary quantities used in generating digital signatures, or for generating challenges in authentication protocols.

NIST test for randomness testing of random number and pseudorandom number generators that may be used for many purposes including cryptographic, modeling and simulation applications. The focus of this test suit is on those applications where randomness is required for cryptographic purposes. A set of statistical tests for randomness is consider for this test suit. The National Institute of Standards and Technology (NIST) believes that these procedures are useful in detecting deviations of a binary sequence from randomness. However, a tester should note that apparent deviations from randomness may be due to either a poorly designed generator or to anomalies that appear in the binary sequence that is tested (i.e., a certain number of failures is expected in random sequences produced by a particular generator). Output of each test is defined by using a probabilistic value or *P-value*. Section 4.1 describes all the statistical tests of this test suit.

In the rest of this chapter following symbols are used to represent different operation or values:

1. M: The length of each block or The number of rows in each matrix.

2. Q: The number of columns in each matrix.

3. N: The number of blocks; selected in accordance with the value of M.
4. n: The length of the bit string.
5. m: The length in bits of each template.
6. ϵ: The sequence of bits as generated by the RNG or PRNG being tested; $\epsilon = \epsilon_1, \epsilon_2, \ldots, \epsilon_n$.
7. $\chi^2(obs)$ or $V_n(obs)$: Test statistic and reference distribution.
8. $P - value$: Output Probabilistic value of a test.
9. d: The normalized difference between the observed and the expected number of frequency components that are beyond the 95% threshold.
10. K: The number of degrees of freedom.
11. f_n: The sum of the log_2 distances between matching L-bit templates, i.e., the sum of the number of digits in the distance between L-bit templates.
12. $\nabla\Psi^2_m(obs)$ and $\nabla^2\Psi^2_m(obs)$: A measure of how well the observed frequencies of m-bit patterns match the expected frequencies of the m-bit patterns.
13. z: The largest excursion from the origin of the cumulative sums in the corresponding $(-1, +1)$ sequence.
14. ξ: For a given state x, the total number of times that the given state is visited during the entire random walk.
15. $erfc()$: Complimentary Error Function.
16. $igamc()$: Incomplete Gamma Function.

4.1 Random Number Generation Tests

The NIST Test Suite is a statistical package consisting of 15 tests that were developed to test the randomness of (arbitrarily long) binary sequences produced by either hardware or software based cryptographic random or pseudorandom number generators. These tests focus on a variety of different

types of non-randomness that could exist in a sequence. Some tests are decomposable into a variety of subtests. The 15 tests are:

1. The Frequency (Monobit) Test.
2. Frequency Test within a Block.
3. The Runs Test.
4. Tests for the Longest-Run-of-Ones in a Block.
5. The Binary Matrix Rank Test.
6. The Discrete Fourier Transform (Spectral) Test.
7. The Non-overlapping Template Matching Test.
8. The Overlapping Template Matching Test.
9. Maurers *Universal Statistical* Test.
10. The Linear Complexity Test.
11. The Serial Test.
12. The Approximate Entropy Test.
13. The Cumulative Sums (Cusums) Test.
14. The Random Excursions Test.
15. The Random Excursions Variant Test.

4.1.1 Frequency (Monobit) Test

The focus of the test is the proportion of zeroes and ones for the entire sequence. The purpose of this test is to determine whether the number of ones and zeros in a sequence are approximately the same as would be expected for a truly random sequence. The test assesses the closeness of the fraction of ones to 1/2, that is, the number of ones and zeroes in a sequence should be about the same. All subsequent tests depend on the passing of this test. Algorithm 1 defines the method for this test.

Algorithm 1 Method for Monobit test

Input: Bit sequence of N bit

Output: $P - value$

1: **begin**
2: Conversion to ±1: The zeros and ones of the input sequence (ϵ) are converted to values of –1 and +1 and are added together to produce $S_n = X_1 + X_2 + ... + X_n$, where, $X_i = 2\epsilon_i$–1.
3: Compute the test statistic $s_{obs} = \frac{|S_n|}{\mu}$
4: Compute $P - value = erfc(\frac{S_{obs}}{\sqrt{2}})$
5: **end**

4.1.2 Frequency Test within a Block

The focus of the test is the proportion of ones within M-bit blocks. The purpose of this test is to determine whether the frequency of ones in an M-bit block is approximately $M/2$, as would be expected under an assumption of randomness. For block size $M = 1$, this test degenerates to test 1, the Frequency (Monobit) test. Algorithm 2 defines the method for this test.

Algorithm 2 Method for Frequency Test within a Block

Input: Bit sequence of N bit

Output: $P - value$

1: **begin**
2: Partition the input sequence into $N = \lfloor \frac{n}{M} \rfloor$. Discard any unused bits.
3: Determine the proportion π_i of ones in each M-bit block using the equation $\pi_i = \frac{\sum_{j=1}^{M} \epsilon_{(i-1)M+j}}{M}$, for $1 \leq i \leq N$.
4: Compute the χ^2 statistic: $\chi^2(obs) = 4M \sum_{i=1}^{N} (\pi_i - \frac{1}{2})^2$
5: Compute $P - value = igamc(\frac{N}{2}, \frac{\chi^2(obs)}{2})$, where $igamc$ is the incomplete gamma function.
6: **end**

4.1.3 Runs Test

The focus of this test is the total number of runs in the sequence, where a run is an uninterrupted sequence of identical bits. A run of length k consists of exactly k identical bits and is bounded before and after with a bit of

the opposite value. The purpose of the runs test is to determine whether the number of runs of ones and zeros of various lengths is as expected for a random sequence. In particular, this test determines whether the oscillation between such zeros and ones is too fast or too slow. Algorithm 3 defines the method for this test.

Algorithm 3 Method for Runs Test

Input: Bit sequence of N bit

Output: $P - value$

1: **begin**
2: Compute the pre-test proportion π of ones in the input sequence: $\pi = \frac{\sum_j \epsilon_j}{n}$.
3: Determine if the prerequisite Frequency test is passed: If it can be shown that $|\pi - \frac{1}{2}| \geq \tau$, then the Runs test need not be performed (i.e., the test should not have been run because of a failure to pass test 1, the Frequency (Monobit) test). If the test is not applicable, then the $P - value$ is set to 0.0000. Note that for this test, $\tau = \frac{2}{\sqrt{n}}$.
4: Compute the χ^2 statistic: $\chi^2(obs) = 4M \sum_{i=1}^{N} (\pi_i - \frac{1}{2})^2$
5: Compute the test statistic $V_n(obs) = \sum_{k=1}^{n-1} r(k) + 1$, where $r(k) = 0$ if $\epsilon_k = \epsilon_{k+1}$, and $r(k) = 1$ otherwise.
6: Compute $P - value = erfc(\frac{|V_n(obs) - 2n\pi(1-\pi)|}{2\sqrt{2n\pi(1-\pi)}})$
7: **end**

4.1.4 Test for the Longest Run of Ones in a Block

The focus of the test is the longest run of ones within M-bit blocks. The purpose of this test is to determine whether the length of the longest run of ones within the tested sequence is consistent with the length of the longest run of ones that would be expected in a random sequence. Note that an irregularity in the expected length of the longest run of ones implies that there is also an irregularity in the expected length of the longest run of zeroes. Therefore, only a test for ones is necessary. Algorithm 4 defines the method for this test.

Algorithm 4 Method for Longest Run of Ones test

Input: Bit sequence of N bit

Output: $P - value$

1: **begin**
2: Divide the sequence into M-bit blocks.
3: Tabulate the frequencies v_i of the longest runs of ones in each block into categories, where each cell contains the number of runs of ones of a given length. For the value of M supported by the test code, the v_i cells will hold the counts mentioned in Table 1.
4: Compute $\chi^2(obs) = \sum_{i=0}^{k} \frac{(v_i - N\pi_i)^2}{N\pi_i}$, where the values for π_i are provided in Section 3.4 of [4]. The values of K and N are determined by the value of M in accordance with the Table 2.
5: Compute $P - value = igamc(\frac{K}{2}, \frac{\chi^2(obs)}{2})$.
6: **end**

Table 1: Values of v_i and its corresponding value of M

v_i	$M = 8$	$M = 128$	$M = 10^4$
v_0	≤ 1	≤ 4	≤ 10
v_1	2	5	11
v_2	3	6	12
v_3	≥ 4	7	13
v_4		8	14
v_5		≥ 9	15
v_6			≥ 16

Table 2: Values of $M, K\&N$

M	K	N
8	3	16
128	5	49
10^4	6	75

4.1.5 Binary Matrix Rank Test

The focus of the test is the rank of disjoint sub-matrices of the entire sequence. The purpose of this test is to check for linear dependence among fixed length substrings of the original sequence. Note that this test also ap-

pears in the DIEHARD battery of tests [32]. Algorithm 5 defines the method for this test.

Algorithm 5 Method for Binary Matrix Rank Test

Input: Bit sequence of N bit

Output: $P - value$

1: **begin**

2: Sequentially divide the sequence into $M.Q$-bit disjoint blocks; there will exist $N = \lfloor \frac{n}{MQ} \rfloor$ such blocks. Discarded bits will be reported as not being used in the computation within each block. Collect the $M.Q$-bit segments into M by Q matrices. Each row of the matrix is filled with successive Q-bit blocks of the original sequence ϵ.

3: Determine the binary rank (R_l) of each matrix, where $l = 1, ..., N$.

4: Calculate F_M = the number of matrices with $R_l = M$ (full rank), F_{M-1} = the number of matrices with $R_l = M - 1$ (full rank -1), N–$F_M - F_{M-1}$ = the number of matrices remaining.

5: Compute $\chi^2(obs) = \frac{(F_M - 0.2888N)^2}{0.2888N} + \frac{(F_{M-1} - 0.5776N)^2}{0.5776N} + \frac{(N - F_M - F_{M-1} - 0.1336N)^2}{0.1336N}$

6: Compute $P - value = e^{-\frac{\chi^2(obs)}{2}}$.

7: **end**

4.1.6 Discrete Fourier Transform (Spectral) Test

The focus of this test is the peak heights in the Discrete Fourier Transform of the sequence. The purpose of this test is to detect periodic features (i.e., repetitive patterns that are near each other) in the tested sequence that would indicate a deviation from the assumption of randomness. The intention is to detect whether the number of peaks exceeding the 95% threshold is significantly different than 5%. Algorithm 6 defines the method for this test.

4.1.7 Non-overlapping Template Matching Test

The focus of this test is the number of occurrences of pre-specified target strings. The purpose of this test is to detect generators that produce too many occurrences of a given non-periodic (aperiodic) pattern. For this test and for the Overlapping Template Matching test of Section 4.1.8, an m-bit

Algorithm 6 Method for Discrete Fourier Transform (Spectral) Test

Input: Bit sequence of N bit

Output: $P - value$

1: **begin**
2: The zeros and ones of the input sequence (ϵ) are converted to values of –1 and +1 to create the sequence $X = x_1, x_2, ..., x_n$, where $x_i = 2\epsilon_i - 1$.
3: Apply a Discrete Fourier transform (DFT) on X to produce: $S = DFT(X)$. A sequence of complex variables is produced which represents periodic components of the sequence of bits at different frequencies.
4: Calculate $M = modulus(S^{|}) \equiv |S^{|}|$, where $S^{|}$ is the substring consisting of the first $\frac{n}{2}$ elements in S, and the modulus function produces a sequence of peak heights.
5: Compute $T = \sqrt{(log\frac{1}{0.05})n}$ = the 90% peak height threshold value. Under an assumption of randomness, 95% of the values obtained from the test should not exceed T.
6: Compute $N_0 = 0.95\frac{n}{2}$. N_0 is the expected theoretical (95%) number of peaks (under the assumption of randomness) that are less than T.
7: Compute N_1 = the actual observed number of peaks in M that are less than T.
8: Compute $d = \frac{(N_1 - N_0)}{\sqrt{\frac{n(0.95)(0.05)}{4}}}$.
9: Compute $P - value = erfc(\frac{|d|}{\sqrt{2}})$.
10: **end**

window is used to search for a specific m-bit pattern. If the pattern is not found, the window slides one bit position. If the pattern is found, the window is reset to the bit after the found pattern, and the search resumes. Algorithm 7 defines the method for this test.

4.1.8 Overlapping Template Matching Test

The focus of the Overlapping Template Matching test is the number of occurrences of pre-specified target strings. Both this test and the Non-overlapping Template Matching test of Section 4.1.7 use an m-bit window to search for a specific m-bit pattern. As with the test in Section 4.1.7, if the pattern is not found, the window slides one bit position. The difference between this test and the test in Section 4.1.7 is that when the pattern is found, the win-

Algorithm 7 Method for Non-overlapping Template Matching Test

Input: Bit sequence of N bit

Output: $P - value$

1: **begin**
2: Partition the sequence into N independent blocks of length M.
3: Let $W_j(j = 1, ..., N)$ be the number of times that B (the template) occurs within the block j. Note that $j = 1, \ldots, N$. The search for matches proceeds by creating an m-bit window on the sequence, comparing the bits within that window against the template. If there is no match, the window slides over one bit , e.g., if $m = 3$ and the current window contains bits 3 to 5, then the next window will contain bits 4 to 6. If there is a match, the window slides over m bits, e.g., if the current (successful) window contains bits 3 to 5, then the next window will contain bits 6 to 8.
4: Under an assumption of randomness, compute the theoretical mean $\mu = \frac{M-m+1}{2^m}$ and variance $\sigma^2 = M(\frac{1}{2^m} - \frac{2m-1}{2^{2m}})$.
5: Compute the $\chi^2(obs) = \sum_{j=1}^{N} \frac{(W_j-\mu)^2}{\sigma^2}$.
6: Compute $P - value = igamc(\frac{N}{2}, \frac{\chi^2(obs)}{2})$.
7: **end**

dow slides only one bit before resuming the search. Algorithm 8 defines the method for this test.

4.1.9 Maurers *Universal Statistical* Test

The focus of this test is the number of bits between matching patterns (a measure that is related to the length of a compressed sequence). The purpose of the test is to detect whether or not the sequence can be significantly compressed without loss of information. A significantly compressible sequence is considered to be non-random. Algorithm 9 defines the method for this test.

4.1.10 Linear Complexity Test

The focus of this test is the length of a linear feedback shift register (LFSR). The purpose of this test is to determine whether or not the sequence is complex enough to be considered random. Random sequences are characterized

Algorithm 8 Method for Overlapping Template Matching Test

Input: Bit sequence of N bit

Output: $P - value$

1: **begin**
2: Partition the sequence into N independent blocks of length M.
3: Calculate the number of occurrences of B in each of the N blocks. The search for matches proceeds by creating an m-bit window on the sequence, comparing the bits within that window against B and incrementing a counter when there is a match. The window slides over one bit after each examination, e.g., if $m = 4$ and the first window contains bits 42 to 45, the next window consists of bits 43 to 46. Record the number of occurrences of B in each block by incrementing an array v_i (where $i = 0, \ldots, 5$), such that v_0 is incremented when there are no occurrences of B in a substring, v_1 is incremented for one occurrence of B, and v_5 is incremented for 5 or more occurrences of B.
4: Compute values for $\lambda = \frac{M-m+1}{2^m}$ and $\eta = \frac{\lambda}{2}$.
5: Compute the $\chi^2(obs) = \sum_{i=0}^{5} \frac{(v_i - N\pi_i)^2}{N\pi_i}$, where $\pi_0 = 0.364091$, $\pi_1 = 0.185659$, $\pi_2 = 0.139381$, $\pi_3 = 0.100571$, $\pi_4 = 0.070432$ and $\pi_5 = 0.139865$.
6: Compute $P - value = igamc(\frac{5}{2}, \frac{\chi^2(obs)}{2})$.
7: **end**

Table 3: Precomputed values of $expectedValue(L)$ and $Variance$

L	expectedValue	Variance or Σ
6	5.2177052	2.954
7	6.1962507	3.125
8	7.1836656	3.238
9	8.1764248	3.311
10	9.1723243	3.356
11	10.170032	3.384
12	11.168765	3.401
13	12.168070	3.410
14	13.167693	3.416
15	14.167488	3.419
16	15.167379	3.421

Algorithm 9 Method for Maurers *Universal Statistical* Test

Input: Bit sequence of N bit

Output: $P - value$

1: **begin**

2: The n-bit sequence (ϵ) is partitioned into two segments: an initialization segment consisting of QL-bit non-overlapping blocks, and a test segment consisting of KL-bit non-overlapping blocks. Bits remaining at the end of the sequence that do not form a complete L-bit block are discarded. The first Q blocks are used to initialize the test. The remaining K blocks are the test blocks ($K = \lfloor \frac{n}{L} \rfloor - Q$). Figure 6 summarizes the whole process of the aforementioned partitioning procedure.

3: Using the initialization segment, a table is created for each possible L-bit value (i.e., the L-bit value is used as an index into the table). The block number of the last occurrence of each L-bit block is noted in the table (i.e., For i from 1 to Q, $T_j = i$, where j is the decimal representation of the contents of the i^{th} L-bit block).

4: Examine each of the K blocks in the test segment and determine the number of blocks since the last occurrence of the same L-bit block (i.e., i–T_j). Replace the value in the table with the location of the current block (i.e., $T_j = i$). Add the calculated distance between re-occurrences of the same L-bit block to an accumulating log_2 sum of all the differences detected in the K blocks (i.e., $sum = sum + log_2(i–T_j)$).

5: Compute the test statistic, $f_n = \frac{1}{K} \sum_{i=Q+1}^{Q+K} log_2(i - T_j)$ where T_j is the table entry corresponding to the decimal representation of the contents of the i^{th} L-bit block.

6: Compute $P-value = erfc(|\frac{f_n - expectedValue(L)}{\sqrt{2}\sigma}|)$, where $expectedValue(L)$ and σ are taken from the Table 3 of precomputed values [1]. Under an assumption of randomness, the sample mean, $expectedValue(L)$, is the theoretical expected value of the computed statistic for the given L-bit length. The theoretical standard deviation is given by $\sigma = c\sqrt{\frac{variance(L)}{K}}$, where $c = 0.7 - \frac{0.8}{L} + (4 + \frac{32}{L})\frac{K^{-\frac{3}{L}}}{15}$.

7: **end**

by longer LFSRs. An LFSR that is too short implies non-randomness. Algorithm 10 defines the method for this test.

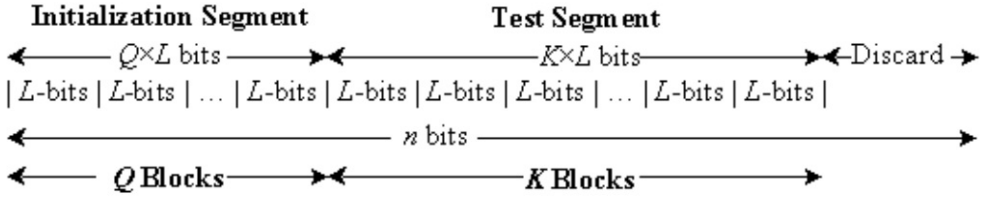

Figure 6: Partitioning of the input n-bit sequence

Algorithm 10 Method for Linear Complexity Test

Input: Bit sequence of N bit

Output: $P - value$

1: **begin**
2: Partition the n-bit sequence into N independent blocks of M bits, where $n = MN$.
3: Using the Berlekamp-Massey algorithm, determine the linear complexity L_i of each of the N blocks ($i = 1, \ldots, N$). L_i is the length of the shortest linear feedback shift register sequence that generates all bits in the block i. Within any L_i-bit sequence, some combination of the bits, when added together modulo 2, produces the next bit in the sequence (bit $L_i + 1$).
4: Under an assumption of randomness, compute the theoretical mean $\mu = \frac{M}{2} + \frac{(9+(-1)^{M+1})}{36} - \frac{(\frac{M}{3}+\frac{2}{9})}{2^M}$.
5: For each substring, calculate a value of T_i, where $T_i = (-1)^M(L^i - \mu) + \frac{2}{9}$
6: Record the T_i values in $v_0, \ldots, v_6$ as follows: If $T_i \leq -2.5$, then increment v_o by one, if $-2.5 \leq T_i \leq -1.5$, then increment v_1 by one, if $-0.5 \leq T_i \leq 0.5$, then increment v_2 by one, if $0.5 \leq T_i \leq 1.5$, then increment v_3 by one, if $1.5 \leq T_i \leq 2.5$, then increment v_4 by one, if $T_i > 2.5$, then increment v_6 by one.
7: Compute $\chi^2(obs) = \sum_{i=0}^{K} \frac{(v_i - N\pi_i)^2}{N\pi_i}$, where $\pi_0 = 0.010417$, $\pi_1 = 0.03125$, $\pi_2 = 0.125$, $\pi_3 = 0.5$, $\pi_4 = 0.25$,$\pi_5 = 0.0625$, $\pi_6 = 0.020833$ are the values of π_i.
8: Compute $P - value = igamc(\frac{K}{2}, \frac{\chi^2(obs)}{2})$.
9: **end**

4.1.11 Serial Test

The focus of this test is the frequency of all possible overlapping m-bit patterns across the entire sequence. The purpose of this test is to determine

whether the number of occurrences of the 2^m m-bit overlapping patterns is approximately the same as would be expected for a random sequence. Random sequences have uniformity; that is, every m-bit pattern has the same chance of appearing as every other m-bit pattern. Note that for $m = 1$, the Serial test is equivalent to the Frequency test of Section 4.1.1. Algorithm 11 defines the method for this test.

Algorithm 11 Method for Serial Test

Input: Bit sequence of N bit
Output: $P - value$

1: **begin**
2: Form an augmented sequence $\epsilon^{|}$: Extend the sequence by appending the first $m-1$ bits to the end of the sequence for distinct values of n.
3: Determine the frequency of all possible overlapping m-bit blocks, all possible overlapping $(m-1)$-bit blocks and all possible overlapping $(m-2)$-bit blocks. Let denote the frequency of the m-bit pattern $i_1 \dots i_m$; let denote the frequency of the $(m-1)$-bit pattern $i_1 \dots i_{m-1}$; and let $v_{i_1 \dots i_{m-1}}$ denote the frequency of the $(m-1)$-bit pattern $i_1 \dots i_{m-1}$; and let $v_{i_1 \dots i_{m-2}}$ denote the frequency of the $(m-2)$-bit pattern $i_1 \dots i_{m-2}$.
4: Compute: $\Psi^2_m = \frac{2^m}{n} \sum_{i_1 \dots i_m} (v_{i_1 \dots i_m} - \frac{n}{2^m})^2 = \frac{2^m}{n} \sum_{i_1 \dots i_m} (v^2_{i_1 \dots i_m} - n)$,
$\Psi^2_{m-1} = \frac{2^{m-1}}{n} \sum_{i_1 \dots i_{m-1}} (v_{i_1 \dots i_{m-1}} - \frac{n}{2^{m-1}})^2 = \frac{2^{m-1}}{n} \sum_{i_1 \dots i_{m-1}} (v^2_{i_1 \dots i_{m-1}} - n)$,
$\Psi^2_{m-2} = \frac{2^{m-2}}{n} \sum_{i_1 \dots i_{m-2}} (v_{i_1 \dots i_{m-2}} - \frac{n}{2^{m-2}})^2 = \frac{2^{m-2}}{n} \sum_{i_1 \dots i_{m-2}} (v^2_{i_1 \dots i_{m-2}} - n)$.
5: Compute: $\nabla \Psi^2_m = \Psi^2_m - \Psi^2_{m-1}$,
$\nabla^2 \Psi^2_m = \Psi^2_m - 2\Psi^2_{m-1} + \Psi^2_{m-2}$.
6: Compute: $P - value1 = igamc(2^{m-2}, \nabla \Psi^2_m)$,
$P - value2 = igamc(2^{m-3}, \nabla^2 \Psi^2_m)$.
7: **end**

4.1.12 Approximate Entropy Test

As with the Serial test of Section 4.1.11, the focus of this test is the frequency of all possible overlapping m-bit patterns across the entire sequence. The purpose of the test is to compare the frequency of overlapping blocks of two consecutive/adjacent lengths (m and $m+1$) against the expected result for a random sequence. Algorithm 12 defines the method for this test.

Algorithm 12 Method for Approximate Entropy Test

Input: Bit sequence of N bit

Output: $P - value$

1: **begin**
2: Augment the n-bit sequence to create n overlapping m-bit sequences by appending $m-1$ bits from the beginning of the sequence to the end of the sequence.
3: A frequency count is made of the n overlapping blocks (e.g., if a block containing ϵ_j to ϵ_{j+m-1} is examined at time j, then the block containing $\epsilon_{j+1} to \epsilon_{j+m}$ is examined at time $j+1$). Let the count of the possible m-bit ($(m+1)$-bit) values be represented as C_m^i, where i is the m-bit value.
4: Compute $C_m^i = \frac{\#i}{n}$ for each value of i.
5: Compute $\varphi^{(m)} = \sum_{i=0}^{2^{m-1}} (\pi_i log \pi_i)$, where $\pi_i = C_j^3$, and $j = log_2 i$.
6: Repeat steps 1-4, replacing m by $m+1$.
Augment the n-bit sequence to create n overlapping m-bit sequences by appending $m-1$ bits from the beginning of the sequence to the end of the sequence.
Extract n overlapping m-bit blocks.
Compute C_i^m by using the aforementioned equation.
Compute $\varphi^{(m)}$ by using the aforementioned equation.
7: Compute the test statistic: $\chi^2 = 2n[log2 - ApEn(m)]$, where $ApEn(m) = \varphi^{(m)} - \varphi^{(m+1)}$.
8: Compute $P - value = igamc(2^{m-1}, \frac{\chi^2}{2})$.
9: **end**

4.1.13 Cumulative Sums (Cusum) Test

The focus of this test is the maximal excursion (from zero) of the random walk defined by the cumulative sum of adjusted $(-1, +1)$ digits in the sequence. The purpose of the test is to determine whether the cumulative sum of the partial sequences occurring in the tested sequence is too large or too small relative to the expected behavior of that cumulative sum for random sequences. This cumulative sum may be considered as a random walk. For a random sequence, the excursions of the random walk should be near zero. For certain types of non-random sequences, the excursions of this random walk from zero will be large. Algorithm 13 defines the method for this test.

Algorithm 13 Method for Cumulative Sums (Cusum) Test

Input: Bit sequence of N bit

Output: $P - value$

1: **begin**

2: Form a normalized sequence: The zeros and ones of the input sequence (ϵ) are converted to values X_i of –1 and +1 using $X_i = 2\epsilon_i$–1.

3: Compute partial sums S_i of successively larger subsequences, each starting with X_1 (if $mode = 0$) or X_n (if $mode = 1$). That is, $S_k = S_{k-1} + X_k$, for $mode = 0$, and $Sk = S_{k-1} + X_{n-k+1}$, for $mode = 1$. Detailed procedure is presented in Table 4.

4: Compute the test statistic $z = max_{1 \leq k \leq n}|S_k|$, where $max_{1 \leq k \leq n}|S_k|$ is the largest of the absolute values of the partial sums S_k.

5: Compute $P - value = 1 - \sum_{k=\frac{(\frac{-n}{z}+1)}{4}}^{\frac{(\frac{n}{z}-1)}{4}} [\Phi(\frac{(4k+1)z}{\sqrt{n}}) - \Phi(\frac{(4k-1)z}{\sqrt{n}})] + \sum_{k=\frac{(\frac{-n}{z}-3)}{4}}^{\frac{(\frac{n}{z}-1)}{4}} [\Phi(\frac{(4k+3)z}{\sqrt{n}}) - \Phi(\frac{(4k+1)z}{\sqrt{n}})]$.

6: **end**

Table 4: Partial sum with different mode

$Mode = 0$ (forward)	$Mode = 1$ (backward)
$S_1 = X_1$	$S_1 = X_n$
$S_2 = X_1 + X_2$	$S_2 = X_n + X_{n-1}$
$S_3 = X_1 + X_2 + X_3$	$S_3 = X_n + X_{n-1} + X_{n-2}$
.	.
.	.
$S_k = X_1 + X_2 + X_3 + ... + X_k$	$S_k = X_n + X_{n-1} + X_{n-2} + ... + X_{n-k+1}$
.	.
.	.
$S_n = X_1 + X_2 + X_3 ... + X_k + ... + X_n$	$S_n = X_n + X_{n-1} + X_{n-2} ... + X_{k-1} + ... + X_1$

4.1.14 Random Excursions Test

The focus of this test is the number of cycles having exactly K visits in a cumulative sum random walk. The cumulative sum random walk is derived from partial sums after the $(0, 1)$ sequence is transferred to the appropriate $(-1, +1)$ sequence. A cycle of a random walk consists of a sequence of steps

of unit length taken at random that begin at and return to the origin. The purpose of this test is to determine if the number of visits to a particular state within a cycle deviates from what one would expect for a random sequence. This test is actually a series of eight tests (and conclusions), one test and conclusion for each of the states: -4, -3, -2, -1 and $+1$, $+2$, $+3$, $+4$. Algorithm 14 defines the method for this test.

4.1.15 Random Excursions Variant Test

The focus of this test is the total number of times that a particular state is visited (i.e., occurs) in a cumulative sum random walk. The purpose of this test is to detect deviations from the expected number of visits to various states in the random walk. This test is actually a series of eighteen tests (and conclusions), one test and conclusion for each of the states: -9, -8, ..., -1 and $+1$, $+2$, ..., $+9$. Algorithm 15 defines the method for this test.

4.2 Decision Rule (at the 1% Level)

For each test, if the computed $P - value$ is < 0.01, then conclude that the sequence is non-random. Otherwise, conclude that the sequence is random. If the computed $P - value$ for all tests is ≥ 0.01, then only it will be concluded with 99% confidence, that the output of the PRNG or RNG will be random.

Algorithm 14 Method for Random Excursions Test

Input: Bit sequence of N bit

Output: $P - value$

1: **begin**

2: Form a normalized $(-1, +1)$ sequence X: The zeros and ones of the input sequence (ϵ) are changed to values of –1 and +1 via $X_i = 2\epsilon_i – 1$.

3: Compute the partial sums S_i of successively larger subsequences, each starting with X_1. Form the set $S = \{S_i\}$.

$S_1 = X_1$

$S_2 = X_1 + X_2$

$S_3 = X_1 + X_2 + X_3$

.

.

$S_k = X_1 + X_2 + X_3 + ... + X_k$

.

.

$S_n = X_1 + X_2 + X_3 + ... + X_k + ... + X_n$

4: Form a new sequence $S^{|}$ by attaching zeros before and after the set S. That is, $S^{|} = 0, s_1, s_2, \ldots, s_n, 0$.

5: Let $J =$ the total number of zero crossings in $S^{|}$, where a zero crossing is a value of zero in $S^{|}$ that occurs after the starting zero. J is also the number of cycles in $S^{|}$, where a cycle of $S^{|}$ is a subsequence of $S^{|}$ consisting of an occurrence of zero, followed by no-zero values, and ending with another zero. The ending zero in one cycle may be the beginning zero in another cycle. The number of cycles in $S^{|}$ is the number of zero crossings. If $J < 500$, discontinue the test.

6: For each cycle and for each non-zero state value x having values $–4 \leq x \leq -1$ and $1 \leq x \leq 4$, compute the frequency of each x within each cycle.

7: For each of the eight states of x, compute $V_k(x) =$ the total number of cycles in which state x occurs exactly k times among all cycles, for $k = 0, 1, \ldots, 5$ (for $k = 5$, all frequencies ≥ 5 are stored in $V_5(x)$). Note that, $\sum_{k=0}^{5} V_k(x) = J$.

8: For each of the eight states of x, compute the test statistic $\chi^{(}obs) = \sum_{0}^{5} \frac{(V_k(x) - J\pi_k(x))^2}{J\pi_k(x)}$, where $\pi_k(x)$ is the probability that the state x occurs k times in a random distribution. Finally eight χ^2 statistics will be produced (i.e., for $x = -4, -3, -2, -1, 1, 2, 3, 4$).

9: Compute, $P - value = igamc(\frac{5}{2}, \frac{\chi^2(obs)}{2})$. Eight $P - values$ will be produced.

10: **end**

Algorithm 15 Method for Random Excursions Variant Test

Input: Bit sequence of N bit

Output: $P - value$

1: **begin**

2: Form the normalized $(-1, +1)$ sequence X: The zeros and ones of the input sequence (ϵ) are converted to values of –1 and +1 via $X = X_1, X_2, \ldots, X_n$, where $X_i = 2\epsilon_i$–1.

3: Compute the partial sums S_i of successively larger subsequences, each starting with X_1. Form the set $S = \{S_i\}$.

$S_1 = X_1$

$S_2 = X_1 + X_2$

$S_3 = X_1 + X_2 + X_3$

.

.

$S_k = X_1 + X_2 + X_3 + ... + X_k$

.

.

$S_n = X_1 + X_2 + X_3 + ... + X_k + ... + X_n$

4: Form a new sequence $S^{|}$ by attaching zeros before and after the set S. That is, $S^{|} = 0, s_1, s_2, \ldots, s_n, 0$.

5: For each of the eighteen non-zero states of x, compute $\xi(x)$ = the total number of times that state x occurred across all J cycles.

6: For each $\xi(x)$, compute $P - value = erfc(\frac{|\xi(x) - J|}{\sqrt{2J(4|x| - 2)}})$. Finally eighteen $P - values$ are computed.

7: **end**

Chapter 5

Proposed Method

The proposed technique consists of three step process. In the first step an initial population of size n is encoded. Here n is the number of chromosomes. In the second step different chaotic maps are consider separately. The system parameters of one of these chaotic maps are optimized by using genetic algorithm. In the third step these optimized system parameters are used to generate a pseudorandom bit sequence by using the equation of the chaotic map. The randomness of the generated bit sequence is tested using NIST test suit.

This proposed method consider Logistic, Skew tent, Cross coupled Logistic and Cross Coupled Skew tent map equations. One of these equation can be used for the generation of pseudorandom bit sequence. Randomly taken seeds for a chaotic map is consider as the initial population for this proposed method.

The length of the pseudorandom bit sequence and n are taken as a user input. A seed is constituted by using two parameters X and μ of a chaotic map. So, the length of the chromosomes is *2*. We have assumed that the values of P_c and P_m are *0.9* and *0.1* respectively. The value of P_c and P_m can also be taken as user input, but it have to consider that P_c must be a very high probabilistic value and P_m must be a very low probabilistic value. Total number of generations is *200*. The proposed algorithm is presented in Algorithm 16.

Section 5.1 consider the initial encoding process. Section 5.2 consider the optimization of seed values. Section 5.3 consider the pseudorandom bit sequence generation process.

Algorithm 16 Proposed method for Pseudorandom Bit stream Generation

Input: Number of chromosomes n, Length of the pseudorandom bit sequence is q

Output: Optimized seed for chaos and pseudorandom bit stream

1: **begin**
2: Initial population encoding, where each chromosome is the seed for chaos and its real coded
3: Optimal seed value for chaotic map optimization by using RCGA
4: Optimal pseudorandom bit sequence generation.
5: **end**

5.1 Initial Population Encoding

Initial mating pool or population has n number of chromosomes. Each chromosome is real coded and it represents a seed of a chaotic map. The first and second value of a seed are X_0 and μ. The value of X_0 should be in the range *0 to 1*, and it is chosen randomly. The range of the value of μ is different for different chaotic maps. For Equation 2.1 and for Equation 2.5-2.6 the value of μ should be in the range *3 to 4*, and for the Equation 2.4 and Equation 2.8-2.9 the value of μ should be in the range *0 to 1*.

5.2 Optimal seed value for chaos optimization

Optimal seed values for chaos are optimized by using a RCGA. This method is presented in Algorithm 17. Here the fitness value of each chromosome is calculated first. It generates a bit sequence of size *128* bit for each chromosome by using a chaotic map. After that the randomness of the bit sequence correspond to each chromosomes is tested by using Poker test. The result of the poker test is a chi-square value χ^2 which is also the fitness value for a chromosome. The proposed method consider this Poker test function, i.e. Equation 5.1 as the objective function to minimize the χ^2 value in each generation. In a poker test for a K bit of bit-stream, a positive integer L is selected in such a way that $\frac{X_i}{\mu} \geq 5 \times 2^L$. After that $Z = \frac{n}{L}$ is calculated. After that the bit sequence is divided into Z non-overlapping sub blocks, each of which is L bit of length. K_i is the count of the i^{th} type of occurrences in L bit sub

blocks[35, 1]. The calculation of χ^2 value can be expressed mathematically in Equation 5.1.

$$\chi^2 = \frac{2^L}{Z}(\sum_{i=1}^{2^L} {K_i}^2) - K \tag{5.1}$$

Elitism is performed by comparing the fitness value of the fittest chromosome X from the previous generation with the fitness value of the weakest chromosome Y from the current generation, i.e. if $fitness(X) > fitness(Y)$, then Y is replaced by the X in the output mating pool, otherwise Y survives in that generation, the selection is done by using Binary Tournament Selection(BTS) process. BTS is easy to implement and it is also efficient. BTS can avoid the premature convergence, because it gives the chance for selection to each and every chromosome, and that can degrade the convergence[17, 36]. In BTS two chromosomes are selected randomly and after comparing their fitness values, the fittest chromosome survives in the mating pool[18]. The size of the mating pool is n.

The next operation is crossover. Proposed crossover technique on real coded chromosomes is presented in Algorithm 18. This method minimizes the χ^2 value of a bit sequence of size *128* bit for a seed value of a chaotic map by using the objective function 5.1. In this method, two chromosomes are selected from the mating pool and copied to variables randomly. Then a crossover probabilistic value $Cross_{prob}$ is also chosen within the range *0 to 1*. Then this $Cross_{prob}$ is compared with the P_c. If it is greater than the P_c, then these chromosomes are copied directly to the output mating pool, otherwise crossover operation is performed on them. In a crossover operation, first the values of X_0 and μ are subtracted and the absolute results are stored in variables $temp_1$ and $temp_2$ respectively. Then some amount of values, i.e. a and b are taken randomly from $temp_1$ and $temp_2$ within the range *0 to* $temp_1$ and *0 to* $temp_2$ respectively. Then one of the following set of operation is performed over them. The first set of operation is *Addition then Subtraction*, i.e. a and b are added with X_0 and μ of the first chromosome, and a and b are subtracted with X_0 and μ of the second chromosome. The second set of operation is Subtraction then Addition , i.e. a and b are subtracted with X_0 and μ of the first chromosome, and a and b are added with X_0 and μ of the second chromosome. The set of operation is also chosen randomly. The output of this procedure is the offspring, which is then copied to the mating pool.

Algorithm 17 Optimal seed value for chaotic map optimization

Input: n, Initial population
Output: Optimized seed for chaos

begin
for gen= 1 to 200 **do** ▷ Number of Generations
    for each chromosome from the mating pool **do**
        By using chromosome as a seed and the objective function, generate a pseudorandom bit sequence X.
        Compute the fitness value, by applying the Poker test of randomness on X.
    end for
    for i= 1 to n **do** ▷ Elitism operation is done here
        Select the weakest chromosome Y from the current generation.
        Select the fittest chromosome Z from the previous generation.
        Compute the fitness values of Y and Z. Keep the fittest chromosome in the current generation and discard the other one.
    end for
    Create a mating pool of size N by using BTS.
    Perform Crossover operation on the mating pool of real coded chromosomes.
    Perform Mutation operation on the mating pool of real coded chromosomes.
end for
end

After that mutation is executed over the mating pool. Proposed mutation technique on real coded chromosomes is presented in Algorithm 19. In this technique, one chromosome from the mating pool is selected randomly. Then a mutation probabilistic value '$Mute_{prob}$' is chosen randomly within the range 0 to 1. Then $Mute_{prob}$ is compared with the P_m. If it is greater than P_m, then this selected chromosome is copied directly to the output mating pool, otherwise it get mutated. In mutation operation the integer part of the μ is neglected here, thus the X_0 and the fractional part of μ are copied in two separate variables, $temp_1$ and $temp_2$ respectively. Like crossover, here two set of operations are also present and the set of operation is also chosen randomly. In first set of operation X is calculated, where $X = temp2 \times Mute_{prob}$. Then this X is added with $temp_1$ and subtracted with $temp_2$. In second set of

Algorithm 18 Proposed crossover technique on real coded chromosomes

Input: Mating pool, n, P_c

Output: Mating pool contains offspring

1: **begin**
2: **for** i=1 to $n/2$ **do**
3: Select two chromosomes X_1 and X_2 randomly from the mating pool.
4: Copy X_1 and X_2 randomly in two variables $chrom_1$ and $chrom_2$.
5: Select a crossover probability $Cross_{prob}$ within the range *0 to 1*.
6: **if** $Cross_{prob} \leq P_c$ **then**
7: Compute the absolute difference of μ's of $chrom_1$ and $chrom_1$, i.e. $temp1 =\mid \mu_{chrom1} - \mu_{chrom2} \mid$
8: Compute the absolute difference of X_0's of $chrom_1$ and $chrom_2$', i.e. $temp2 =\mid X_{0_{chrom1}} - X_{0_{chrom2}} \mid$
9: Select two random values a and emphb within the range *0 to* $temp_1$ and *0 to* $temp_2$ respectively.
10: Select an integer value *Choice* randomly, within the range *1 to 2*.
11: **if** $Choice = 1$ **then**
12: Add μ with A and X_0 with B of the $chrom_1$
13: Subtract μ with A and X_0 with B of the $chrom_2$
14: **else**
15: Subtract μ with A and X_0 with b of the $chrom_1$
16: Add μ with A and X_0 with B of the $chrom_2$
17: **end if**
18: Boundary restriction is performed here
19: **end if**
20: $chrom_1$ and $chrom_2$ are copied back to their respective positions in the resultant mating pool
21: **end for**
22: **end**

operation X is also calculated, where $X = temp1 \times Mute_{prob}$. Then this X is subtracted with $temp_1$ and added with $temp_2$. Finally the integer part of μ, which was neglected previously is added with current $temp_1$ to get the updated μ. The $temp_2$ is copied to X_0. The resultant seed is the offspring, which is then copied to the resultant mating pool.

Algorithm 19 Proposed mutation technique on real coded chromosomes

Input: Mating pool, n, P_m
Output: Mating pool contains offspring

begin
for i=1 to $n/2$ **do**
    Select one chromosome *chrom* randomly from the mating pool
    if $Mute_{prob} \leq P_m$ **then**
        Copy the X_0 and the fractional part of μ of *chrom* in $temp1$ and $temp2$ respectively
        Select a integer value *Choice* randomly, within the range *1 to 2*
        if $Choice = 1$ **then**
            Take $(Mute_{prob} * 100)\%$ amount of value X from $temp_2$. Add X to $temp_1$ and subtract X from $temp_2$
        else
            Take $(Mute_{prob} * 100)\%$ amount of value X from $temp_1$. Add X to $temp_2$ and subtract X from $temp_1$
        end if
        Boundary restriction is performed here
    end if
    Update μ of *chrom* by adding the previously discarded integer part of μ with $temp_2$. $temp_1$ will be the updated X_0 of *chrom*
    chrom is copied to the resultant mating pool in the same position, where it was during the selection
end for
end

5.3 Optimal pseudorandom bit sequence generation

q bit optimal pseudorandom bit sequence can be generated by using the same objective function, that was used in Section 5.2. The optimal seed value which is the output of the Section 5.2 is used here as a seed for the aforementioned objective function on chaotic map.

Chapter 6

Result and Analysis

The randomness of the optimal pseudorandom bit sequence is tested by using NIST statistical test suit. This test determines the unpredictability of a Pseudo Random Number Generator (PRNG). In recent years NIST becomes the industry norm for the randomness testing and it is the most stringent test suit. It contains 15 nearly independent statistical tests, which is focused on finding all possible types of non-randomness pattern that could exist in a pseudorandom bit sequence. If any type of non-randomness pattern is found by any of this test, then the sequence is declared as non-random, otherwise it declares as random. 15 statistical test of NIST are: The frequency (Monobit) test, Frequency test within a block, The runs test, Tests for the longest run of ones ina block, The binary matrix run test, The Discrete Fourier Transform (spectral) test, The non overlapping template matching test, The overlapping template matching test, Maurer's *Universal Statistical* test, The linear complexity test, The serial test, The approximation entropy test, The Cumulative sums(Cusum) test, The random excursion test, The random excursion variant test[4]. The results of these tests are represented by *P-values*. *P-value* is the probabilistic value. If the output *P-value* of a test is greater than or equal to *0.01*, then PRNG will pass that test for randomness with a *99%* confidence. NIST specified tests are performed to conform the randomness of the result of the proposed algorithm with different chaotic map functions. The results are shown in Table 5-8.

Table 5: NIST test for the proposed method with Logistic chaotic map

Index	**Test Index**	**P-value**	**Result**
1	Frequency	0.9005	Pass
2	Block Frequency	0.5681	Pass
3	Runs	0.1849	Pass
4	Longest-Run-of-Ones	0.4746	Pass
5	Binary Matrix Rank	0.2701	Pass
6	DFT(Spectral)	0.8711	Pass
7	Non Overlapping Template Matching	0.9907	Pass
8	Overlapping Template Matching	0.2610	Pass
9	Maurer's *Universal Statistical*	0.9673	Pass
10	Linear Complexity	0.9856	Pass
11	Serial 1	0.3644	Pass
	Serial 2	0.9231	Pass
12	Approximation Entropy	0.0607	Pass
13	Cumulative Sums (Cusums)	0.9555	Pass
14	Random Excursion		
	-4	0.3965	Random
	-3	0.2284	Random
	-2	0.9058	Random
	-1	0.2670	Random
	1	0.9478	Random
	2	0.5497	Random
	3	0.5385	Random
	4	0.0399	Random
15	Random Excursions Variant		
	-9	0.3965	Random
	-8	0.2284	Random
	-7	0.9058	Random
	-6	0.2670	Random
	-5	0.3965	Random
	-4	0.3965	Random
	-3	0.2284	Random
	-2	0.9058	Random
	-1	0.2670	Random

Continued on next page

Table 5 – *Continued from previous page*

Index	**Test Index**	**P-value**	**Result**
	1	0.9478	Random
	2	0.5497	Random
	3	0.5385	Random
	4	0.0399	Random
	5	0.9478	Random
	6	0.5497	Random
	7	0.5385	Random
	8	0.0399	Random
	9	0.0399	Random

Table 5 represent the *P-value* and *Result* of each test, which is present in NIST test suit. A pseudorandom bit sequence of finite length is the input for each test. This bit sequence is generated by using the proposed method where Logistic map equation is taken into consideration. Obtained *P-value* for a test if greater than or equal to *0.01*, then it can conclude that this proposed method is passed the particular test. For the each sub test of Random Excursion and the Random Excursion Variant test, if $P - value \geq 0.01$, then the sequence will be random, otherwise it will be non-random. From this table it can be clearly observed that all the P-values are greater than or equal *0.01*. So, it can conclude that this proposed method will generate pseudorandom bit sequence by using Logistic chaotic map.

Table 6: NIST test for the proposed method with Skew Tent chaotic map

Index	**Test Index**	**P-value**	**Result**
1	Frequency	0.1336	Pass
2	Block Frequency	0.7178	Pass
3	Runs	0.5949	Pass
4	Longest-Run-of-Ones	0.1767	Pass
5	Binary Matrix Rank	0.2159	Pass
6	DFT(Spectral)	0.8711	Pass
7	Non Overlapping Template Matching	0.2769	Pass

Continued on next page

Table 6 – *Continued from previous page*

Index	**Test Index**	**P-value**	**Result**
8	Overlapping Template Matching	0.9156	Pass
9	Maurers *Universal Statistical*	0.9272	Pass
10	Linear Complexity	0.8685	Pass
11	Serial 1	0.3449	Pass
	Serial 2	0.1249	Pass
12	Approximation Entropy	0.0141	Pass
13	Cumulative Sums (Cusums)	0.8690	Pass
14	Random Excursion		
	-4	0.0166	Random
	-3	0.7985	Random
	-2	0.8221	Random
	-1	0.0572	Random
	1	0.3503	Random
	2	0.4405	Random
	3	0.5228	Random
	4	0.4966	Random
15	Random Excursions Variant		
	-9	0.5211	Random
	-8	0.5259	Random
	-7	0.5294	Random
	-6	0.6485	Random
	-5	0.7055	Random
	-4	0.7210	Random
	-3	0.9326	Random
	-2	0.8273	Random
	-1	0.3447	Random
	1	0.0890	Random
	2	0.2752	Random
	3	0.3525	Random
	4	0.3531	Random
	5	0.3778	Random
	6	0.4250	Random
	7	0.4631	Random
	8	0.4945	Random

Continued on next page

Table 6 – *Continued from previous page*

Index	**Test Index**	**P-value**	**Result**
	9	0.5211	Random

Table 6 shows that all the *P-values* are greater than or equal *0.01*. So, it can conclude that this proposed method will generate Pseudorandom bit sequence by using Skew tent chaotic map.

Table 7: NIST test for the proposed method with Cross Coupled Logistic chaotic map

Index	**Test Index**	**P-value**	**Result**
1	Frequency	0.8231	Pass
2	Block Frequency	0.9665	Pass
3	Runs	0.0903	Pass
4	Longest-Run-of-Ones	0.0799	Pass
5	Binary Matrix Rank	0.1285	Pass
6	DFT(Spectral)	0.1443	Pass
7	Non Overlapping Template Matching	0.6161	Pass
8	Overlapping Template Matching	0.0205	Pass
9	Maurers *Universal Statistical*	0.9616	Pass
10	Linear Complexity	0.4232	Pass
11	Serial 1	0.0338	Pass
	Serial 2	0.7023	Pass
12	Approximation Entropy	0.1435	Pass
13	Cumulative Sums (Cusums)	0.9555	Pass
14	Random Excursion		
	-4	0.1027	Random
	-3	0.1201	Random
	-2	0.3130	Random
	-1	0.7576	Random
	1	0.4629	Random
	2	0.4499	Random
	3	0.5671	Random
	4	0.5281	Random

Continued on next page

Table 7 – *Continued from previous page*

Index	**Test Index**	**P-value**	**Result**
15	Random Excursions Variant		
	-9	0.3960	Random
	-8	0.6056	Random
	-7	0.8352	Random
	-6	0.4739	Random
	-5	0.5320	Random
	-4	0.3447	Random
	-3	0.2404	Random
	-2	0.2199	Random
	-1	0.0801	Random
	1	0.3173	Random
	2	0.5637	Random
	3	0.4674	Random
	4	0.5083	Random
	5	0.5320	Random
	6	0.3657	Random
	7	0.2983	Random
	8	0.3017	Random
	9	0.3320	Random

Table 7 shows that all the *P-values* are greater than or equal *0.01*. So, it can conclude that this proposed method will generate Pseudorandom bit sequence by using Cross coupled Logistic chaotic map.

Table 8: NIST test for the proposed method with Cross Coupled Skew Tent chaotic map

Index	**Test Index**	**P-value**	**Result**
1	Frequency	0.4952	Pass
2	Block Frequency	0.9918	Pass
3	Runs	0.0133	Pass
4	Longest-Run-of-Ones	0.1381	Pass
5	Binary Matrix Rank	0.6420	Pass

Continued on next page

Table 8 – *Continued from previous page*

Index	**Test Index**	**P-value**	**Result**
6	DFT(Spectral)	0.1443	Pass
7	Non Overlapping Template Matching	0.6410	Pass
8	Overlapping Template Matching	0.0528	Pass
9	Maurers *Universal Statistical*	0.9822	Pass
10	Linear Complexity	0.6767	Pass
11	Serial 1	0.0112	Pass
	Serial 2	0.8521	Pass
12	Approximation Entropy	0.2402	Pass
13	Cumulative Sums (Cusums)	0.9642	Pass
14	Random Excursion		
	-4	0.3898	Random
	-3	0.9593	Random
	-2	0.8101	Random
	-1	0.8366	Random
	1	0.7274	Random
	2	0.8700	Random
	3	0.4736	Random
	4	0.9253	Random
15	Random Excursions Variant		
	-9	0.5695	Random
	-8	0.5448	Random
	-7	0.5154	Random
	-6	0.4795	Random
	-5	0.4344	Random
	-4	0.4203	Random
	-3	0.5673	Random
	-2	0.9020	Random
	-1	0.8312	Random
	1	0.6698	Random
	2	0.8055	Random
	3	0.5673	Random
	4	0.8090	Random
	5	0.6698	Random
	6	0.5629	Random
	7	0.5154	Random

Continued on next page

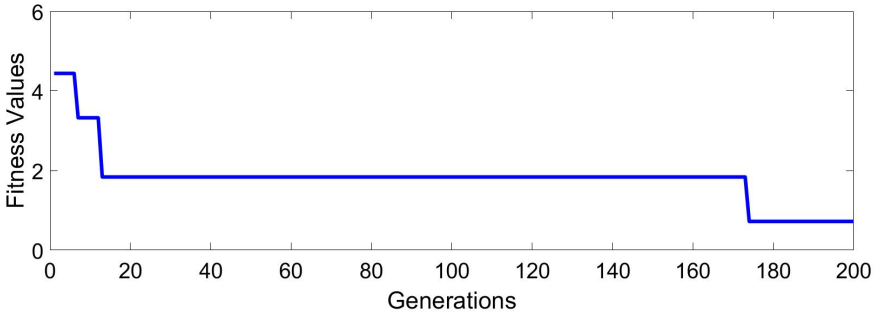

Figure 7: Bar Diagram Showing Minimization of Fitness Values in 200 iteration with logistic chaotic map

Table 8 – *Continued from previous page*

Index	**Test Index**	**P-value**	**Result**
	8	0.5448	Random
	9	0.5695	Random

Table 8 shows that all the *P-values* are greater than or equal *0.01*. So, it can conclude that this proposed method will generate Pseudorandom bit sequence by using Cross coupled skew tent chaotic map.

The generation wise growth in fitness values for *200* generations by using different type of chaotic map functions are shown as a bar diagram in Figure 7-10. The X-axis of the graph represents the number of generations and the Y-axis represent the fitness values of the fittest chromosome in each generation.

From the Figure 7-10 it can be clearly observe that the fitness value minimized after iterating the proposed method with Logistic or Skew Tent or Cross Coupled Logistic or Cross Coupled Skew Tent chaotic map for *200* times.

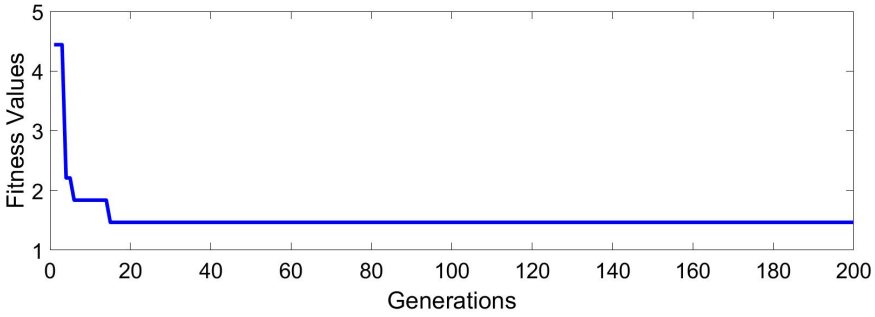

Figure 8: Bar Diagram Showing Minimization of Fitness Values in 200 iterations, with Skew Tent chaotic map

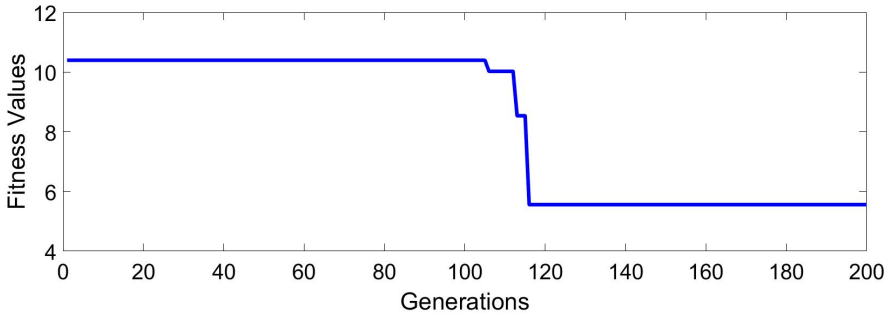

Figure 9: Bar Diagram Showing Minimization of Fitness Values in 200 iterations, with Cross Coupled Logistic chaotic map

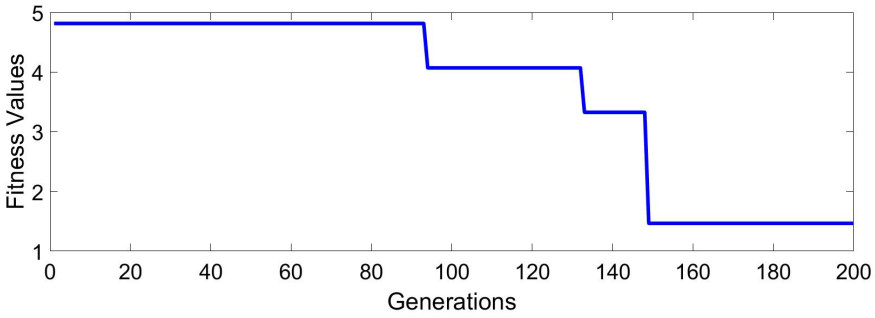

Figure 10: Bar Diagram Showing Minimization of Fitness Values in 200 iterations, with Cross Coupled Skew Tent chaotic map

Chapter 7

Conclusion and Future Scope

This research shown that the optimal seed value for a chaotic map can be optimized by using evolutionary algorithm(RCGA). This optimal seed values can be use to generate the an optimal pseudorandom bit sequence. This paper proposes a RCGA enabled pseudorandom bit sequence generator, which passes all the statistical tests of NIST, by using 4 different type of chaotic maps. So, it can be concluded that the proposed method will generate optimal pseudorandom bit sequence with *99%* confidence. Since the output of this generator is a pseudorandom bit sequence, i.e. it will generate same bit sequence for an optimized seed, so this bit sequence can be used as a key for Cryptographic applications.

This research is focused on finding the optimal seed values for a chaotic map for the generation of an optimal pseudorandom bit sequence. In future, focusing on finding the optimal chaotic map from a set of chaotic maps and its optimal seed value can be made. Multi objective evolutionary algorithms will also going to be used for the finding of the optimal seed values for chaotic maps. In last two decades all works are done to implement True Random Number Generator or TRNG is based on hardware designs. These type of TRNG is costly. A program base TRNG, without using Chaos theory and hardware design is also theoretically possible. In future focusing on finding these type of TRNG can be made.

Bibliography

[1] A.J. Menezes, P.C. Van Oorschot, S.V.: Handbook of Applied Cryptography. CRC Press, Boca Raton, Florida, USA (1996)

[2] Akhshani, A., Akhavan, A., Mobaraki, A., Lim, S.C., Hassan, Z.: Pseudo random number generator based on quantum chaotic map. Communications in Nonlinear Science and Numerical Simulation 19(1), 101–111 (January 2014)

[3] et al., D.B.: A novel secure image steganography method based on chaos theory in spatial domain. International Journal of Security, Privacy and Trust Management (IJSPTM) 3(1), 11–22 (February 2014)

[4] et al., L.E.B.: Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications. National Institute of Standards & Technology (April 2010)

[5] et al., M.B.: Genetic algorithm and its applications to mechanical engineering: A review. 4th International Conference on Materials Processing and Characterization 2(4-5), 2624–2630 (July 2015)

[6] Bao, L., Zhou, Y., Chen, C.L.P., Liu, H.: A new chaotic system for image encryption. In: 2012 International Conference on System Science and Engineering (ICSSE). pp. 69–73 (June 2012)

[7] Barangi, M., Chang, J.S., Mazumder, P.: Straintronics-based true random number generator for high speed and energy-limited applications. IEEE Transactions on Magnetics 52(1) (January 2016)

[8] Bergamo, P., D'Arco, P., Santis, A.D., Kocarev, L.: Security of public-key cryptosystems based on chebyshev polynomials. IEEE Transactions on Circuits and Systems I: Regular Papers 52(7), 1382–1393 (July 2005)

[9] Boeing, G.: Visual analysis of nonlinear dynamical systems: Chaos, fractals, self-similarity and the limits of prediction 4(4), 37–54 (November 2016)

[10] Bucolo, M., Caponetto, R., Fortuna, L., Frasca, M., Rizzo, A.: Does chaos work better than noise? IEEE Circuits and Systems Magazine 2(3), 4–19 (March 2002)

[11] Callegari, S., Rovatti, R., Setti, G.: Embeddable adc-based true random number generator for cryptographic applications exploiting nonlinear signal processing and chaos. IEEE Transactions on Signal Processing 53(2), 793–805 (Feb 2005)

[12] Caponetto, R., Fortuna, L., Fazzino, S., Xibilia, M.G.: Chaotic sequences to improve the performance of evolutionary algorithms. IEEE Transactions on Evolutionary Computation 7(3), 289–304 (June 2003)

[13] Chen, J., Zhou, J., Wong, K.W.: A modified chaos-based joint compression and encryption scheme. IEEE Transactions on Circuits and Systems II: Express Briefs 58(2), 110–114 (Feb 2011)

[14] Chen, S.L., Hwang, T., Lin, W.W.: Randomness enhancement using digitalized modified logistic map. IEEE Transactions on Circuits and Systems II: Express Briefs 57(12), 996–1000 (Dec 2010)

[15] Coelho, L.S., Mariani, V.C.: Combining of chaotic differential evolution and quadratic programming for economic dispatch optimization with valve-point effect. IEEE Transactions on Power Systems 21(2), 989–996 (May 2006)

[16] Dabal, P., Pelka, R.: A chaos-based pseudo-random bit generator implemented in fpga device. In: 14th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems. pp. 151–154 (April 2011)

[17] David E. Goldberg, K.D.: A comparative analysis of selection schemes used in genetic algorithms. Foundations of Genetic Algorithms 1, 69–93 (1991)

[18] D.E., G.: Genetic algorithm in search, optimization and machine learning. Addison- Wesley (1989)

[19] Enayatifar, R., Abdullah, A.H., FauziIsnin, I.: Chaos-based image encryption using a hybrid genetic algorithm and a dna sequence. Optics and Lasers in Engineering 56, 83–93 (May 2014)

[20] García-Martínez, M., Campos-Cantón, E.: Pseudo-random bit generator based on multi-modal maps. Nonlinear Dynamics 82(4), 2119–2131 (December 2015)

[21] Hasler, M., Maistrenko, Y.L.: An introduction to the synchronization of chaotic systems: coupled skew tent maps. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications 44(10), 856–866 (October 1997)

[22] Heidari-Bateni, G., McGillem, C.D.: A chaotic direct-sequence spread-spectrum communication system. IEEE Transactions on Communications 42(234), 1524–1527 (Feb 1994)

[23] Hu, H., Liu, L., Ding, N.: Pseudorandom sequence generator based on the chen chaotic system. computer physics communications. Computer Physics Communications 184(3), 765–768 (March 2013)

[24] Kocarev, L., Jakimoski, G.: Pseudorandom bits generated by chaotic maps. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications 50(1), 123–126 (Jan 2003)

[25] Kohda, T., Tsuneda, A.: Statistics of chaotic binary sequences. IEEE Transactions on Information Theory 43(1), 104–112 (Jan 1997)

[26] Liu, H., Zhu, Z., Jiang, H., Wang, B.: A novel image encryption algorithm based on improved 3d chaotic cat map. In: 2008 The 9th International Conference for Young Computer Scientists. pp. 3016–3021 (Nov 2008)

[27] m. Liu, J., s. Qiu, S., Xiang, F., j. Xiao, H.: A cryptosystem based on multiple chaotic maps. In: 2008 International Symposiums on Information Processing. pp. 740–743 (May 2008)

[28] Liu, L., Miao, S., Cheng, M., Gao, X.: A pseudorandom bit generator based on new multi-delayed chebyshev map. Information Processing Letters 116(11), 674–681 (November 2016)

[29] Lorenz, E.N.: The Essence of Chaos. CRC Press, 3 edn. (1995)

[30] Luca, A., Ilyas, A., Vlad, A.: Generating random binary sequences using tent map. In: ISSCS 2011 - International Symposium on Signals, Circuits and Systems. pp. 1–4 (June 2011)

[31] M. François, T. Grosges a, D.B.R.E.: Pseudo-random number generator based on mixing of three chaotic maps. Communications in Nonlinear Science and Numerical Simulation 19(4), 887–895 (April 2014)

[32] Marsaglia, G.: Diehard statistical tests. Journal of Statistical Software (1995), `https://stat.fsu.edu/pub/diehard/`

[33] Mukhopadhyay, S., Mandal, J.K.: Denoising of digital images through pso based pixel classification. Central European Journal of Computer Science, Springer Vienna 3(4), 158–172 (2013)

[34] Mukhopadhyay, S., Mandal, J.K.: A fuzzy switching median filter of impulses in digital imagery (fsmf). Circuits System Signal Processing 33(7), 2193–2216 (Jul 2014), `http://dx.doi.org/10.1007/s00034-014-9739-z`

[35] Narendra K Pareek, V.P., Sud, K.K.: A random bit generator using chaotic maps. International Journal of Network Security 10(1), 32–38 (January 2010)

[36] Noraini Mohd Razali, J.G.: Genetic algorithm performance with different selection strategies in solving tsp. Proceedings of the World Congress on Engineering, IAENG II, 1134–1139 (July 2011)

[37] Paral, P., Dasgupta, T., Bhattacharya, S.: Colour image encryption based on cross-coupled chaotic map and fractional order chaotic systems. In: 2014 International Conference on Communication and Signal Processing. pp. 1947–1952 (2014)

[38] Pastijn, H.: Chaotic Growth with the Logistic Model of P.-F. Verhulst. In: Ausloos M. Springer, Berlin, Heidelberg (June 2006)

[39] Sanjib Ganguly, D.S.: Distributed generation allocation on radial distribution networks under uncertainties of load and generation using genetic algorithm. IEEE Transactions on Sustainable Energy 6(3), 688–697 (July 2015)

[40] Sattar B. Sadkhan, R.S.M.: Proposed random unified chaotic map as prbg for voice encryption in wireless communication. International Conference on Communication, Management and Information Technology 65(6), 314–323 (September 2015)

[41] Singh, S., Siddiqui, T.J., Singh, R., Singh, H.V.: Dct-domain robust data hiding using chaotic sequence. In: 2011 International Conference on Multimedia, Signal Processing and Communication Technologies. pp. 300–303 (Dec 2011)

[42] Socek, D., Li, S., Magliveras, S.S., Furht, B.: Short paper: Enhanced 1-d chaotic key-based algorithm for image encryption. In: First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05). pp. 406–407 (Sept 2005)

[43] Somnath Mukhopadhyay, Tamal Datta Chaudhuri, J.K.M.: A hybrid pso-fuzzy based algorithm for clustering indian stock market data. (eds) Computational Intelligence, Communications, and Business Analytics. CICBA 2017. Communications in Computer and Information Science 776, 475–487 (September 2017)

[44] Stojanovski, T., Kocarev, L.: Chaos-based random number generators-part i: analysis [cryptography]. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications 48(3), 281–288 (Mar 2001)

[45] Stojanovski, T., Pihl, J., Kocarev, L.: Chaos-based random number generators. part ii: practical realization. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications 48(3), 382–385 (Mar 2001)

[46] Stoyanov, B., Kordov, K.: Novel secure pseudo-random number generation scheme based on two tinkerbell maps. Advanced Studies in Theoretical Physics 9(9), 411 – 421 (September 2015)

[47] Strogatz, S.H.: Nonlinear Dynamics and Chaos With Applications to Physics, Biology, Chemistry, and Engineering. CRC Press, Boca Raton, Florida, USA, 2 edn. (July 2014)

[48] Vinod Patidar, K. K. Sud, N.K.P.: A pseudo random bit generator based on chaotic logistic map and it's statistical testing. Informatica 33(4), 441–452 (May 2009)

[49] Wang, L., Ye, Q., Xiao, Y., Zou, Y., Zhang, B.: An image encryption scheme based on cross chaotic map. In: 2008 Congress on Image and Signal Processing. vol. 3, pp. 22–26 (May 2008)

[50] Wang, Y., Ren, G., Jiang, J., Zhang, J., Sun, L.: Image encryption method based on chaotic map. In: 2007 2nd IEEE Conference on Industrial Electronics and Applications. pp. 2558–2560 (May 2007)

[51] Wong, K.W., Lin, Q., Chen, J.: Simultaneous arithmetic coding and encryption using chaotic maps. IEEE Transactions on Circuits and Systems II: Express Briefs 57(2), 146–150 (Feb 2010)

[52] Yi, X.: Hash function based on chaotic tent maps. IEEE Transactions on Circuits and Systems II: Express Briefs 52(6), 354–357 (June 2005)