A NEW LIGHTWEIGHT STATISTICAL RANDOMNESS TEST SUITE AND ITS EVALUATION BY COMPARISON WITH OTHER TEST SUITES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ZİYA AKCENGİZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
CRYPTOGRAPHY

AUGUST 2021

Approval of the thesis:

A NEW LIGHTWEIGHT STATISTICAL RANDOMNESS TEST SUITE AND ITS EVALUATION BY COMPARISON WITH OTHER TEST SUITES

submitted by **ZİYA AKCENGİZ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Cryptography Department, Middle East Technical University** by,

Prof. Dr. Ayşe Sevtap Kestel
Director, Graduate School of **Applied Mathematics**

Prof. Dr. Ferruh Özbudak
Head of Department, **Cryptography**

Assoc. Prof. Dr. Ali Doğanaksoy
Supervisor, **Department of Mathematics, METU**

Assoc. Prof. Dr. Fatih Sulak
Co-supervisor, **Department of Mathematics, Atilim University**

Examining Committee Members:

Prof. Dr. Ali Aydın Selçuk
Department of Computer Engineering, TOBB ETU

Assoc. Prof. Dr. Ali Doğanaksoy
Department of Mathematics, METU

Assoc. Prof. Dr. Murat Cenk
Institute of Applied Mathematics, METU

Assoc. Prof. Dr. Zülfükar Saygı
Department of Mathematics, TOBB ETU

Assoc. Prof. Dr. Oğuz Yayla
Institute of Applied Mathematics, METU

Date:

ABSTRACT

A NEW LIGHTWEIGHT STATISTICAL RANDOMNESS TEST SUITE AND ITS EVALUATION BY COMPARISON WITH OTHER TEST SUITES

AKCENGİZ, ZİYA
Ph.D., Department of Cryptography
Supervisor : Assoc. Prof. Dr. Ali Doğanaksoy
Co-Supervisor : Assoc. Prof. Dr. Fatih Sulak

August 2021, 64 pages

Playing rolling dice or toss a coin is fair or not depends on whether the material played is fair or not. Generating a random number is equivalent to both dice and coin game. Random numbers have a wide usage area. Hence generating a random number is very important, and it should be fair. In other words, the generator should not have any bias. Using statistical randomness tests, we can determine whether a generator generates random numbers or not, that is, whether the generated numbers follow a pattern or not. In the literature, there are many statistical randomness tests. Some of these tests were selected, and they form a test suite. While defining a randomness test suit, the suites in the literature, the features of the sequences are not considered. In this thesis, a test suite is proposed to test a long sequence. Some tests in the literature are already in a suitable format to be applied to long sequences. In addition, some approximations have been used to apply those tests designed for relatively shorter sequences to long sequences, or the sequences have been manipulated using to make them suitable for those tests while ignoring whether manipulations damage the structure of sequence or not. It has been evaluated that the test suites in the literature using such manipulations to long sequences may give incomplete results. In this thesis, the appropriate tests in the literature are modified into long sequence tests using three different methods, and new long sequence tests are proposed. Sequences generated from true and pseudo-random number generators are tested with the proposed test suite. By collecting proposed tests together according to their time performance, super lightweight and lightweight test suites are proposed. The mutual correlation of the tests with each other is evaluated. Sensitivities to bias sequences are tested and compared with tests in the literature.

Keywords: Randomness, Random Number, Long Sequences, Stream Cipher, Cryptography, Randomness Tests, Test Suites

ÖZ

UZUN DİZİLER İÇİN YENİ BİR HAFİF SIKLET İSTATİSTİKSEL RASTGELELİK TEST PAKETİ VE DİĞER TEST PAKETLERİYLE DEĞERLENDİRMELİ OLARAK KARŞILAŞTIRILMASI

AKCENGİZ, ZİYA
Doktora, Kriptografi Bölümü
Tez Yöneticisi : Doç. Dr. Ali Doğanaksoy
Ortak Tez Yöneticisi : Doç. Dr. Fatih Sulak

Ağustos 2021, 64 sayfa

Bir zar atmanın veya yazı tura oynanın adil olup olmadığı zara veya paraya bağlıdır. Rastgele bir sayı üretmek zar atma ve yazı tura oyununa mantık olarak eşittir. Rastgele sayılar kullanım alanı oldukça geniştir, bundan dolayı rastgele sayı üretimi önelidir ve üretimin adil olması gereklidir, yani üretilen sayıların bir yönelimi olmamalıdır. İstatistiksel rastgelelik testleri kullanılarak üretecin ürettiği sayıların rastgele olup olmadığını yani üretilen sayıların belli bir döngü izleyip izlemediğini belirleyebiliriz. Literatürde, bir çok istatistiksel rastgelelik testi vardır. Bu testlerden bazıları seçilerek bir test paketi haline getirilmiştir. Literatürde ki test paketleri oluşturulurken, test edecekleri dizilerin özellikleri dikkate alınmamıştır. Bu tez de uzun dizileri test edecek bir test paketi önerildi. Literatürde ki bazı testler uzun dizileri test etmek için uygun değildir. Ek olarak, diğer testleri de uzun diziler uygulamak için bazı yaklaşımlar kullanılmıştır veya dizi üzerinde bazı metodlarla oynanarak test edilebilecek duruma getirilmiştir. Literatürde kullanılan metodlarda dizinin yapısında bir kayıp olup olmadığı gözardı edilmiştir. Literatürdeki test paketlerinin bundan dolayı yeterli sonuç veremeyeceği değerlendirilmiştir. Bu tezde, seçilen uygun testler üç farklı metod kullanılarak uzun dizi testlerine dönüştürüldü ve bazı metodlar için yeni uzun dizi testleri önerildi. Önerilen test paketi ile gerçek ve sözde rastgele sayı üreticiyle üretilmiş uzun diziler test edildi. Önerilen testler zaman performanslarına göre bir araya getirilerek süper hafif ve hafif test takımları önerilmiştir. Testlerin ikili ilişkileri değerlendirildi. Testlerin eğilimli dizilere karşı duyarlılıkları belirlendi ve literatürde yer alan diğer testlerle karşılaştırıldı.

Anahtar Kelimeler: Rastgelelik, Rastgele Sayılar, Uzun Diziler, Akan şifreler, Kriptografi, Rastgelelik Testleri, Test Paketleri

To my family

ACKNOWLEDGMENTS

First of all, I would like to gratefully thank my supervisor, Assoc. Prof. Ali DOĞANAKSOY not only for his support, guidance and motivation that make my thesis start and finish but also for being the main factor that makes me choose Cryptography.

I would like to express my deep thanks to my co-supervisor, Assoc. Prof. Dr. Fatih SULAK and Dr. Muhiddin UĞUZ, although his name does not express as a supervisor, for their support and contributions to my thesis with their great experiences and knowledge.

I am very grateful to Batuhan AVLAYAN for his unrequited help, support and, most importantly being a good friend.

I would like to thank my examining committee members, for their evaluations, advices and contributions.

I would like to thank Onur KOÇAK, Okan ŞEKER, Murat Burhan İLTER and Erkan USLU, who provided motivation and support for a long time with their friendships, even though we could not work together for Ph.D.

I thank my friends from TUBITAK-UEKAE and friends from FAME-CRYPT for their support and friendships.

I greatly thank my father and my mother for their support to take the best education during my life and for their love and encouragement.

I greatly appreciate my wife, Ezgi ORBAY AKCENGİZ, for her endless support and help while writing my thesis, but most importantly, for her love for me. I would like to thank my son Timuçin AKCENGİZ, who has been the source of endless energy and joy in my house throughout this stressful process. At last, I thank my second son, Teoman AKCENGİZ, who will be with us soon and will be miss my endless student career, maybe the give the last big motivation to end this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

S	n bit binary sequence
S'	derivative sequence of S
$\tilde{S}$	integer sequence of S
C_k	Auto Correlation
NIST	National Institute of Standards and Technology
$E(X) = \mu$	Expected Value
$V(X) = \sigma^2$	Variance
LC	Linear Complexity
ICP	Index Coincidence Point
AES	Advance Encryption Standard
ECB	Electronic Codebook
CBC	Cipher Block Chaining
F.L	Fixed Length Partitioning
Dyn	Dynamic Partitioning

CHAPTER 1

INTRODUCTION

Random sequences can be described as a set of symbols without any order or pattern. They are used in many areas of science such as biology, statistics, finance, information science, and mathematics. Moreover, random sequences are used commonly in cryptographic algorithms such as encryption keys, nonces, salts.

When we talk about a ‘random number’ or a ‘number sequence,’ we mean a ‘number’ or the ‘sequence’ generated by a random process. Random processes which generate a sequence with terms from the set $T = \{0, 1, 2, \ldots, n\}$ possess certain statistical properties such as

- Terms of sequence should be generated independently, that is, any information about the first k term should not provide any information to predict the $(k+1)^{st}$ term,
- Terms should be generated identically,
- In most cases uniform distribution of elements of T is also required. In particular, if the elements are taken from the set $T = \{0, 1\}$ sequence is called a binary sequence. Then, it is expected that each element of $\sigma \in T$ occurs with probability $\frac{1}{2}$, called balancedness.

Random sequences are generated by Truly Random Number Generators (TRNG), which extract randomness from physical phenomena such as atmospheric noise, movement of an electron, heat measurements, Etc. [39]. Random number generators depending on quantum events are called Quantum Random Number Generator [17].

These types of generators are sub-case of TRNGs. However, TRNGs are not useful for cryptographic purposes in most cases because it is necessary to provide the same environment to reproduce a generated sequence. Moreover, transmission and storage of the data produced by TRNGs have a cost. Another type of random number generator is Pseudo-Random Number Generators (PRNG). The structure of PRNGs depends on some deterministic mathematical algorithms. Since they rely on deterministic algorithms, sequences generated are generally periodic and depend on the seed, so by definition, PRNGs are not random processes. However, if the PRNG is cryptographically safe, in other words, without knowing the seed that is nearly impossible to guess the sequence, they are used for cryptographic algorithms. It is vital to test the sequences generated by PRNGs for being indistinguishable from true random number sequences. If PRNGs do not satisfy randomness criteria, they reduce the strength and security of the cryptosystem in which they are used so that system can be cryptanalyzed. Some of the attacks on PRNGs are given in [20]. Therefore, PRNGs and cryptographic algorithms should be designed well so that outputs should be indistinguishable from an output of a TRNG. Although it is impossible to say numbers are theoretically random, in general, the testing process is done statistically and based on some combinatorial aspects.

Deciding the pseudo-randomness of an infinite sequence is a difficult task. Golomb's Postulates are one of the good examples for this purpose [11]. These postulates are among the most significant attempts to create some necessary properties for a finite (or periodic) sequence to be pseudo-random. Sequences satisfying the following three properties are called pseudo-random sequences.

Let $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)^\infty$ be an infinite binary sequence periodic with some integer n. A *run* is defined as an uninterrupted maximal subsequence of identical bits. Runs of 0's are called gaps, and runs of 1's are called blocks. The following R1, R2, and R3 are Golomb's randomness postulates:

(R1) In a period of σ, the number of 1's should differ from the number of 0's by at most 1. In other words, the sequence should be balanced.

(R2) In a period of σ, at least half of the total number of runs of 0's or 1's should have length one, at least one-fourth should have length 2, at least one-eighth should have length 3, and so on. Moreover, for each of these lengths, there should be (almost) equal gaps and blocks.

(R3) The auto-correlation function C_k should be two-valued. That is, for some integer K and for all $k = 0, 1, 2, \dots, n-1$,

$$C_k = \sum_{j=0}^{n-1} (-1)^{s_j + s_{j+k}} = \begin{cases} n & \text{if} \quad k = 0 \\ K & \text{where} \quad k \neq 0 \end{cases} . \tag{1.1}$$

Although these postulates are essential for the philosophy of randomness, they are too strict and valid only for periodic infinite sequences. A non-periodic finite partition of an infinite random sequence may not provide these postulates. In the literature, rather than Golomb's postulates, statistical approaches are studied to measure whether the finite sequence is random with a certain probability. Statistical randomness tests use different random variables, mainly inspired by these postulates [2, 7, 10, 13, 14, 16, 18, 30, 35, 41, 42, 43, 48].

Statistical tests can be published individually or as a suite. Kendall and Smith suggest one of the oldest test suites [21], which involves four basic tests, named *frequency, serial, poker and gap test*. In years many test suites were proposed, but Donald Knuth in [22] suggested the idea of most of these statistical suites. He suggested ten statistical tests named *birthday spacings, collision, coupon collector's, frequency, gap, maximum of t, permutation, poker, run and serial*. Later, Koçak suggested some modifications to these tests in [24].

P. L'Ecuyer investigated some statistical tests and applied them in [26]. Then, P. L'Ecuyer and Simard defined a test suite which is the suite of test batteries [25].

Marsaglia defined some basic tests in [28]. Then, Marsaglia introduced test suite DIEHARD in [27]. In this test suite, there are twelve statistical randomness tests. Later, Brown introduced the DIEHARDER test suite, including all tests in DIEHARD and ten more tests [4].

Rukhin defined a test suite including eleven statistical tests in [33]. Then, NIST proposed a test suite using these tests and some additional tests [34]. NIST test suite involves fifteen tests. For testing the randomness of AES candidates, Soto [37] uses this suite. In years, corrections in some tests proposed in this test suite were done [6, 12]. Since it is a well-known test suite, some algorithm improvements were made to apply this suite faster [45].

Walker proposed the test suite ENT, including five tests [49] and Srinivasanet proposed a test suite to test problems in parallel implementations of PRNGs [38]. Although generated sequences can be long or short, none of these tests suites the length of the sequences was disregarded.

In the literature, test suites are designed for testing without considering the length of the sequences. For testing short sequences, especially for testing the outputs of block ciphers or hash functions, Sulak proposed a new approach for randomness tests [40]. Sulak proposed a test suite for the short sequences, which give better results and do not need concatenating short outputs. On the other hand, Koçak proposed a test suite that includes tests from the literature with a mathematical background and is essential for determining randomness [23]. Koçak gave the distribution functions and necessary recursions to compute actual probability calculations. Both test suites are designed to test short sequences, and the given actual probability calculations work only for short sequences.

In the literature, for testing short sequences, probabilities are computed either by doing exact calculations or use recursions. For testing long sequences, either good approaches can be made, for example, normal distribution for weight test, these approaches will be used directly without changing the structure of the sequence, or the sequence is partitioned into short subsequences to which we can evaluate test statistics. For the testing bundle of subsequences, for each of them, random-variable of tests, named *t-values* are computed. Then using statistical approximations, usually, χ^2 goodness of fit test sequences are tested. For using these methods, the probability distribution function or necessary recursions should be computed. However, this type of partitioning may cause distortion. In the literature, one of the standard methods for testing long sequences is fixed length partition.

According to Soto [36], independence and coverage are important issues. Turan et al. observed that some tests for short sequences on the NIST test suite are correlated [46]. Doğanaksoy et al. and Sulak et al. observed those correlations and dependencies in

NIST test suite [1, 8, 44]. In addition to these, there are many studies in the literature about correlations, independence, and coverage of statistical randomness tests [9, 15, 19].

In this thesis, we adopt the method of dynamic partition depending on the random variable. In this method, while testing a long sequence, no bit of the sequence is wasted, and partitioning depends on random variables. This approach is constructed by inspiring the collision estimate method, which is one of the entropy estimation methods recommended in the NIST [47]. By classifying the tests suitable for the three proposed methods, each test result for the appropriate method is obtained. In addition, if the test is appropriate, results are obtained in other methods, and comparisons are made. Finally, a lightweight and a super lightweight test suite that can be applied to long sequences are proposed. In addition, we evaluate the mutual correlations of the tests in the proposed test suite by using Pearson correlation [32]. Moreover, the sensitivity of the tests to bias sequences is examined. Based on the data obtained, evaluations are made on how the test suite should be created. In addition, comparisons are made if the test has a version in the NIST test suite.

The content of this thesis is briefly processed according to the chapters as follows. In the first chapter, a brief introduction is made, and preliminaries that should be known are given. In the second chapter, methods and definitions of randomness tests that are used are presented. In the third chapter, applications and mutual correlations of defined tests are given. In the fourth chapter, sensitivities and results of the tests comparisons with NIST test suite are given. Finally, the conclusion is made, and future works are mentioned.

1.1 Preliminaries

1.1.1 Preliminary Statistics

In this thesis, three methods are proposed to apply randomness tests to long sequences. It is essential to know some statistical terms, inequalities, and distributions to apply these methods to statistical randomness tests. In this section, the explanations of mean, variance, normal distribution, and chi-square goodness fit test are given.

1.1.1.1 Mean and Variance

Mean, or expected value, is the average value that the probability distribution of the random variable should be taken, denoted by μ or $E(X)$, where X is a random variable.

For a discrete random variable X,

$$E(X) = \mu = \sum xp(x) \tag{1.2}$$

where $p(x) = probability\ of\ x\ occurrence.$

For a continuous random variable X,

$$E(X) = \mu = \int xf(x)\,dx. \tag{1.3}$$

where $f(x) = probability\ distribution\ function\ of\ X.$

Variance is the expected squared deviation of a random variable from its mean, denoted by σ^2 or $V(X)$.

$$V(X) = \sigma^2 = E(X^2) - E(X)^2. \tag{1.4}$$

1.1.1.2 Normal Distribution

Many phenomena' probability distributions depend on the normal distribution. The normal distribution is a symmetric distribution depending on mean and variance. If the mean is 0 and variance is 1, then the distribution is called the standard normal distribution.

Probability density function of normal distribution is:

$$N_{\mu,\sigma^2}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \tag{1.5}$$

1.1.1.3 Chi Square Goodness of Fit Test

The Chi-square goodness of fit test is one of the most effective statistical techniques used to examine how much a sample taken from a set reflects the entire set. When using this test, the exact distribution of the random variable is calculated and divided into bins. The values in the sample taken from the set are put in the appropriate bin, and the number of elements contained in each bin is found. Finally, the result of the test is obtained by performing the following procedure:

$$\chi^2(X) = \sum_{i=1}^{n} \frac{(Obs(X_i) - Exp(X_i)^2}{Exp(X_i)} \tag{1.6}$$

1.1.1.4 Complementary Error Function and Incomplete Gamma Function

In this thesis, the probability distribution functions of proposed tests approach two different distributions. One of them is the normal distribution. To compute tail probability of normal distribution *complementary error function*, $erfc(z)$, is used, which gives *p-value* of test.

$$erfc(z) = 1 - \frac{2}{\pi} \int_{z}^{\infty} e^{-t^2}\,dt \tag{1.7}$$

The second type approximation is χ^2 goodness of fit test. To use this approximation, one should use χ^2 distribution. The get test results which are used chi^2 distribution *incomplete gamma function*, $igamc(\chi^2, dof)$ (dof=degree of freedom), is used which result give the *p-value* of test. To compute $igamc$ we need to compute gamma function, $\Gamma(z)$.

$$\Gamma(z) = \int_{0}^{\infty} t^{z-1} e^{-t}\,dt \tag{1.8}$$

$$igamc(\chi^2, dof) = \frac{1}{\Gamma(\chi^2)} \int_{dof}^{\infty} e^{-t} t^{a-1}\,dt \tag{1.9}$$

1.1.1.5 Hypothesis Testing, Error Types and Significance Level

While performing statistical tests, the hypothesis that the number generator is random is tested. The null hypothesis, H_0, is number generator is random. The alternative hypothesis H_a is number generator is not random. For each case, there exists an error type given in Table 1.1 [34].

Table 1.1: Hypothesis Testing

Random sources	Accept H_0	Accept H_a
Generator random	Correct	Type-I error
Generator non-random	Type-II error	Correct

A predetermined value α called significance level is used for test Type-II error used for randomness testing. Each statistical test is checked whether the test result called *p-value* is greater than α or not. If the result is correct, then H_0 is accepted. Otherwise, the random number generator is non-random, so that H_0 is rejected.

CHAPTER 2

RANDOMNESS TESTS AND PARTITIONING METHODS

In this chapter, methods and randomness tests used in this thesis are defined, and their applications are explained. In the literature, there are lots of statistical randomness test to determine a sequence or a bundle of sequences are random or not. Some of these tests are given in Table 2.1. In the literature, tests statistics that test can be applied long sequences not explained well. This type of test either uses some simulation results or some approximation without giving any detailed explanation. In this thesis, we evaluate long sequences' randomness with detailed calculations and propose a new lightweight test suite for testing long sequences.

Table 2.1: Statistical Tests for Long Sequences

TEST NAME	LENGTH	TEST NAME	LENGTH
Test for the Longest Run of Ones in a Block[34]	750000	Non-Overlapping Template Matching[34]	10^6
Binary Matrix Rank Test[27]	38912	Overlapping Template Matching Test[34]	10^6
Maurer's "Universal Statistical" Test[30]	≥ 387840	Lempel-Ziv Compression Test [34]	10^6
Linear Complexity Test[34]	10^6	Random Excursions Test[34]	10^6
Random Excursion Variant Test[34]	10^6	The birthday spacings test[27]	2^{26}
The overlapping 5-permutation test[27]	10^6	The bitstream test[27]	$2^{21} + 19$
The tests OPSO, OQSO and DNA[27]	2^{21}	The craps test[27]	200000

It is not possible to decide whether a sequence is truly random or not. However, using statistical properties can decide whether the sequence looks random with a certain probability. A randomness test is defined in three steps. The first step is defining a random variable $X : \Omega \to \mathbb{R}$, where Ω is the set of sequences. Then, obtaining its probability distribution function $F(X)$. The second step is computing $X(S) = t_s$ where $S \in \Omega$ is the sequence under consideration and t_s is the count of random variable. The last step is evaluating the sequence.

Let α be the predefined significance level of null hypothesis H_0 then statistical randomness test is the result of probability of $X(\S) = t_p$ is defined as follows,

$$Prob(X(\sigma) = t_s) \begin{cases} \leq \alpha & \sigma \quad \text{is not random} \\ > \alpha & \sigma \quad \text{pass the test.} \end{cases} \tag{2.1}$$

When the sequence is reasonably short, computing $X(S)$ and $Prob(X(S = t_s)$ can be easy, when the length of sequence gets longer, computing $Prob(X(S) = t_s) = F(t)$ is generally difficult unless it is impossible. In the literature, to overcome this problem, statistical properties are used. For computing $F(t)$, some good approaches should be found. For example, for the weight test, computing exact value of $Prob((w(S) = k) = \frac{1}{2^k}\binom{n}{k}$ directly is easy until length of the sequence n is about 1000. However, when n gets larger, this computation will not work. Since, in most cases, direct computation is infeasible for large values, explicit expression of F is not enough. A feasible way of computing $F(t)$ is using recursions which is an advantageous method for reasonably short sequences [23]. However, for the length of the sequence to get larger, recursion cannot be computed. For this reason, an accurate approximation of statistics of the random variables should be found to compute test results. For the weight test, when n is large enough, F approaches the normal distribution. Although some tests using approximations give results, like weight and run [34], there is no accurate approximation for many tests used for cryptographic purposes. The third method is testing a collection of short subsequences of the sequence since all computations are much easier for short sequences. In Table 2.2, the methods to be used for testing sequences of different lengths are given. If the length of the sequence is short, then exact computation can be done. Since exact computations can be done, we do not need recursions and approximation, and partitioning methods are not feasible. If the length of the sequence is reasonably long, then exact computation can not be done quickly. Because of this reason, recursions can be used. Since recursion gives the exact results for this type of sequence, it is unnecessary to use approximations and partitioning methods. However, if the sequence is long enough, there are only two options: approximation or partitioning methods.

Let $S_1, S_2, \ldots S_N$ be N overlapping or non-overlapping parts of S with an order. Then, $X(S_1), X(S_2), \ldots, X(S_N)$ are *t-value* of statistical randomness test depending

Table 2.2: Methods according to length of sequence

	exact	recursion	approximation	partition
short	✓	no need	X	X
reasonably long	not feasible	✓	no need	no need
long	X	X	✓	✓

on random variable of X. Then, distribution values of X over Ω_N and distribution values of X over $\{S_1, S_2, \ldots, S_N\}$ should be similar by means of χ^2 distribution.

Definition 2.0.1. Let $S = s_1, s_2, \ldots, s_n$ and $S_1 = s_1, s_2, \ldots, s_{t_1}$, $S_2 = s_{t_1+1}, s_{t_1+2}$ $\ldots, S_{t_1+t_2}, \ldots S_N = s_{t_{N-1}+1}, s_{t_{N-1}+2}, \ldots, s_n$ be a partition of S. If $t_i = t_j$ for all $i, j = 1, 2, \ldots N-1$, then length of all subsequences are equal and it is called *fixed length partition* otherwise *variable length partition*.

If the given partition is the fixed length, then the χ^2 distribution can be used directly. Otherwise, the given partition depends on a random variable. Let $\xi \in X(S)$ and $l_0 \in \mathbb{Z}^+$ be fixed. We define a new random variable $X_\xi : \Omega \mapsto \mathbb{Z}^+$ as

$$X_\xi(\sigma_t) \mapsto \begin{cases} l_0 & \text{if} \quad X(s_{b(t)}, \ldots, s_{b(t)+j} \neq \xi \quad for \quad j = 0, \ldots, l_0 - 1 \\ 1 + min_j\{j | X(s_{b(t)}, \ldots s_{b(t)+j} \ldots s_{b(t)+j}) = \xi\} & \text{otherwise} \end{cases} \tag{2.2}$$

where $b(t)$ is defined by setting $b(1) = 1$ and $b(t+1) = b(t) + X_\xi(S_t)$ for $t > 1$. We call the subsequence $S_t = s_{b(t)}, \ldots, s_{b(t)+1}$ the t^{th} part of S.

In this thesis, tests are classified and defined in three methods: applied to the entire sequence, fixed-length, and dynamically partitioned subsequences. In Table 2.3, we give the statistical randomness tests used in this thesis and relative methods that can be applied. Not all methods are suitable for all tests. There are different reasons for this. For the entire sequence testing method, test statistic does not approach a suitable distribution, or random variable does not work well and does not test every term of a long sequence. For the fixed-length partitioning method, random variables may not test the last term of subsequences. A random variable may not occur for the dynamic partitioning method, and the partitioning method will not work well. Because of these reasons, we do not recommend each test for each method.

Table 2.3: Statistical Randomness Tests and Appropriate Methods

TEST NAME	ENTIRE SEQUENCE	FIXED LENGTH	DYNAMICAL
Weight	✓	✓	✓
Total Run	✓	✓	✓
R2 Run	✓	X	X
Linear Complexity	X	✓	✓
Integer Coverage	X	✓	X
Integer Collision Index	X	X	✓
Integer Collision Distance	X	X	✓
Integer Saturation	X	X	✓
Index Coincidence Point	X	X	✓

2.1 Partition Methods

In this section, partitioning methods and their applications are explained. This thesis proposes a lightweight test suite to test long sequences without destroying their structure to avoid losing distortions. From the properties given in Chapter 1, defined tests should be simple and examine the entire sequence. Three different methods are used in this thesis: *the entire sequence testing method, fixed-length partition method, and dynamic partition method.*

2.1.1 Entire Sequence Testing Method

In this method, the entire sequences will be tested without any partitioning. As mentioned before, some statistical tests can be applied to long sequences using this method in the literature. The primary purpose of this method is to test the sequence roughly. For this method, four tests are chosen: weight, run, R2 run, and autocorrelation from the literature. Tests are selected by considering Golomb's postulates. Approximate distributions are used according to the random variable of the test. In this thesis, test statistics of weight, run, and correlation from the proposed test for the entire sequence testing method approaches normal distribution for long sequences. In addition, we use the χ^2 goodness of fit test in the R2 run test.

For using this method, Algorithm 1 should be followed;

Algorithm 1: Applying Entire Sequence Testing Method

```
Result: If p-value $\geq 0.01$ sequence pass
Generate sequence $S$ length $n \geq 10^6$;
Define test and random variable $X$ and compute test statistic;
If necessary convert sequence to generate integer or $\pm 1$ sequence;
while $t \leq n$ do
    Count the random variable ;
end
Compute p-value with using test statistic;
```

2.1.2 Fixed Length Partition Method

The fixed-length partition method is widely used in the literature. As mentioned before, this method is used frequently to test long sequences. In this method, a fixed number b is chosen, and the generated long sequence is partitioned into short subsequences of length b. Then, tests that can be applied to short sequences as a bundle are applied. The main advantage of this method is that statistical computation can be used automatically. In other words, the statistical calculations that cannot be performed in long sequences become applicable with this method. On the other hand, there are some disadvantages of using this method. While testing for randomness, every information that sequences will carry is critical. However, artificially partitioning the sequence will cause some information to deteriorate. Therefore, this method should be used very carefully, and this method should not be applied to every test. In this thesis, the proposed tests have been chosen that will not deteriorate the structure of the sequence. Weight test and total run test are chosen as default tests. In addition to these two tests, we propose linear complexity and integer coverage tests resistant to structural disorders. The tests in this method can be quickly diversified. For using this method, the given Algorithm 2 should be followed;

```
Algorithm 2: Applying Fixed Length Partition Method
  Result: If p-value$\geq 0.01$ sequence pass
  Generate sequence $S$ length $n \geq 10^6$;
  Define test and random variable $X$ and fixed number $b$;
  Compute test statistic according to $b$ and compute the exact bin values;
  If necessary convert sequence to generate integer or $\pm 1$ sequence ;
  while $b * i \leq n$ do
      Partitioning sequence to non-overlapping subsequence $S_i$ of length $b$ ;
      For $S_i$ count the random variable and get $t_i$;
      Separate $t_i$ to relative bin;
  end
  Count the observed values for each bin;
  Compute p-value by using $\chi^2$ goodness of fit test and incomplete gamma
   function;
```

2.1.3 Dynamic Partition Method

The method in which the random variable determined in the tests partitioned the sequence is called the dynamic partition method. We adapt the method used in collision estimate to statistical tests for randomness of long binary sequences [47] and integrate with seven randomness tests. We recommend this method since tests are given in the literature either can not be applied to long sequences or do not respect the structure of the sequences. Although the implementation of random variables is simple, time complexity issues and computational difficulties arise when dealing with long sequences. Some test suites have overcome this problem by using asymptotic approximation or partitioning the sequence into short and equal length parts and then evaluating the collection obtained for randomness. In the suggested method, a sequence will be partitioned into meaningful parts. The random variables themselves determine the termination of the parts without deteriorating the structure of the sequences.

In this thesis, we chose seven practical and informative tests for this method. Weight tests and total run tests are chosen as default tests. We propose three different tests:

Linear complexity and integer tests named: collision type 1 and type 2 tests and saturation point tests, and a new test named index coincidence point. For using this method, Algorithm 3 should be followed.

Algorithm 3: Applying Dynamic Partition Method

```
Result: If p-value $\geq 0.01$ sequence pass
Generate sequence $S$ length $n \geq 10^6$;
Define test and random variable $X$ and fixed number $b$ according to $X$;
Compute the exact bin values;
If necessary convert sequence to generate integer or $\pm 1$ sequence ;
while $t \leq n$ do
    if $X occur$ then
        Partition the sequence and count variable and separate to bin.;
    end
end
Count the observed values for each bin;
Compute p-value with using test statistic;
```

2.2 Randomness Tests

In this section, the definitions, computations, and evaluations of the randomness tests in all methods used in this thesis are given. From the literature, weight test, run tests, auto-correlation test, linear complexity test, integer tests were used in this method for this thesis. Finally, a a new test called as index coincidence test is proposed.

2.2.1 Weight Test

Definition 2.2.1. Weight is the number of 1's in a binary sequence.

Example 2.2.2. Let $S = 00101100001100010110110100100011111110001$ then the weight $w(S) = 19$.

In a random binary sequence with length n, from the property of balancedness, it is expected that weight should be equal to $n/2$. This test aims to determine the number of $1's$ and $0's$ are equally distributed through the sequence. In a binary sequence with length n, the weight of a binary sequence can be at least 0 ($00\ldots0$) and at most n ($11\ldots1$).

In the NIST test suite, [34], there are two weight tests named Frequency and Frequency within a Block. In the Frequency test, the number of $1's$ in a sequence is tested with two different methods. One of these methods is using normal distribution directly. The other method is partitioning the sequence into equal length subsequences, then computing *p-value* using the normal distribution, and separating *p-value* in ten uniformly divided bins. This approximation will be misleading given in [40]. On the other hand, in the Frequency within in a Block test, the sequence is partitioned into fixed-length non-overlapping subsequences. For each subsequence, weight is computed, and χ^2 distribution is used. However, while test statistic was evaluated for each subsequence expected value of weight was given as $\frac{t}{2}$. The result of this test will be misleading since some parts of randomly generated sequences should look non-random.

In this thesis, the weight test has been included for all methods since it is one of the basic tests is suitable for each method. Application of methods, statistical and theoretical computations are given in the related sections.

2.2.1.1 Entire Sequence Weight Test

For a random binary sequence, since bits are independent and identically distributed, it is expected the probability of generated bit 1 or 0 is equal, that is $P(S_i = 1) = P(S_i = 0) = \frac{1}{2}$.

Let the number of different sequence with weight w and length n be S_w. Then

$$S_w = \binom{n}{w}. \tag{2.3}$$

The probability of generated sequence of length n is:

$$P(w(S) = w) = \binom{n}{w}\left(\frac{1}{2}\right)^{w}\left(\frac{1}{2}\right)^{n-w} = \frac{\binom{n}{w}}{2^n}. \tag{2.4}$$

The probability distribution function of this function is the binomial distribution. From the Central Limit Theorem for big n, the binomial distribution approaches the normal distribution. To make computations easy, we use random variable X that is equal to the difference between weight and mean, $X = \mu - S_w$. Then mean of X is equal to 0 and standard deviation $\sigma = \sqrt{n}/\sqrt{2}$. Since X has normal distribution it becomes standard normal distribution by using transformation $\frac{X-\mu}{\sigma} = \frac{X\sqrt{2}}{\sqrt{n}} = Z$

From Section 1.1.1.2 we get,

$$P\left(\frac{S_w}{\sqrt{n}} < z\right) = 2\phi(z) - 1 = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{z} e^{\frac{-u^2}{2}}\, du. \tag{2.5}$$

To test a generated sequence the following steps should be done [34]:

1. Count the weight S_w of the generated sequence S.
2. Use standard normal distribution, compute $X = |\mu - S_w|$ and divide X by $\sqrt{n}$.
3. Compute *p-value*, which is $2(1 - \phi(X)) = erfc(\frac{X}{\sqrt{2}})$

Remark 2.2.3. This test is completely equal to the Frequency test in NIST test suite [34]. In our opinion, this test should be done for every sequence. If the sequence fails, then the sequence can be declared as non-random without applying the other tests. However, if the sequence passes the test, this does not imply that sequence is random. For example, $S = 11001100....$ will pass from the weight test.

2.2.1.2 Fixed Length Partition Weight Test

The disadvantages of weight tests in the NIST test suite were stated. In this thesis, we overcome this problem by using weight test statistics proposed by Koçak [23].

Bin Bounds and Probabilities

Fact 2.2.4. Let $P(w \leq t)$ denote the probability of weight less than t. From $P(w = k) = 2^{-n}\binom{n}{k}$

$$P(w \leq t) = 2^{-n} \sum_{i=0}^{t} \binom{n}{i} \tag{2.6}$$

By using this fact, the following recursion can be found:

$$P_n(w \leq t) = \frac{1}{2}[P_{n-1}(w \leq t) + P_{n-1}(w \leq t-1)] \tag{2.7}$$

where $P_n(w \leq t)$ is equal to probability of weight less than or equal to t where the sequence length is equal to n. The following steps should be done to apply weight test in this method;

1. Set b and compute the relative bin values by using Equation 2.7. In this thesis we set $b = 256$. The bin bounds and the probabilities according to $b = 256$ are given in Table 2.4.
2. Partition the generated sequence according to b.
3. Compute the weight of each subsequence and separate them into relative bins.
4. Multiply each bin probability with the number of subsequences to get the expected number of sequences.
5. Compute χ^2 goodness of fit test and compute *p-value* by using $igamc(\chi, dof)$. (degree of freedom of this test is *number of bins-1*).

For computing bin probabilities, the Equation 2.7 can be used. For example, for the first bin $P_{256}(w \leq 0) + P_{256}(w \leq 1) + +P_{256}(w \leq 118)$ should be computed.

Table 2.4: Bin values of Fixed Length Partition Weight Test for $b = 256$

bins	1	2	3	4	5	6	7	8
weight	$[0-119)$	$[119-123)$	$[123-126)$	$[126-129)$	$[129-131)$	$[131-134)$	$[134-138)$	$[138-256]$
expected probability	0.11749	0.12844	0.13144	0.14754	0.09773	0.13144	0.12844	0.11748

Remark 2.2.5. The fourth and the fifth bin values are not equal since the probability of $P(w = 128)$ for $b = 256$ has the greatest probability. If we change bin bounds

or use nine bins with a bin that includes only probability $P(w = 128)$, we can get symmetric bin values. However, while we compute bin bounds, we separate them in probabilities as close to each other as possible.

2.2.1.3 Dynamic Partition Weight Test

In this section, the computation of the weight test in the dynamic partition method will be given. In this method length of subsequences depend on the weight of parts. Let S be n bit binary sequence, and w is the predestined weight. Record the first index k at which weight achieves w and delete this subsequence. The same process is repeated throughout the sequence. Suppose the weight does not reach weight w before the fixed index t, record the index t, and continue the process. t is chosen so that the probability of not reaching the weight w till index t is negligible.

Example 2.2.6. Let $S = 001010100010100010110110100101001110110 01$ and take $w = 2$, $t = 7$. (Notice that the probability of not reaching the weight $w = 2$ until 7^{th} term is $\frac{1}{2^7}$)

$S_1 = 00101$, $S_2 = 010001$, $S_3 = 010001$, $S_4 = 011$, $S_5 = 011$, $S_6 = 01001$, $S_7 = 01001$, $S_8 = 11$, $S_9 = 011$ the rest is deleted. Hence, $w_2 = 1$, $w_3 = 3$, $w_4 = 0$, $w_5 = 3$, $w_6 = 2$, $w_{\geq 7} = 0$

Fact 2.2.7. Let $P_{w(k)}$ denote the probability that weight achieves w at bit k for the first time. Then,

$$P_{w(k)} = \frac{1}{2^k}\binom{k-1}{w-1}. \tag{2.8}$$

Remark 2.2.8. Weight cannot be equal to w before the term w ; that is $k \geq w$.

Bin Bounds and Probabilities

Bin values should be determined to use the χ^2 distribution. In order to make test statistics more understandable, the bin probabilities are chosen as close to each other as possible.

The suggested parameter values for w and t are 128 and 280, respectively. The range with respect to $w = 128$ is $\{128, 129, \cdots\}$. Bounds and corresponding probabilities of bins can be seen in the following Table 2.5. For computing bin probabilities, the Equation 2.8 can be used. For example, to get the first bin probability $P_{128} + P_{129} + \ldots + P_{238}$ should be computed.

Table 2.5: Bin values of Dynamic Partition Weight Test

i	1	2	3	4	5	6	7	8
partition length	128-238	239-245	246-250	251-255	256-260	261-266	267-274	275 $\leq$
probabilities	0.13522	0.12626	0.11445	0.12404	0.12171	0.12823	0.12455	0.12549

2.2.2 Run Tests

Definition 2.2.9. In a binary sequence, uninterrupted sub-sequences of identical bits are called run.

Let $S = \sigma_1, \sigma_2 \ldots, \sigma_n$ be a binary sequence with length n. Then S' is first derivative of S with each entry of S' is $S'_i = \sigma_i + \sigma_{i+1}$ and without loss of generality $S'_n = 1$.

Until the end of the sequences, the end of each run means the beginning of the new run. In a binary sequence with length n, number of a runs can be at least 0 ($00\ldots0$ or $11\ldots1$) and at most $n-1$ ($1010\ldots10$ or ($0101\ldots01$).

Example 2.2.10. Let $S = 001011000011$ then runs of S are; $r_1 = 00,\ r_2 = 1,\ r_3 = 0,\ r_4 = 11,\ r_5 = 0000,\ r_6 = 11$ respectively.

As weight test, run tests is one of the basic tests that almost all suites involve. In the NIST test suite [34], there are two run test named *"run test"* and *"test for the longest run of ones in a block"*. The run test evaluates the number of total runs directly by using the normal distribution. Test for the longest run of ones in a block evaluates the count of those in a block by using χ^2 distribution with exact probabilities. In this thesis, we defined two different run tests named the total number of runs test and the R2 run test.

2.2.2.1 Entire Sequence Total Number of Runs Test

In a random binary sequence with length n the expected total number of the run is equal to $n/2$ [11]. The total number of runs test is the basic test of test suites, has been included in the test suite for all four methods. This test aims to determine whether the number of ones and zeros are uniformly distributed along the sequence or not.

As defined in Section 2.2.2, consecutive bits that involve only 1 or only 0 that continues unchanged are called run. This test in this method aims to determine whether the total number of runs in the generated sequence looks like the number of runs in a random sequence.

For finding the total number of runs in a sequence, the weight of the derivative S', given in Section 2.2.2 of the sequence, should be calculated. Then, the total number of runs, $R_t(S)$ is, $R_t(S) = S'_w$.

One of the templates, "01" or "10" should be generated for a run. There are four different 2-bit templates, "00, 01, 10, 11". As in weight test, since bits are independent identically distributed, it is expected that the probability of generated bits is one of the given templates are $P(S_{i,i+1} = 00) = P(S_{i,i+1} = 01) = P(S_{i,i+1} = 10) = P(S_{i,i+1} = 11) = \frac{1}{4}$. In other words, a run occurs in 2 bits with a probability $\frac{1}{2}$. Therefore, expected total number of run is equal to $\frac{n}{2}$.

As weight test, Total Number of Run Test in this method is very similar to the run test in NIST test suite [34]. Since we use this test for long sequences without partitioning, the only difference is that we do not use weight to evaluate the expected total number of runs. The rest of this test is very similar to the weight test in this method. For computing *p-value*, the following step should be done;

1. Evaluate the derivative, S' of generated sequence S.
2. Compute S'_w.
3. Since we use standard normal distribution compute $X = |\mu - S'_w|$ and divide X by $\sqrt{n}$.
4. Since we need to compute *p-value*, which is $2(1 - \phi(X)) = erfc(\frac{X}{\sqrt{2}})$

Remark 2.2.11. As weight test, the total number of runs test should be done. Although sequence fails to imply that it is non-random, sequence passes is not imply randomness.

2.2.2.2 Fixed Length Partition Total Number of Run Test

The only run test we use in this method is the number of total run tests. To apply this test for long sequences, statistical computations of the run test proposed by Koçak in [23] are using.

Bin Bounds and Probabilities

Fact 2.2.12. Let $P(r \leq t)$ denote the probability of number of runs less than or equal to t. Number of runs is equal to $t-1$ means total number of 01 and 10 in the sequence is $t-1$. That is, weight of derivative sequence S' is equal to t. Then

$$P(S'_w = t) = P(r = t) = \frac{1}{2^n}\binom{n-1}{t-1} \tag{2.9}$$

From this equation we get,

$$P(r \leq t) = \frac{1}{2^n}\sum_{i=0}^{t}\binom{n-1}{i-1} \tag{2.10}$$

By using this fact the following recursion can be found.

$$P_n(r \leq t) = \frac{1}{2}[P_{n-1}(r \leq t) + P_{n-1}(r \leq t-1)] \tag{2.11}$$

where $P_n(r \leq t)$ is equal to probability of number of runs less than t where the sequence length is equal to n. For applying this test to a long sequence, the following steps should be done;

1. Set b and compute the relative bin values by using Equation 2.11. In this thesis we set $b = 256$. The bin bounds and the probabilities according to b is given in Table 2.6.

2. Partition the generated sequence according to b.

3. Compute run of each subsequence and separate them relative bins to get observed number of sequences.

4. Multiply each bin probability with number of subsequence to get expected number of sequences.

5. Compute χ^2 goodness of fit test and compute *p-value* by using $igamc(\chi, dof)$. (degree of freedom of this test is number of bins-1).

For computing bin probabilities, the Equation 2.11 can be used. For example, for the first bin $P_{256}(r \leq 0) + P_{256}(r \leq 1) + +P_{256}(r \leq 118)$ should be computed.

Table 2.6: Bin values of Fixed Length Partition Total Number of Run Test for $b = 256$

bins	1	2	3	4	5	6	7	8
number of run	$[1-119)$	$[119-124)$	$[124-127)$	$[127-129)$	$[129-132)$	$[132-135)$	$[135-139)$	$[139-256]$
expected probability	0.12981	0.13582	0.13551	0.09887	0.14640	0.12738	0.12106	0.10516

2.2.2.3 Dynamic Partition Total Number of Runs Test

As in the weight test in this method, we decide the test point that the number of runs reaches the predestined values.

Let S be n bit binary sequence, and r is the predestined number of runs. Record the first index k at which the number of runs and delete this subsequence. The same process is repeated throughout the sequence. If the number of runs does not reach to number r before the index t, record the index t and delete the subsequence. t is chosen so that the probability of not reaching the number of runs r till index t is negligible. The process is familiar with the weight test in this method.

Example 2.2.13. Let $S = 0011100001010000000111$ and take $r = 2, t = 7$. (Notice that the probability of not reaching the number of runs $r = 2$ until $t = 7^{th}$ term is $\frac{1}{2^8}$) Since it is easy the see the number of runs from the derivative of the sequence, compute the S'

$S' = 010010001111000000100$ then

$S_1' = 01001$, $S_2' = 00011$, $S_3' = 11$, $S_4' = 0000001$, the rest is deleted. Hence, $r_2 = 1$, $r_5 = 2$, $r_{\geq 7} = 1$

Bin Bounds and Probabilities

For using χ^2 distribution, bin values have to be determined. In order to make the test statistics more understandable, the bin values should be chosen as close to each other as possible.

Let S be a binary sequence. Then compute S' for compute number of runs. It is expected that S' is random. Let r be the predestined number of runs that sequence partition. In other words, r is the weight of S' that S' is partitioned. The only difference between this test and the weight test in this method is since S' is generated from the original sequence S by compute $\sigma_i + \sigma_{i+1}$, when i^{th} index of S' is generated in the original sequence the $i + 1^{th}$ index is generated. Other computations are the same as the weight test. However, it can be seen from Table 2.7 bin bounds are different.

The suggested parameter values for the r and t are 128 and 281, respectively. The range with respect to $r = 128$ is $\{128, 129, \cdots\}$. Bounds and corresponding probabilities of bins can be seen in the following Table 2.7. For computing bin probabilities, the Equation 2.8 can be used. For example, to get the first bin probability $P_{129} + P_{130} + \ldots + P_{239}$ should be computed.

Table 2.7: Bin values of Dynamic Partition Method Total Run Test

i	1	2	3	4	5	6	7	8
partition length	129-239	240-246	247-251	252-256	257-261	262-267	268-275	276 $\leq$
probabilities	0.13522	0.12626	0.11445	0.12404	0.12171	0.12823	0.12455	0.12549

2.2.2.4 Entire Sequence R2 Run Test

There are many run tests in the literature. They are designed according to the total number of runs or the lengths of the runs. However, neither of them fully corresponds to the second postulate of Golomb. When testing a long sequence, this postulate was not cited as a test. This test aims to evaluate the sequence exactly like in Golomb's postulate and determine whether frequencies of the length of runs are uniformly distributed or not. Since this test depends on the total number of runs, if the sequence fails the entire sequence total number of tuns test 2.2.2.1, this test should not be applied.

Due to the nature of the test, it can be applied only to long sequences. Hence, this test is used only in the entire sequence testing method.

Remark 2.2.14. As mentioned before, To occur a run, one of the templates "01" or "10" should be generated. So that to occur, a run of length 1 one of template "010" or "101" should be generated. Each case can occur with probability $\frac{1}{2}$ implies that the expected total number of runs of length 1 is half of the total number of runs. While the expected total numbers of runs of length l compute, the same method can be used. To see a run of length l $"01-(l-2\ ones)-10"$ or $"10-(l-2\ zeros)-01"$ should be generated. For each case, the probability is $\frac{1}{2^l}$, which implies that the expected total number of runs of length l is one-l^{th} of the total number of runs.

Test Description:

To perform this test, the followings should be done;

1. Total number of runs should be counted.
2. Total number of runs of length i, R_i, for $i = 1,2,3,4,5,6,7,\geq 8$ should be counted.
3. Using bin values given Table 2.13, counts of bins should be calculated, note that last bin can be calculated by using other values.
4. χ^2 value should be calculated by using Equation 1.6 and by using values in Table 2.8.
5. At the end compute *p-value* by using Equation 1.9, $igamc(\chi^2, 7)$.

Bin Bounds and Probabilities

Let r be the total number of runs of σ, where length of $sigma$ is n. Then expected number of runs of length i, $E(r_i)$, is computed as follows.

Case r_1:

Let $S' = s'_1, s'_2, \ldots, s'_{n-1}, 1$ be derivative of σ. Then a run means that $s_i = 1$. Except from first and the s'_{n-1} term runs of length 1 means that $s'_i = s'_{i+1} = 1$. For the first term and and the s'_{n-1} term it is enough to $s'_1 = 1$and $s'_{n-1} = 1$. Then expected number of runs of length 1 is,

$$Prob(s'_1 = 1) = Prob(s'_{n-1} = 1) = \frac{(r-1)}{n-1} \tag{2.12}$$

$$Prob(s'_i = s'_{i+1} = 1) = \frac{(n-2)(n-1)(r-2)}{(n-1)(n-2)} \tag{2.13}$$

$$E(r_1) = \frac{r(r-1)}{n-1} \tag{2.14}$$

Case r_2:

Except from first two and last three term runs of length 2 means that $s'_i = s'_{i+2} = 1$ and $s'_{i+1} = 0$. For the first term and and the s'_{n-2}, s'_{n-1} term it is enough to $s'_1 = 0$ and $s'_2 = 1$ and $s'_{n-1} = 0$ and $s'_{n-2} = 1$. Then expected number of runs of length 2 is,

$$Prob(s'_1 = 0 \; and \; s'_2 = 1) = Prob(s'_{n-1} = 0 \; and \; s'_{n-2} = 1) = \frac{(r-1)(n-r-1)}{(n-1)(n-2)} \tag{2.15}$$

$$Prob(s'_i = s'_{i+2} = 1 \; and \; s'_{i+1} = 0) = \frac{(n-3)(r-1)(r-2)(n-r-1)}{(n-1)(n-2)(n-3)} \tag{2.16}$$

$$E(r_2) = \frac{r(r-1)(n-r-1)}{(n-1)(n-2)} \tag{2.17}$$

Case r_i:

To generalize we get;

$$E(r_i) = \frac{r(r-1)(n-r-1)\ldots(n-r-(i-1))}{(n-1)(n-2)\ldots(n-t)} \tag{2.18}$$

Table 2.8: Bin values of Entire Sequence R2 Run Test

bins	1	2	3	4	5	6	7	≥ 8
length of run	1	2	3	4	5	6	$7-8$	≤ 9
expected count	$E(r_1)$	$E(r_2)$	$E(r_3)$	$E(r_4)$	$E(r_5)$	$E(r_6)$	$E(r_7)$	$E(r_{\geq 8})$

2.2.3 Auto-correlation Test

Another proposed test is in this thesis from Golomb's postulates [11] is auto-correlation. According to Golomb's third postulate in a random binary sequence, the auto-correlation function should be two-valued.

Definition 2.2.15. Let S be a binary sequence with length n. k auto-correlation of S is;

$$C_k = \sum_{j=0}^{n-k-1} s_j + s_{j+k} \tag{2.19}$$

where s_j is j^{th} term of S.

In a random binary sequence auto-correlation function should be two valued. Obviously if $k = 0$, the $C_0 = n$. The expectation when $k \neq 0$ is $C_k = K$ for some integer $0 < K < n$.

Koçak proposed a version of this test in [23], that can be applied only to short sequences.

2.2.3.1 Entire Sequence Auto-correlation Test

Remember that, let S be random binary sequence then; k auto-correlation of S is computed from the Equation 2.19. Since each bit of random sequence is independent and identically distributed, then shifted sequence of the random binary sequence is expected that random. The sum of two random sequences should be random implies that it is expected that the auto-correlation of C_k is equal to $\frac{n}{2}$ for any k.

The test statistic is equal to weight test. For applying auto-correlation test, the following steps should be done;

1. Define a number b such that auto-correlation function should be run. In this thesis we choose $b = 256$

2. Identify shifted b sequences to be calculated auto-correlation. In this thesis we choose $k = 1, 2, \ldots, 256$
3. Calculate auto-correlation c_k for each k.
4. Calculate *p-value* by using standard normal distribution as in weight test.
5. Count number of *p-value*< 0.01, say t.
6. If $t > \frac{b}{100}$ then determine as sequence non-random.

This test aims to determine the periodic structure of the sequences. So that, it is recommended that to choose b as big as possible.

2.2.4 Linear Complexity

Definition 2.2.16. Let σ be a binary sequence, linear complexity of σ is the length of the shortest linear feed back shift register(LFSR) that can produce σ.

Linear complexity test is also included in NIST test suite [34]. Time complexity exponentially increases with respect to the length of sequence while computing linear complexity. NIST overcomes this problem by using the fixed-length partition method and use the χ^2 distribution. In this thesis, we use this method to apply linear complexity. Moreover, we used the dynamic partition method to make the linear complexity test more meaningful. These tests aim to determine whether or not the linear complexity of a given sequence is as expected. These tests should be applied only for short sequences. Because of this, the linear complexity test is used in fixed-length partition and dynamic partition methods.

2.2.4.1 Fixed Length Partition Linear Complexity Test

Linear complexity test in this method is completely the same as in NIST test suite [34]. Since our opinion is linear complexity test is significant, it is appropriate to include it in this method in this thesis.

Bin Bounds and Probabilities

Berlekamp-Massey algorithm can be used to calculate linear complexity of a sequence. [3, 29]. Let $L(S) = L_1, \ldots, L_n$ be linear complexity profile of the binary sequence $S = s_1, \ldots, s_n$ of length n, that is, for each, $k = 1, \ldots\ n,\ L_k$ is the linear complexity of the subsequence $s_1, \ldots, s_k$. Denoted the linear complexity of S by Λ, i.e, $\Lambda = L_n$. Then probability of a sequence S have linear complexity $\Lambda = l$ is;

$$Prob(\Lambda = l) = \begin{cases} 2^{-n} & \text{if} \quad l = 0 \\ 2^{2l-n-1} & \text{if} \quad 1 \leq l \leq n/2 \\ 2^{n-2l} & \text{if} \quad n/2 < l \leq n \end{cases} \tag{2.20}$$

By using the equation, relative bin values can be computed. For applying this test to a long sequence, the following steps should be done;

1. Set b and compute the relative bin values by using Equation 2.20. In this thesis we set $b = 256$. The bin bounds and the probabilities according to b is given in Table 2.9.

2. Partition the generated sequence according to b.

3. Compute linear complexity of each subsequence and separate them relative bins to get observed number of sequences.

4. Multiply each bin probability with number of subsequence to get expected number of sequences.

5. Compute χ^2 goodness of fit test and compute *p-value* by using $igamc(\chi, dof)$. (degree of freedom of this test is *number of bins-1*). For $b = 256$, it is suitable to use 7 bins.

For computing bin probabilities the Equation 2.20 can be used. For example, to get the first bin probability $P(\Lambda = 0) + P(\Lambda = 1) + \ldots + P(\Lambda = 125)$ for $n = 256$ should be computed.

Table 2.9: Bin values of Fixed Length Partition Linear Complexity Test for $b = 256$

bins	1	2	3	4	5	6	7
linear complexity	$[0-126)$	$[126-127)$	$[127-128)$	$[128-129)$	$[129-130)$	$[130-131)$	$[131-256)$
expected probability	0.01042	0.03125	0.125	0.5	0.25	0.0625	0.0208

2.2.4.2 Dynamic Partition Linear Complexity Test

As in the weight and total run test, the linear complexity test was used in the dynamic partition method. We decided that the linear complexity test will take place in this method since this test can be modified for the dynamic partition method. The calculations of this test in the dynamic partition method can better understand.

Computation of Probabilities

Given a non-negative integer $l \leq n$, recall that the Equation (2.20). For a given positive integer $\lambda \leq n$, there are two possibilities: either $\Lambda < \lambda$ or $\Lambda \geq \lambda$. The probability of the first event can be computed directly from above as;

$$Prob(\Lambda < \lambda) = \begin{cases} 2^{-n} & \text{if} \quad \lambda = 1 \\ \frac{1}{3 \cdot 2^n}(1 + 2^{2\lambda - 1}) & \text{if} \quad 2 \leq \lambda \leq n/2 \\ 1 - \frac{1}{3 \cdot 2^n}(4^{n-\lambda+1} - 1) & \text{if} \quad n/2 < \lambda \leq n \end{cases} \cdot \tag{2.21}$$

Now, assume that $\Lambda \geq \lambda$. It is clear that $L(S)$ need not to assume the value λ but certain that $L_k \geq \lambda$ for some index k. Define $\overline{\lambda}$ be the smallest value not less than λ which is assumed by let $L(S)$ let $i(\lambda)$ is index of the term at which $L(S)$ assumes that value $\overline{\lambda}$ for the first time. Namely,

$$i(\lambda) = min\{k | L_k \geq \lambda\}.$$

For any $\lambda \leq \lambda' \leq \overline{\lambda}$, it is obvious that $\overline{\lambda'} = \overline{\lambda}$ and also $i(\lambda') = i(\lambda)$. It follows from minimality of $\overline{\lambda}$ that $L_k < \lambda \leq \overline{\lambda}$ for all $k = 1, \ldots, i(\lambda) - 1$ so we conclude that

$$\delta_{i(\lambda)-1} = 1 \text{ and } L_{i(\lambda)} = i(\lambda) - \overline{\lambda},$$

where δ is the next discrepancy function. Last equation implies that $i(\lambda) - \overline{\lambda} < \lambda$ which leads to

$$\overline{\lambda} \leq i(\lambda) \leq \lambda + \overline{\lambda} - 1.$$

Now we have

$$Prob\big(i(\lambda) = k \; and \; \overline{\lambda} = t\big) = \begin{cases} 2^{-t} & \text{if} \quad k = t \\ \frac{1}{2} Prob(L_{k-1} = k - t) & \text{if} \quad t+1 \leq k \leq t+\lambda-1 \\ 0 & \text{if} \quad t+\lambda < k \leq n \end{cases} \cdot$$

The condition $i(\lambda) \leq \lambda + \overline{\lambda} - 1 \leq 2\overline{\lambda} - 1$ implies that $i(\lambda) - \overline{\lambda} \leq \frac{1}{2}(i(\lambda) - 1)$ so that

$$Prob(L_{k-1} = k - \overline{\lambda}) = 2^{k-2\overline{\lambda}},$$

hence

$$Prob\big(i(\lambda) = k \; and \; \overline{\lambda} = t\big) = \begin{cases} 2^{-t} & \text{if} \quad k = t \\ 2^{k-2t-1} & \text{if} \quad t+1 \leq k \leq t+\lambda-1 \\ 0 & \text{if} \quad t+\lambda \leq k \leq n \end{cases} \cdot$$

Fix an integer λ_0 with $\lambda \leq \lambda_0 \leq n$. We conclude the probability of the event $\overline{\lambda} = \lambda_0$:

Case 1: $\lambda_0 + \lambda - 1 \leq n$

$$\begin{aligned} Prob(\overline{\lambda} = \lambda_0) &= \sum_{k=\lambda_0}^{\lambda_0+\lambda-1} Prob(i(\lambda) = k \; and \; \overline{\lambda} = \lambda_0) \\ &= 2^{-\lambda_0} + \sum_{k=\lambda_0+1}^{\lambda_0+\lambda-1} 2^{k-2\lambda_0-1} \\ &= 2^{-\lambda_0} + 2^{-2\lambda_0}(2^{\lambda_0+\lambda-1} - 2^{\lambda_0}) \\ &= 2^{-\lambda_0+\lambda-1}. \end{aligned}$$

Case 2: $n < \lambda_0 + \lambda - 1$

$$\begin{aligned}
Prob(\overline{\lambda} = \lambda_0) &= \sum_{k=\lambda_0}^{n} Prob(i(\lambda) = k \; and \; \overline{\lambda} = \lambda_0) \\
&= 2^{-\lambda_0} + \sum_{k=\lambda_0+1}^{n} 2^{k-2\lambda_0-1} \\
&= 2^{-\lambda_0} + 2^{-2\lambda_0}(2^n - 2^{\overline{\lambda}}) \\
&= 2^{n-2\lambda_0}.
\end{aligned}$$

We summarize these computations as

$$Prob(\overline{\lambda} = \lambda_0) = \begin{cases} 2^{-\lambda_0+\lambda-1} & \text{if} \quad \lambda_0 + \lambda - 1 \leq n \\ 2^{n-2\lambda_0} & \text{if} \quad n < \lambda_0 + \lambda - 1 \end{cases}.$$

Now we focus on computation of $Prob(i(\lambda = k))$.

$$Prob(i(\lambda) = k) = \sum_{t=\lambda}^{k} Prob(i(\lambda) = k) \; and \; \overline{\lambda} = t).$$

Case 1. $\lambda \leq k \leq 2\lambda - 1$

$$\begin{aligned}
Prob(i(\lambda) = k) &= \sum_{t=\lambda}^{k-1} 2^{k-2t-1} + 2^{-k} \\
&= 2^{k-1} \sum_{t=\lambda}^{k-1} \frac{1}{4^t} + 2^{-k} \\
&= \frac{1}{3} 2^{k+1} \left((\frac{1}{4})^{\lambda} - (\frac{1}{4})^{k} \right) + 2^{-k} \\
&= \frac{1}{3}(2^{-2\lambda+k+1} + 2^{-k}).
\end{aligned}$$

Case 2. $2\lambda \leq k \leq n$

$$\begin{aligned}
Prob(i(\lambda) = k) &= \sum_{t=k-\lambda+1}^{k-1} 2^{k-2t-1} + 2^{-k} \\
&= 2^{k-1} \sum_{t=\lambda}^{k-1} \frac{1}{4^t} + 2^{-k} \\
&= \frac{1}{3} 2^{k+1} \left((\frac{1}{4})^{k-\lambda+1} - (\frac{1}{4})^k \right) + 2^{-k} \\
&= \frac{1}{3}(2^{2\lambda-k-1} + 2^{-k}).
\end{aligned}$$

Hence

$$Prob(i(\lambda) = k) = \begin{cases} \frac{1}{3}(2^{-2\lambda+k+1} + 2^{-k}) & \text{if} \quad \lambda \leq k \leq 2\lambda - 1 \\ \frac{1}{3}(2^{2\lambda-k-1} + 2^{-k}) & \text{if} \quad 2\lambda \leq k \leq n \end{cases}.$$

Bin Bounds and Probabilities

We partition the long sequence where linear complexity achieves a value larger than or equal to 128 for the first time. If it cannot reach until term 280, we also partition term 280.

You may find bin bounds and probabilities in the Table 2.10. Note that the last bin also includes the probability that the sequence achieves a linear complexity strictly less than 128 at term 280.

Table 2.10: Bin values of Dynamic Partition Linear Complexity Test

i	1	2	3	4	5	6	7
bounds	≤ 252	253	254	255	256	257	258-280
probabilities	0,083333	0,083333	0,166667	0,333333	0,166667	0,083333	0,083333

2.2.5 Integer (Non-overlapping Template) Tests

Let S be a binary sequence, and $\tilde{S}$ is the integer (non-overlapping template) sequence of it. Integer values depend on the length of the non-overlapping templates. For different length templates, regenerated sequences will change. In literature, integer tests are used in tests suites frequently [4, 27, 25, 34]. In this thesis, we propose three different integer tests named collision, saturation point, and coverage. These tests are proposed because which method should be used in which tests can be understood very clearly after applying these tests.

2.2.5.1 Fixed Length Partition Coverage Test

Definition 2.2.17. Let $\tilde{S} = \tilde{S}_1, \ldots, \tilde{S}_n$ be an integer sequence. The cardinality of the set $I = \{\tilde{S}_1, \ldots, \tilde{S}_n\}$, that is the number of distinct terms of the sequence $\tilde{S}$, is called the coverage of the sequence $\tilde{S}$.

In the literature, it is a version of coupon collector test proposed in [22]. Later, Sulak proposed it as a test in [40]. Then, Koçak proposed integer tests in [23]. He proposed eight different integers tests; however, one of our aims is to differentiate the methods so that we choose the coverage test as the most suitable one for applying this method to long sequences.

From the definition of this test, the sequence should end at some point. Otherwise, test statistics can become useless. Because of this reason, only the fixed-length partition method can be used for this test. Koçak proposed integer coverage tests for the short sequence in [23]. The statistical computation is similar for long sequences.

Bin Bounds and Probabilities

Consider a binary sequence S of length $b2^b$ to be converted to $b-bit$ integer sequence $\tilde{S}$ of length 2^b. Let c be the coverage of the sequence $\tilde{S}$. As each b-bit integer can take 2^b different values and there exist c different integers in the sequence, these integers that appear in $\tilde{S}$ can be chosen from $\binom{2^b}{c}$ different integer sets. Since the length of the sequence, $\tilde{S}$ is 2^b, and there are exactly c different integers, the sum of all frequencies of each integer is 2^b. The number of the piece of the integers separated in the sequence is $\left\{ {2^b \atop c} \right\}$, where $\{\}$ is the Stirling numbers of the second kind. Finally, there are $c!$ different orders of integers. Then, the probability of the sequence $\tilde{S}$ having coverage equal to c is:

$$P(\tilde{S} = c) = \frac{c!}{2^{b2^b}} \binom{2^b}{c} \left\{ {2^b \atop c} \right\}. \tag{2.22}$$

To apply this test to long sequences the following steps should be done;

1. Set b and convert the sequence and compute the relative bin values by using Equation 2.22. In this thesis we set $b = 8$. The bin bounds and the probabilities according to $b = 8$ is given in Table 2.11.
2. Partition the generated sequence according to b. In this thesis according to $b = 8$, subsequence lengths are 256.
3. Compute coverage of each subsequence and separate them relative bins to get observed number of sequences.
4. Multiply each bin probability with number of subsequence to get expected number of sequences.
5. Compute χ^2 goodness of fit test and compute *p-value* by using $igamc(\chi, dof)$. (degree of freedom of this test is *number of bins-1*).

For computing bin probabilities the Equation 2.22 can be used. For example, to get the first bin probability $P(c = 1)+P(c = 2)+\ldots+P(c = 156)$ should be computed.

Table 2.11: Bin values of Fixed Length Partition Integer Coverage Test for $b = 8$

bins	1	2	3	4	5	6	7	8
coverage	$[1 - 156)$	$[156 - 159)$	$[159 - 161)$	$[161 - 163)$	$[163 - 164)$	$[164 - 166)$	$[166 - 169)$	$[169 - 256]$
expected probability	0.095975	0.144646	0.140103	0.158122	0.078397	0.140815	0.145806	0.096136

2.2.5.2 Dynamic Partition Collision Tests

The random variable used in the collision estimate method described in [47] suggests partitioning sequences dynamically in testing. Since our approach originated from this method, we first adapt collision tests to binary sequences to test their randomness.

Definition 2.2.18. Let $\tilde{S}$ be an integer sequence with the first t entries are different and $(t+1)^{th}$ entry is equal to one of first t entries where $t = 1, 2..., 2^b - 1$ then *the index* $(t+1)$ and the $(t+1)^{th}$ entry are called *collision index* and *colliding integer*, respectively.

In this thesis, two different collision tests are proposed. Collision tests are appropriate for the new method since each collision point is a perfect partitioning point without losing any information. Collision tests aim to determine whether each integer in the sequence is uniformly and equally generated randomly or not. In the NIST test suite [34], there is a non-overlapping template matching test. However, the test examines the sequence considering only one template all around it and repeating this test for non-periodic templates. On the other hand, Koçak [23] proposed an integer repetition test in his thesis. Although the test uses exact probabilities of the random variable, it is valid only for the sequences with restricted length. Still, it requires partitioning the sequence. We aim to modify this test for long sequences without losing statistical information. Since each collision will cause partitioning, only the dynamic partition method is used for this test. The use of other methods is not suitable for these tests.

Let S be a binary sequence of length n and $\tilde{S}$ be the integer sequence of it. By the Definition 2.2.18, the total number of collisions, collision indexes, and colliding integers will be determined. There are two different versions of collision tests, the distribution of recorded indexes and colliding integers.

Example 2.2.19. Let $S = 001010100010100010110110100101001110110 01$ and $b = 3$, then $\tilde{S} = 1\ 2\ 4\ 2\ 4\ 2\ 6\ 6\ 4\ 5\ 1\ 6\ 6$ and delete residual part at the end.

Remark 2.2.20. Residual part for a long sequence will not change the result. It will be the negligible.

Table 2.12: Example of Collision

$\tilde{s}_0$	$\tilde{s}_1$	$\tilde{s}_2$	$\tilde{s}_3$	$\tilde{s}_4$	$\tilde{s}_5$	$\tilde{s}_6$	$\tilde{s}_7$	$\tilde{s}_0$	$\tilde{s}_1$	$\tilde{s}_0$	$\tilde{s}_1$	$\tilde{s}_2$
1	2	4	2	4	2	6	6	4	5	1	6	6
0	1	2	$\underbrace{3}$	0	1	2	$\underbrace{3}$	0	1	2	3	$\underbrace{4}$

2 and 6 appear once and twice, respectively, as a colliding integer. Collision indexes 3 and 4 are seen twice and once, respectively. Others are not seen. The total number of collisions is 3.

Bin Bounds and Probabilities

Proposition 2.2.21. *Let $\tilde{S}$ be integer sequence. If the total number of coincides is C then the expected number of collisions of each integer is $E(i) = C/256$.*

Remark 2.2.22. By the pigeonhole principle, if the first $2^b - 1$ entries are different, $2^b th$ entry must be a colliding integer. The collision cannot be seen at the first entry.

For example, for $b = 8$, by the pigeonhole principle, the maximum collision index can be 257. From this point, the bin values are determined as follows.

In this test, χ^2 reference distribution is used to evaluate the sequence. We separate collisions into eight equal bins. For each bin, the expected number of the collision is equal as in Table 2.13. Since the last bin depends on the others, the degree of freedom is 7.

Table 2.13: Bin values of Dynamic Partition Collision Test about integers

bins	1	2	3	4	5	6	7	8
integer range	$[0-32)$	$[32-64)$	$[64-96)$	$[96-128)$	$[128-160)$	$[160-192)$	$[192-224)$	$[224-255]$
expected count	$C/8$	$C/8$	$C/8$	$C/8$	$C/8$	$C/8$	$C/8$	$C/8$

The second version is collision indexes.

Proposition 2.2.23. *Let d be a sequence whose terms are from a set T of $M = 2^8 = 256$ elements , and that C_k denote the event that first collision happens at index k Then*

$$Prob(C_k) = M^{-k}\binom{M}{k-1}(k-1)!(k-1), for k = 2\ldots 256$$

To make probability calculations feasible and faster, we use the following recursion in constructing Table 2.14

$$Prob(C_k) = Prob(C_{k-1})\left(\frac{1}{k-2} - \frac{1}{M}\right)(k-1).$$

The expected values for the bins in Table 2.14 can be computed as

$$E = \sum_{k=bin\ start}^{t=bin\ end} 256^{-k}\binom{256}{k-1}(k-1)!(k-1)C.$$

Table 2.14: Bin values of Dynamic Partition Collision Test about indexes

bins	1	2	3	4	5	6	7	8
index range	0-8	9-12	13-15	16-18	19-22	23-26	27-32	33-257
Expected count	0.13256·C	0.13387·C	0.11391·C	0.11545C	0.14315C	0.11964 · C	0.12566·C	0.11581·C

2.2.5.3 Dynamic Partition Saturation Point Test

Definition 2.2.24. Let $I = 0, 1, \ldots, t$ be a finite integer set, the elements of the sequence $\tilde{S}$ are generated from I. The point that all elements of I are generated is called the saturation point.

By the Definition of saturation point sequence is partitioning when all elements are generated. Therefore, only the dynamic partition method is used for this test. The entire sequence testing method should not be used because the saturation point will be at the very beginning of the sequence. The fixed-length partition method should not be used because most short-length subsequences cannot be achieved saturation. Otherwise, most of the subsequences achieved saturation point at the very beginning of the sequence. In both cases, the test result will be misleading.

Same as in the collision test, The random variable used in the collision estimate method described in [47] suggests partitioning sequences dynamically in testing. The original of this test depends on [40]. Koçak proposed saturation point as a test in [23]. However, this test can be applied for short sequences. Definition of saturation point is given in Section 2.2.5.3. In this thesis, we use saturation point for partition to the long sequence. Then tested values are the length of subsequences. From the Example 2.2.25 it will be understood well.

Example 2.2.25. Let $S = 10010111110001100101100111001100011 00$ and $b = 2$ then $\tilde{S} = 211330121121303012$ and delete residual part at the end, this not important 2.2.20. Then, subsequences are $\tilde{S}_1 = 211330, \tilde{S}_2 = 12112130, \tilde{S}_3 = 3012$ are generated from the definition of saturation point 2.2.5.3. Length of subsequences are 6, 8, 4 respectively.

Bin Bounds and Probabilities

Proposition 2.2.26. *Let $\tilde{S}$ be a $b-bit$ integer sequence of length n. Let $\tilde{S}_k$ be number of sequences that saturation point is k^{th} index. In such a sequence it is known that before the k^{th} index there should be $2^b - 1$ different integers generated and so last integer should be seen at the k^{th} index. This implies that first $k-1$ integers cover $2^b - 1$ integer. At the end the k^{th} term should be equal to the non-generated integer, probability of this is $\frac{1}{2^b}$. From the Equation 2.22 we get;*

$$P_K(\tilde{S}_k = k) = \frac{1}{2^b}\frac{(2^b-1)!}{2^{b^{k-1}}}\binom{2^b}{2^b-1}\left\{ {k-1 \atop 2^b-1} \right\}. \tag{2.23}$$

The saturation point test is one of the most appropriate tests for this method. However, in a binary sequence, there will be no saturation point. For this test, this should be considered. In this thesis, we overcome this problem by choosing a fixed-length index t if saturation point did not happen until index t partition the sequence and started the random variable again.

For computing bin probabilities the Equation 2.23 can be used. For example, to get the first bin probability $P(k = 256) + P(k = 257) + \ldots + P(k = 1236)$ should be computed.

Table 2.15: Bin values of Dynamic Partition Saturation Point Test for $b = 8$

bins	1	2	3	4	5	6	7	8
index range	[256-1236]	[1237-1337]	[1338-1424]	[1425-1511]	[1512-1608]	[1609-1732]	[1733-1927]	[1928]
probabilities	0.125207	0.123911	0.125964	0.125170	0.124287	0.125269	0.125292	0.124901

2.2.6 Dynamic Partition Index Coincidence Test

One of the new tests is proposed in this theses is Index-Coincidence Test (ICT).

Definition 2.2.27. The point where the entry of the sequence and the index of the sequence coincide is called index-coincidence.

Due to the structure of the test, only the dynamic method was used. In fact, the entire sequence testing test is also valid for this test. Even though the random variable split the sequence, it is treated as a whole. This test aims to evaluate the correlation between the index and the generated integer.

Collision tests perfectly fit this new approach. To vary such tests, we defined ICT. The ICT focuses on determining if a frequency of stop points defined below matches the expected $1's$. The test procedure can be summarized as follows:

- **Step 1:** This test requires an integer-valued sequence. However, if the sequence S is binary, one can convert this sequence into $\tilde{S}$ as described. It is proposed that $b = 8$ for long sequences.
- **Step 2:** Consider the $2^b + 1$ possible occurrences of index-value coincidences and determine the frequencies of different index-coincidence points that appear across the entire sequence.

- **Step 3:** Compute the *p-value* of test using the exact distribution of the test statistic.

Definition 2.2.28. The sequence $\tilde{S}$ is formed by the elements in $I = \{0, 1, 2, \ldots, 2^b - 1\}$. The assignment in I to $\tilde{S}$ is set as follows. $\tilde{s_1} \leftrightarrow 1, \tilde{s_2} \leftrightarrow 2, \ldots, \tilde{s_i} \leftrightarrow i$ in order. The case $\tilde{s_i} = i$ is called **i-index-value coincidence(i-ivc).** In that case, the assignment starts with the remaining part of the sequence whose indexes start from 0. If there is no index-value coincidence when all elements in s have been assigned, the assignment starts.

Example 2.2.29. Let $\tilde{S} = 37652144312320354$ be a sequence of length 16 and having $b = 3$. Then,

Table 2.16: Example

$\tilde{s}_0$	$\tilde{s}_1$	$\tilde{s}_2$	$\tilde{s}_3$	$\tilde{s}_4$	$\tilde{s}_5$	$\tilde{s}_6$	$\tilde{s}_7$	$\tilde{s}_0$	$\tilde{s}_1$	$\tilde{s}_0$	$\tilde{s}_1$	$\tilde{s}_2$	$\tilde{s}_0$	$\tilde{s}_0$	$\tilde{s}_1$	$\tilde{s}_2$
3	7	6	5	2	1	4	4	3	1	2	3	2	0	3	5	2
0	1	2	3	4	5	6	$\underbrace{7}_{8-ivc}$	0	$\underbrace{1}_{1-ivc}$	0	1	$\underbrace{2}_{2-ivc}$	$\underbrace{0}_{0-ivc}$	0	1	2

Computation of Probabilities

For computing the theoretical probability of occurrence of index-value coincide in a random sequence, the following functions had been defined:

$T(n)$: *the probability of occurrence of index-value coincidence or termination of assignment of elements of S without any coincidence.*

$T_i(n)$:*the probability of occurrence of i-index-value coincidence in sequence of length n.*

$T_{2^b}(n)$:*the probability of termination of assignment without any coincidence in a sequence of length n.*

The recurrence relation for $T(n)$:

$$\begin{aligned} T(n) =& T(n-1)p_0 + T(n-2)(1-p_0)p_1 + \ldots + T(n-k)(1-p_0)(1-p_1) + \ldots \\ &+ (1-p_k-1)p_k + (T(n-k)(1-p_0)(1-p_1)\ldots(1-p_{k-1})(1-p_k) \end{aligned}$$

where $p_i = \frac{1}{2^b}$ is the probability of $\tilde{s_k} = i$ (at index k) since there exists 2^b different integers.

Claim 2.2.30. Let t_i be number of i-index-value coincidences for $i \in SI$ and t_{2^b} be number of terminations of index assignment without index-value coincidence.Then the number of all index-value coincidence stops is $t_t = \sum_{i=0}^{2^b} t_i$. For a long sequence(sufficiently large n), $T(n)$ converges to

$$T(n) \approx \frac{t_0}{t_1} + \frac{t_1}{t_1} + \ldots + \frac{t_{2^b}}{t_1} \approx 1.$$

We use χ^2 as reference distribution. We form $2^b + 1$ bins so that each bin shows the number of index-value coincidences at corresponding value. In other words i^{th} bin measures t_i where $i = 0, 1, \ldots, 2^b$.

To apply $\chi^2 - distribution$, we need to evaluate expected numbers E_i for each bin. You can see the bins in Table 2.17

Table 2.17: Bin values of Dynamic Partition Index Coincidence Test

0	1	2	$\ldots$				2^b
E_0	E_1	E_1	$\ldots$				E_{2^b}
t_0	t_1	t_2	$\ldots$				t_{2^b}

Let $T = T_0 + T_1 + \ldots + T_{2^b}$. Note date t is not necessarily 1. The probability $\tilde{s_k}$ (for some random k) has i-index-value coincidence where $i \in I$ as follows.

$$\begin{aligned} Prob(\tilde{s_k} \in T_0) &= T(\frac{1}{2^b}) \\ Prob(\tilde{s_k} \in T_1) &= T(1 - \frac{1}{2^b}\frac{1}{2^b}) \\ &\vdots \\ Prob(\tilde{s_k} \in T_{2^b}) &= T(\frac{2^b - 1}{2^b})^{2^b}. \end{aligned}$$

Remark 2.2.31. $\sum_{j=0}^{2^b} Prob(\tilde{s_k} \in T_j) = T$

Since the length of the sequence is sufficiently large, T can be taken as $T = \frac{t_t}{|\tilde{S}|}$. Then expected values computed as follows:

$$\mu_{t_i} = E_i = \frac{t_t}{|\tilde{S}|}(\frac{2^b - 1}{2^b})^i(\frac{1}{2^b})|\tilde{S}| = t_t(\frac{2^b - 1}{2^b})^i(\frac{1}{2^b})$$

where $i = 0, 1, \ldots, 2^b - 1$ and

$$\mu_{t_{2^b}} = E_{2^b} = t_t(\frac{2^b - 1}{2^b})^{2^b}.$$

Hence, $\chi^2 - distribution$ is applied with degree of freedom 256. We can decrease the number of bins by making groups depends on the values of i.

CHAPTER 3

APPLICATIONS AND CORRELATIONS

In this chapter, applications and the mutual correlation of the tests are given. For evaluating propose randomness test suite, we fix significance level α as 0.01, determination of $alpha$ depend on the user. A random sequence should pass all propose tests except the auto-correlation test to be accepted as look random. For the auto-correlation test, random sequence fails at most $b * 0.01$ test, where b is the necessary variable for the auto-correlation test. In this chapter, while we test sequences, we use test parameters as $b = 8$ for integer tests, $b = 256$ for fixed-length partition weight, run, and linear complexity tests, and $b = 128$ for dynamic partitioning weight, run, and linear complexity tests. For different parameters, tests results will change.

Remark 3.0.1. If it is desired to test with different parameters, bin bounds and probabilities should be calculated using probability distribution functions in the required tests.

3.1 Applications

3.1.1 Test Performance and Proposed Test Suites

In this section, we give the time spent for each test and propose two different types of test suites.

In this test suite, we propose fifteen statistical randomness tests for long sequences. To find time spent, we test different lengths of sequences. To get test results we use *ASUS N752VX Intel Core i7 6700HQ 2.6GHZ-24GB RAM*. The test codes are not

Table 3.1: Time Spent for Each Test in Seconds

Length of the sequence	10^6	$2 * 10^6$	$2^2 * 10^6$	$2^4 * 10^6$	$2^5 * 10^6$
E. Weight	0.05	0.08	0.12	0.29	0.63
E. Total Run	3.80	10.75	32.01	337.56	1176.34
E. R2 run	2.31	5.51	24.34	245.37	800.08
Fixed Length Weight ($b = 256$)	0.05	0.08	0.12	0.30	0.65
Fixed Length Run ($b = 256$)	0.21	0.36	0.72	1.95	3.96
Fixed Length LC ($b = 256$)	0.45	0.92	1.73	5.41	10.29
Fixed Length Coverage ($b = 8$)	0.05	0.08	0.12	0.29	0.63
Dynamic Weight ($b = 128$)	0.05	0.09	0.13	0.30	0.64
Dynamic Run ($b = 128$)	3.210	11.40	30.08	258.24	952.60
Dynamic LC ($b = 128$)	26.53	53.55	118.60	796.51	2016.56
Dynamic Saturation ($b = 8$)	0.05	0.09	0.13	0.72	0.94
Dynamic Collision Index ($b = 8$)	0.06	0.10	0.13	0.72	0.94
Dynamic Collision Distance ($b = 8$)	0.06	0.10	0.13	0.72	0.94
Dynamic ICP ($b = 8$)	0.06	0.10	0.13	0.72	0.94
Entire Auto-Correlation ($b = 256$)	1.13	1.28	3.03	9.82	17.92

optimized. From Table 3.1 time spent in seconds can be seen.

The dynamic linear complexity test worked as slow as we expected. However, the entire sequence total number of runs, R2 runs, and dynamic partitioning run tests will work faster in optimizing codes.

According to Table 3.1, two different test suites should includes following test 3.2. Time spent comparisons of Test suites are given in Table 3.3. According to our results, the proposed super lightweight test suite will work faster without optimized codes. On the other hand, the lightweight test suite works slower than other suites. However, with optimized codes, it will work better.

3.1.2 Test Results of Random Sources

In this section, sequences, generated by some random number sources, are tested by the proposed suites. When we propose a test, our first goal is not to test generators. Because of this reason, we give the result of known sources for giving test data. In this thesis we use two cryptographic algorithm, AES [5], SHA256 [31], and two true random source π and $\sqrt{2}$. For each source, random sequences of length 2^{21} bit are

Table 3.2: Proposed Test Suites

Super Lightweight Test Suite	Lightweight Test Suite
Entire Sequence Weight	Addition to Super Lightweight Test Suite
Fixed Length Weight	Entire Sequence Total Number of Runs
Fixed Length Run	Entire Sequence R2 Run
Fixed Length LC	Dynamic Run
Fixed Length Coverage	Dynamic Linear Complexity
Dynamic Weight	
Dynamic Saturation	
Dynamic Collision Index	
Dynamic Collision Distance	
Dynamic ICP	
Entire Auto-Correlation	

Table 3.3: Time Spent Comparisons of Test Suites

	10^6	$2 * 10^6$	$2^2 * 10^6$	$2^4 * 10^6$	$2^5 * 10^6$
Super Lightweight	0.75	1.55	3.14	12.55	24.50
Lightweight	38.6716	82.30	200.18	1298.03	3443.02
NIST Test Suite	6.29	12.19	24.24	96.34	194.78

generated. For all tests, required parameters are defined according to the tables given in the test in this thesis.

Table 3.4: Test results of random sources

	SHA256	$AES-ECB_1$	$AES-ECB_2$	$AES-ECB_3$	$AES-ECB_4$	$AES-ECB_{10}$	$AES-CBC_1$	$AES-CBC_2$	$AES-CBC_{10}$	e	π
Entire-Weight	0.865798	7.47E-39	0.075575	0.997179	0.597845	0.456096	1.07E-22	0.542644	0.702582	0.813847	0.863018
Entire Total Run	0.575942	0	0	0.96165	0.887537	0.721555	0.627623	0.885862	0.811104	0.920581	0.787616
Entire R2 run	0.478541	0	0	0.89989	0.164497	0.373823	9.49E-82	0.956374	0.465059	0.088629	0.910985
fixed-length weight	0.398372	0	8.51E-22	0.56667	0.496774	0.136412	3.58E-75	0.352512	0.525168	0.967253	0.675774
fixed-length total run	0.781195	0	0	0.047491	0.100653	0.760978	0.628531	0.848256	0.825739	0.901402	0.802514
fixed-length LC	0.476056	0.250141	0.197791	0.544145	0.894592	0.470508	0.744748	0.891335	0.270308	0.321771	0.247618
fixed-length coverage	0.674674	0	0	4.10E-67	0.523469	0.054655	2.71E-12	0.484274	0.412321	0.045151	0.284231
dynamic weight	0.243881	0	5.00E-19	0.658786	0.72713	0.082498	1.95E-73	0.108468	0.235192	0.707463	0.198469
dynamic total run	0.8295	0	0	0.437743	0.71094	0.641365	0.006236	0.794497	0.820097	0.815136	0.790355
dynamic LC	0.534411	0.580216	0.597901	0.793294	0.047608	0.021774	0.140781	0.687511	0.195195	0.597192	0.225064
dynamic saturation	0.639852	7.19E-191	7.19E-191	1.89E-35	0.650671	0.360752	0.001035	0.925122	0.488234	0.19936	0.658515
dynamic collision index	0.203426	0	0	0.390992	0.741874	0.679007	5.55E-23	0.108887	0.5611	0.410391	0.131045
dynamic collision distance	0.548477	0	0	0.000648	0.006214	0.244448	2.17E-11	0.013145	0.194122	0.088347	0.846566
dynamic ICP	0.726868	5.16E-201	0	0.278581	0.44067	0.185368	5.10E-13	0.062173	0.943476	0.873902	0.135132
E. auto-correlation(rate)	$256/256$	$20/256$	$28/256$	$255/256$	$256/256$	$256/256$	$169/256$	$256/256$	$256/256$	$256/256$	$256/256$

From Table 3.4, except reduced round AES outputs, all random sources pass the tests. Different length sequences are tested in same way and we always get the same results.

3.2 Correlations of Test

In this section, Pearson correlations of proposed tests are given [32]. For discovering correlations of proposed tests, 1000 random number sequences of length $4,000,000$ are generated. In other words, approximately 4 billion bits are generated. For each

Table 3.5: Pearson correlation results of tests

	E. Weight	E.T. Run	E. R2. Run	F.L Weight	F.L Run	F.L LC	F.L Cov.	Dyn. Weight	Dyn. Run	Dyn. LC	Dyn.Sat.	Dyn.Coll.İnd	Dyn.Coll.Dist.	Dyn. ICP
E. Weight	1.0000	0.0204	0.0644	0.3210	0.0301	-0.0183	-0.0601	0.3308	0.0211	0.0119	-0.0361	0.0050	-0.0261	-0.0529
E.T. Run	0.0204	1.0000	-0.0098	0.0326	0.2870	-0.0209	-0.0108	-0.0260	0.2813	0.0259	0.0120	0.0312	0.0603	0.0142
E. R2 Run	0.0644	-0.0098	1.0000	0.0245	0.0094	-0.0016	0.0148	-0.0112	0.0079	-0.0028	0.0005	-0.0487	0.0219	0.0023
F.L Weight	0.3210	0.0326	0.0245	1.0000	0.0080	-0.0089	-0.0252	0.1711	0.0160	-0.0260	-0.0469	0.0313	-0.0243	-0.0046
F.L Run	0.0301	0.2870	0.0094	0.0080	1.0000	-0.0561	0.0599	0.0153	0.1883	0.0016	0.0789	-0.0043	-0.0227	0.0085
F.L LC	-0.0183	-0.0209	-0.0016	-0.0089	-0.0561	1.0000	0.0020	-0.0312	-0.0060	-0.0081	-0.0014	0.0092	-0.0064	0.0027
F.L Cov.	-0.0601	-0.0108	0.0148	-0.0252	0.0599	0.0020	1.0000	0.0017	0.0115	0.0136	0.0510	-0.0520	0.0206	0.0462
Dyn. Weight	0.3308	-0.0260	-0.0112	0.1711	0.0153	-0.0312	0.0017	1.0000	0.0224	-0.0046	0.0032	-0.0179	-0.0325	-0.0332
Dyn. Run	0.0211	0.2813	0.0079	0.0160	0.1883	-0.0060	0.0115	0.0224	1.0000	-0.0239	0.0196	0.0054	0.0317	0.0538
Dyn. LC	0.0119	0.0259	-0.0028	-0.0260	0.0016	-0.0081	0.0136	-0.0046	-0.0239	1.0000	-0.0216	-0.0161	0.0088	0.0279
Dyn. Sat.	-0.0361	0.0120	0.0005	-0.0469	0.0789	-0.0014	0.0510	0.0032	0.0196	-0.0216	1.0000	-0.0712	0.0016	-0.0320
Dyn.Coll.İnd.	0.0050	0.0312	-0.0487	0.0313	-0.0043	0.0092	-0.0520	-0.0179	0.0054	-0.0161	-0.0712	1.0000	0.0497	-0.0015
Dyn.Coll.Dist	-0.0261	0.0603	0.0219	-0.0243	-0.0227	-0.0064	0.0206	-0.0325	0.0317	0.0088	0.0016	0.0497	1.0000	-0.0193
Dyn. ICP	-0.0529	0.0142	0.0023	-0.0046	0.0085	0.0027	0.0462	-0.0332	0.0538	0.0279	-0.0320	-0.0015	-0.0193	1.0000

generated sequence, *p-values* are evaluated from all proposed tests. While generating sequences, the random function of $c\#$ is used. For different random sequences from different sources, similar results will be obtained.

In Table 3.5 results of all tests are given. In Table 3.8, only constant partition method tests are given, and in Table 3.9 only dynamic partition method tests are given. Since the auto-correlation test generates more than one *p-value*, we does not apply the correlation test. When we evaluate the Pearson correlation, we say; for the results between $[-1.0, -0.50]$ negatively correlated, for the results between $(-0.50, -0.10]$ negatively half correlated, for the result between $(-0.10, 0.10)$ not correlated, for the results between $[0.10, 0.50)$ positively half correlated and for the results between $[0.50, 1.0]$ correlated.

From Table 3.5 the following correlations are found;

- Weight tests are positively half correlated with each other.
- Except from R2 run test, run tests are positively half correlated with each other.
- No significant negative correlation between any tests.
- No correlation between dynamic partitioning method tests (see Table 3.9).
- No correlation between fixed length partitioning method tests (see Table 3.8)

Table 3.6: Pearson correlation results of Weight Tests

	E.Weight	F.L Weight	Dyn. Weight
E. Weight	1.0000	0.3210	0.3308
F.L Weight	0.3210	1.0000	0.1711
Dyn. Weight	0.3308	0.1711	1.0000

Table 3.7: Pearson correlation results of Run Tests

	E. Total Run	E. R2 Run	F.L Run	Dyn. Run
E. Total Run	1.0000	-0.0098	0.2870	0.2813
E. R2 Run	-0.0098	1.0000	0.0094	0.0079
F.L Run	0.2870	0.0094	1.0000	0.1883
Dyn. Run	0.2813	0.0079	0.1883	1.0000

Table 3.8: Pearson correlation results of Fixed Length

	F.L weight	F.L run	F.L LC	F.L coverage
F.L weight	1.0000	0.0080	-0.0089	-0.0252
F.L run	0.0080	1.0000	-0.0561	0.0599
F.L LC	-0.0089	-0.0561	1.0000	0.0020
F.L coverage	-0.0252	0.0599	0.0020	1.0000

The correlation results are consistent with the results we expect. In other words, since weight tests (or run tests) in different methods use the same random variable, their test results are correlated. Using this result, we can conclude that the dynamic partitioning method, in which we test every term of the sequence, is useful for other tests.

Table 3.9: Pearson correlation results of Dynamic Partition

	Dyn. Weight	Dyn. Run	Dyn. LC	Dyn. Satur.P.	Dyn. Coll. İndex	Dyn. Coll.Dist.	Dyn. ICP
Dyn. Weight	1.0000	0.0224	-0.0046	0.0032	-0.0179	-0.0325	-0.0332
Dyn. Run	0.0224	1.0000	-0.0239	0.0196	0.0054	0.0317	0.0538
Dyn. LC	-0.0046	-0.0239	1.0000	-0.0216	-0.0161	0.0088	0.0279
Dyn. Saturation	0.0032	0.0196	-0.0216	1.0000	-0.0712	0.0016	-0.0320
Dyn. Coll. İndex	-0.0179	0.0054	-0.0161	-0.0712	1.0000	0.0497	-0.0015
Dyn. Coll. Distance	-0.0325	0.0317	0.0088	0.0016	0.0497	1.0000	-0.0193
Dyn. ICP	-0.0332	0.0538	0.0279	-0.0320	-0.0015	-0.0193	1.0000

CHAPTER 4

SENSITIVITIES OF TESTS AND COMPARISON OF TEST SUITES

In this chapter, sensitivities of tests to bias sources and comparison of the result propose test suite and NIST test suite is given. Since the proposed tests are designed for testing long sequences, we use default parameters for NIST tests and get a single *p-value* for each test for each sequence.

4.1 Test Results of Bias Sources

In this section, biased number sources is tested by the proposed test suite to evaluate the sensitivities of the tests.

Case 1 : Probability of generating 1 and 0

In a random binary sequence, it is expected that ones and zeros are generated with equal probability. In this case, to determine to evaluate sensitivities, the same random source is manipulated by generating 1 with probability $\frac{1}{2} + p$ where $-\frac{1}{2} \leq p \leq \frac{1}{2}$. In other words, we generated sequences with the expected weight of the sequence is not equal to $\frac{1}{2}$. While testing this type of non-random source, we expected that generated sequence fails from weight tests. However, it is significant that all weight tests have similar sensitivities. For determining the sensitivities, 16000000-bit non-random sequences are generated with the probability of generating 1 is $\frac{1}{2} + p$.

From Table 4.1, among all methods, the weight test is the first test detecting the bias as expected. All tests except the linear complexity tests detected non-random sequences

Table 4.1: Test results of biased sources

p	0.00001	0.0001	0.001	0.01	0.02	0.03	0.05
E. Weight	0.547673	0.80665	0.000757	0	0	0	0
E. Total Run	0.638712	0.616723	0.65271	0.192745	0.000329	0	0
E. R2 run	0.910636	0.919032	0.938921	0.087699	2.77E-16	0	0
Fixed Length Weight	0.499011	0.634375	1.61E-08	0	0	0	0
Fixed Length Run	0.331737	0.300516	0.631796	0.135081	4.28E-08	0	0
Fixed Length LC	0.420019	0.298433	0.716327	0.065398	0.168809	0.40643	0.965647
Fixed Length Coverage	0.949513	0.942091	0.980447	0.26261	0	0	0
Dynamic Weight	0.965092	0.491077	5.90E-08	0	0	0	0
Dynamic Run	0.642842	0.721832	0.975537	0.019528	1.03E-09	0	0
Dynamic LC	0.246773	0.328929	0.887367	0.116419	0.528365	0.374191	0.038154
Dynamic Saturation	0.631816	0.855693	0.895069	0.033071	0	0	0
Dynamic Collision Index	0.40568	0.373535	0.028718	0	0	0	0
Dynamic Collision Distance	0.909108	0.907454	0.799053	0.877146	0.294141	8.72E-10	0
Dynamic ICP	0.58621	0.836374	0.11587	0.059847	2.37E-07	0	0
E. Auto-Correlation(pass rate)	256/256	256/256	256/256	256/256	39/256	0/256	0/256

when the bias is 0.02. Because of the test structure, linear-complexity tests could not detect this type of biased sequence. The fail bias of all tests in similar points shows that the tests and methods have similar sensitivities to this type of bias.

Case 2 : Changing generating template

In this case, we give bias to generating templates. For example, for each 10^{th}"1000" template, we change it with an "1100" template. In this thesis, we change different templates for different times. For generated non-random sequences, we use the first 4000000 bit of π. For different cases, we change non-overlapping templates. It is expected that all templates will be generating with equal probability. Because of this reason, we generate the chosen template with greater probability, and we generate the corresponding template with lower probability. The sensitivity of the tests varies according to the changed template. For example, when the weight of templates is different, non-random sequences failed from weight tests. In other cases, non-random sequences pass the weight tests. The results are given in Table 4.2. Table 4.2, and other experimental results show that non-random sequences of this type fail from the R2 run test. If the probabilities of generating templates have close probabilities, then tests do not fail. If the biased templates affect the number of runs, then other run tests, collision index test, and index coincidence point test fail.

Remark 4.1.1. $(t{*}xx\ldots x{-}yy\ldots y)$ means, for each t^{th} $xx\ldots x$ template generated change it with $yy\ldots y$ template along the sequence.

Table 4.2: Test results of biased template sequences

	π4m(3*01110000 -00111000)	π4m(5*00001000 -00000010)	π4m(10*0001-0010)	π4m(20*0001-0010)
E. Weight	0.761508	0.761508847	0.761508847	0.761509
E. Total Run	0.953748	0.953748629	1.49E-19	4.97E-10
E. R2 Run	2.78E-12	2.52E-32	1.46E-68	2.95E-16
F.L Weight	0.766230653	0.794147626	0.842721336	0.856401
F.L Run	0.467805	0.978681911	9.08E-111	7.56E-16
F.L LC	0.7695767	0.709798666	0.180798564	0.55676
F.L coverage	0.3860949	0.131851751	0.394313052	0.148469
Dyn. Weight	0.351313	0.325022708	0.271946719	0.289364
Dyn. Run	0.848894	0.812718761	9.48E-111	3.96E-16
Dyn. LC	0.52973	0.857565413	0.71828367	0.665208
Dyn. Saturation	0.301674	0.80528541	0.367098526	0.154132
Dyn. Coll. Index	0.012233	0.090185203	1.52E-09	0.000613
Dyn. Coll. Distance	0.2900747	0.640132634	0.315079052	0.379265
Dyn.. ICP	0.306376207	0.868683345	1.45E-01	0.047406
Auto Correlation	255/256	255/256	254/256	254/256

Case 3 : Repeating sequence with long period

In this case, we use two different methods; one repeats the random sequence, and the other repeats the random sequence with each term of the sequence complemented. For example, if the sequence is equal to "100101", for the first case, the non-random sequence is equal to "100101100101," and for the second case, the non-random sequence is equal to "100101011010". For this case, we use different lengths of sequences generated from π. It is expected that a random sequence has a significant period. The reason for generating this non-random sequence is to detect sensitivities of the proposed test for a short period.

From Table 4.3, for different non-random sequences of different lengths which are generated by repeating itself, at least one of the proposed tests fails. Especially, index point coincidence tests detect this type of non-randomness.

From Table 4.4, for different non-random sequences of different lengths which are generating by repeating complement of the sequence, at least one of the proposed tests fails.

By repeating different parts in sequences of different lengths, the non-random sequences are generating. Some tests, especially coverage, saturation point, and collision tests, detect the non-random sequences if the period is short.

Table 4.3: Test results of repeating sequences

	π(2m+2m)	π(2m+1m)	π(4m+4m)	π(4m+2m)
E. Weight	0.739123	0.975128	0.667767	0.701161
E. Total run	0.887474	0.87931	0.934627	0.991856
E. R2 run	0.739134	0.403088	0.525452	0.642347
F.L Weight	0.497328	0.671859	0.397206	0.738422
F.L Run	0.848001	0.665962	0.05919	0.584432
F.L LC	0.037603	0.096186	0.654556	0.315588
F.L Coverage	0.001103	0.01482	0.000686	0.000392
Dyn. Weight	0.719222	0.898615	0.232539	0.364713
Dyn. Run	0.420331	0.805917	0.957034	0.826496
Dyn. LC	0.104981	0.410588	0.060226	0.199087
Dyn. Saturation	0.006804	0.050247	0.784898	0.359986
Dyn. Coll. index	0.045062	0.35241	0.000529	0.007022
Dyn. Coll. distance	0.000904	0.034323	0.021276	0.024306
Dyn. ICP	0	4.21E-07	0	3.05E-06

Table 4.4: Test results of repeating complement sequences

	π(2m+2m)	π(4m+2m)	π(4m+4m)	e(4m+4m)	e(4m+2m)
E. Weight	1	0.910934	1	1	0.746134
E. Total run	0.887869	0.991531	0.934908	0.788704	0.756356
E. R2 run	0.066397	0.641874	0.525182	0.009200	0.034696
F.L. Weight	0.841868	0.935787	0.918687	0.105233	0.191092
F.L. Run	0.848001	0.584432	0.05919	0.613499	0.659686
F.L. LC	0.128253	0.589482	0.801001	0.58827	0.594375
F.L. Coverage	0.001103	0.000456	0.000757	0.205925	0.252846
Dyn. Weight	0.803966	0.379659	0.3406745	0.05455	0.104959
Dyn. Run	0.434369	0.831704	0.95146	0.922812	0.904922
Dyn. LC	0.284296	0.173834	0.2743891	0.981879	0.975726
Dyn. Saturation	0.006917	0.374526	0.783355	0.208226	0.334155
Dyn. Coll. index	0.364399	0.088629	0.027327	0.00088	0.007712
Dyn. Coll. distance	0.000863	0.024306	0.021276	0.642837	0.719439
Dyn. ICP	0.288143	0.602396	0.730529	0.339384	0.145964

4.2 Comparison of Tests

In this section, the proposed tests and tests in the NIST test suite are compared. Test results ofπ and e, of length 16000000, and the biased sequence results are given in Table 4.5.

From Tables 4.5, 4.1 and 4.3, non-random sequences are failed for some tests in both suites. In the NIST test suite, non-random sequences are failed from the approximate entropy test and serial test as expected. For the proposed tests, sequences are failed from some dynamic method tests. From Tables 4.5 and 4.4, non-random sequences passed the NIST test suite. For proposed tests, sequences fail at least one test.

From Tables 4.6 and 4.2, non-random sequences are failed for some tests in both suites. In the NIST test suite without non-overlapping template matching, test sequences pass the tests. For proposed tests, non-random sequences are failed from R2 run test.

From the comparison, the following conclusions are made:

- Weight tests and run tests in NIST test suites are very roughly calculated. The proposed versions of weight tests and run tests become sensitive.
- Proposed integer tests are significant to find periodic sequences.
- Using the suggested tests in test suites may affect the test results.

Table 4.5: NIST test suite results

	π16m	16m	π 4m+π 4m	π 4m+π+1 2m	π 4m+π+1 4m	e 4m+e+1 4m	e 4m+e+1 2m	bias0.00001	bias0.0001	bias0.001	bias0.01
App Ent.	0.673321	8.843267	0	0.995689	0.961303	0.515667	0.798712	0.926638	0.919633	0.66378	0
block freq	0.317426	0.349606	0.063889	0.065913	0.063889	0.186986	0.247547	0.834348	0.829309	0.823249	0
CUSUM1	0.0834	0.966648	0.344156	0.556993	0.688386	0.736192	0.608994	0.250297	0.781625	0	0
CUSUM2	0.09682	0.96187	0.449681	0.762346	0.688386	0.736192	0.793969	0.312923	0.880578	0	0
frequency	0.084889	0.721092	0.390656	0.827975	1	1	0.51732	0.22917	0.624488	0	0
linearC.	0.086092	0.523025	0.038915	0.07041	0.189268	0.477112	0.257603	0.535236	0.511801	0.099133	0.621461
longest run	0.905001	0.939874	0.007627	0.094659	0.044635	0.15134	0.3816	0.83335	0.79915	0.613254	0
Non-Overlapping TM	145/147	147/147	134/147	144/147	145/147	147/147	147/147	147/147	147/147	147/147	35/147
Overlapping TM	0.077454	0.494517	0.260901	0.426579	0.289271	0.102043	0.172027	0.559957	0.548902	0.294963	0
R. Excursion V(-1)	0.838724	0.955627	0.838724	0.838724	0.164205	0.473296	0.698551	0.303803	0.238035	NA	NA
R. Excursion V(-2)	0.066521	0.0719915	0.66521	0.6521	0.238889	0.580376	0.421132	0.353353	0.632885	NA	NA
R. Excursion V(-3)	0.872502	0.961855	0.872502	0.872502	0.42613	0.850536	0.955509	0.972404	0.749159	NA	NA
R. Excursion V(-4)	0.076942	0.41475	0.076942	0.076942	0.088969	0.081184	0.259401	0.94828	0.326092	NA	NA
R. Excursion V(1)	0.4843	0.164315	0.4843	0.483	0.077178	0.327657	0.735338	0.689247	0.536326	NA	NA
R. Excursion V(2)	0.287652	0.680462	0.287652	0.287652	0.784358	0.138689	0.648486	0.822329	0.684887	NA	NA
R. Excursion V(3)	0.150719	0.356663	0.150719	0.150719	0.318133	0.792802	0.839016	0.209209	0.861968	NA	NA
R. Excursion V(4)	0.093088	0.228798	0.093088	0.093088	0.923043	0.27491	0.337083	0.487469	0.855487	NA	NA
R.E.Variant	18/18	18/18	18/18	18/18	18/18	18/18	18/18	18/18	18/18	NA	NA
run	0.743863	0.296471	0.870455	0.982395	0.870806	0.592456	0.535554	0.348173	0.317098	NA	NA
binary matrix rank	0.92969	0.566379	0.512248	0.287817	0.274832	0.782794	0.898147	0.838937	0.63319	0.550551	0.680892
Serial1	0.734071	0.438843	0	0.615485	0.589229	0.196616	0.572886	0.83616	0.841721	0.760733	0
Serial2	0.971832	0.539644	0	0.512372	0.648968	0.418589	0.399348	0.842891	0.827452	0.864801	0.634761
universal	0.765338	0.156516	0.228463	0.225479	0.226388	0.588435	0.479789	0.963884	0.991721	0.928601	0.001982

Table 4.6: NIST test suite results of biased template sequences

	π4m(3*01110000-00111000)	π4m(5*00001000-00000010)	π4m(10*0001-0010)	π4m(20*0001-0010)
App Ent.	0	0	0	0
block freq	0.153092	0.13298	0.149941	0.149941
CUSUM1	0.370864	0.370864	0.370864	0.370864
CUSUM2	0.533527	0.533527	0.533527	0.533527
frequency	0.543851	0.543851	0.543851	0.543851
linearC.	0.73799	0.44827	0.742285	0.741376
longest run	0.1898291	0.188291	0.205234	0.188291
Non-Overlapping TM	135/147	121/147	64/147	107/147
Overlapping TM	0.590462	0.590462	0.471968	0.667088
R. Excursion V(-1)	0.790082	0.796378	0.820586	0.821615
R. Excursion V(-2)	0.834858	0.904335	0.80511	0.76303
R. Excursion V(-3)	0.744884	0.876281	0.785522	0.843112
R. Excursion V(-4)	0.122652	0.064636	0.223694	0.098602
R. Excursion V(1)	0.609301	0.447682	0.835204	0.640652
R. Excursion V(2)	0.271901	0.329291	0.268735	0.319342
R. Excursion V(3)	0.177687	0.175985	0.240543	0.274554
R. Excursion V(4)	0.108394	0.097079	0.08507	0.134298
R.E.Variant	18/18	18/18	18/18	18/18
run	0.908591	0.908591	0	0
binary matrix rank	0.084759	0.825892	0.534422	0.919951
Serial1	0	0.0007	0	0.014521
Serial2	0.668664	0.863105	0.58817	0.574905
universal	0.06759	0.043544	0.744355	0.08562

CHAPTER 5

CONCLUSION

Randomness and random number generation are essential concepts. Determining whether a sequence is not distinguishable from a true random number sequence can be challenging. For overcoming this problem, statistical randomness tests are used. There are lots of statistical randomness tests and tests suites in the literature. However, although some of them can test long sequences after some approximation or modification, none are customized to test long sequences. In this thesis, we propose a lightweight test suite to evaluate long sequences. To propose this test suite, we use three different methods. From the literature, weight, run, and auto-correlation tests propose to test the entire sequence without partitioning. Moreover, the entire R2 run test is defined to test the entire sequence without partitioning. For testing fixed-length sub-sequences of a long sequence, weight, run, linear complexity, and integer coverage tests are used. To complete the test suite, we use a new method called the dynamic partitioning method. Weight, run, collision, linear complexity, saturation point, and index coincidence point tests are proposed for the dynamic partitioning method.

In this thesis, random sources are tested with the test suite. The results of the tests are given. Moreover, non-random biased and modified sequences are tested to find the sensitivities of each test. The Pearson correlation is employed to show the correlation between the methods and tests in the test suite. In the end, the results of NIST and test suite is compared. It has been observed that the tests we propose in this paper give better results due to some modifications. Since the main motivation of this thesis is to propose a lightweight test suite to test long sequences, we only give results with selected parameters that tests in each method will be compatible with other methods.

For some other parameters, tests result can be better.

It should not be forgotten in the literature; many tests can be applied to the methods described. Appropriate tests can be selected for testing long sequences for future work, and tests should be adapted according to the methods. In this thesis, we have selected tests that are comprehensive and straightforward and introduced methods. Thus, we contribute literature for evaluating long sequences. We propose two lightweight test suites, and we give the time performance of these suites depending on our codes. For future work, optimization of codes can be done, and tests that can be added to the super lightweight test suite will be proposed. In addition to these, sensitivities of proposed tests to transformed sequences can be evaluated for future work.

REFERENCES

[1] Z. Akcengiz, Mutual correlation of randomness test and analysis of test outputs of transformed and biased sequences, Msc. Thesis, Ankara: METU, 2014.

[2] P. Alcover, A. Guillamón, and M. Ruiz, A new randomness test for bit sequences, Informatica (Netherlands), 2013.

[3] E. R. Berlekamp, *Algebraic Coding Theory - Revised Edition*, World Scientific Publishing Co., Inc., USA, 2015, ISBN 9789814635899.

[4] R. G. Brown, Dieharder: A random number test suite, https://webhome.phy.duke.edu/ rgb/General/dieharder.php, 2013.

[5] J. Daemen and V. Rijmen, *The Design of Rijndael, AES - The Advanced Encryption Standard*, Springer-Verlag Berlin Heidelberg, 2002.

[6] A. Doğanaksoy and F. Göloğlu, On lempel-ziv complexity of sequences, Sequences and Their Applications- SETA, Springer Berlin Heidelberg, pp. 180–189, 2006.

[7] A. Doğanaksoy, F. Sulak, M. Uğuz, O. Şeker, and Z. Akcengiz, New statistical randomness tests based on length of runs, Mathematical Problems in Engineering, Hindawi Publishing Corporation, 2015.

[8] A. Doğanaksoy, F. Sulak, M. Uğuz, O. Şeker, and Z. Akcengiz, Mutual correlation of nist statistical randomness tests and comparison of their sensitivities on transformed sequences, Turkish Journal of Electrical Engineering and Computer Sciences, 25, pp. 655–665, 01 2017.

[9] C. Georgescu, E. Simion, A. Petrescu-Nita, and A. Toma, A view on nist randomness tests (in)dependence, 9th International Conference on Electronics, Computers and Artificial Intelligence, pp. 1–4, 2017.

[10] M. Gil, G. Gonnet, and W. Petersen, A repetition test for pseudorandom number generators, Monte Carlo Methods and Applications, 12, pp. 385–393, 2006.

[11] W. Golomb, *Shift Register Sequences*, Aegean Park Press, 1982.

[12] K. Hamano and T. Kaneko, Correction of overlapping template matching test included in nist randomness test suite, IEICE Transactions, 90-A, pp. 1788–1792, 09 2007.

[13] K. Hamano, F. Sato, and H. Yamamoto, A new randomness test based on linear complexity profile, IEICE Transactions, 92-A, pp. 166–172, 2009.

[14] K. Hamano and H. Yamamoto, A randomness test based on t-complexity, IEICE Transactions, 93-A, pp. 1346–1354, 07 2010.

[15] J. Hernandez-Castro and D. F. Barrero, Evolutionary generation and degeneration of randomness to assess the indepedence of the ent test battery, 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 1420–1427, 2017.

[16] J. Hernandez-Castro, J. Sierra, and A. Seznec, The sac test: A new randomness test, with some applications to prng analysis, Computational Science and Its Applications - ICCSA 2004, 3043, pp. 960–967.

[17] M. Herrero-Collantes and J. C. Garcia-Escartin, Quantum random number generators, Reviews of Modern Physics, 89, 2016.

[18] A. Iwasaki and K. Umeno, A new randomness test solving problems of discrete fourier transform test, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E101.A, 2017.

[19] J. A. Karell-Albo, C. M. Legón-Pérez, E. J. Madarro-Capó, O. Rojas, and G. Sosa-Gómez, Measuring independence between statistical randomness tests by mutual information, Entropy, 22, 2020.

[20] J. Kelsey, B. Schneier, D. Wagner, and C. Systems, Cryptanalytic attacks on pseudorandom number generators, Lecture Notes in Computer Science, 1372, 2000.

[21] M. G. Kendall and B. B. Smith, Randomness and random sampling numbers, Journal of the Royal Statistical Society, 101(1), pp. 147–166, 1938.

[22] D. E. Knuth, *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*, Addison-Wesley Longman Publishing Co., Inc., 1997.

[23] O. Koçak, A unified evaluation of statistical randomness tests and experimental analysis of their relations, Phd. Thesis, Ankara: METU, 2016.

[24] O. Koçak, F. Sulak, A. Doğanaksoy, and M. Uğuz, Modifications of knuth randomness tests for integer and binary sequences, Communications Faculty of Sciences University of Ankara Series A1 Mathematics and Statistics, 67, 2018.

[25] P. L'Ecuyer and R. Simard, Testu01: A c library for empirical testing of random number generators, 33, 2007.

[26] P. L'Ecuyer, Testing random number generators, Theory of Probability and Its Applications, 35, pp. 305–313, 1992.

[27] G. Marsaglia., The marsaglia random number cdrom including the diehard battery of tests of randomness, 1996.

[28] G. Marsaglia and A. Zaman, Monkey tests for random number generators, Computers Mathematics with Applications, 26, pp. 1 – 10, 1993.

[29] J. Massey, Shift-register synthesis and bch decoding, IEEE Transactions on Information Theory, 15(1), pp. 122–127, 1969.

[30] U. M. Maurer, A universal statistical test for random bit generators, Journal of Cryptology, 5, pp. 89 – 105, 1992.

[31] NIST, Secure hash standard (shs), fips pub 180-2, 2002.

[32] K. Pearson, *Notes on Regression and Inheritance in the Case of Two Parents*, volume 58, Proceedings of the Royal Society of London, 1895.

[33] A. Rukhin, Testing randomness: A suite of statistical procedures, Theory of Probability Its Applications, 45, 2000.

[34] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, M. L. Stefan Leigh, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo, A statistical test suite for random and pseudo random number generators for cryptographic applications.technical report, 2001.

[35] B. Ryabko, V. Stognienko, and Y. Shokin, A new test for randomness and its application to some cryptographic problems, Journal of Statistical Planning and Inference, 123, pp. 365 – 376, 2004.

[36] J. Soto, Statistical testing of random number generators, 1999.

[37] J. Soto, Randomness testing of the advanced encryption standard candidate algorithms, 03 2001.

[38] A. Srinivasan, M. Mascagni, and D. Ceperley, Testing parallel random number generators, Parallel Computing, 29, pp. 69 – 94, 2003.

[39] M. Stipčević and C. K. Koç, True random number generators, Open Problems in Mathematics and Computational Science, Springer International Publishing Switzerland, pp. 275–315, 2014.

[40] F. Sulak, Statistical analysis of block ciphers and hash functions, Phd. Thesis, Ankara: METU, 2012.

[41] F. Sulak, A new statistical randomness test: Saturation point test, International Journal of Information Security Science, 2, pp. 81 – 85, 2013.

[42] F. Sulak, New statistical randomness tests: 4-bit template matching tests, Turkish Journal of Mathematics, 41, pp. 80 – 95, 2017.

[43] F. Sulak, A. Doğanaksoy, M. Uğuz, and O. Koçak, Periodic template tests: A family of statistical randomness tests for a collection of binary sequences, DISCRETE APPLIED MATHEMATICS, pp. 191–204, 2019.

[44] F. Sulak, M. Uğuz, O. Koçak, and A. Doğanaksoy, On the independence of statistical randomness tests included in the nist test suite, Turkish Journal of Electrical Engineering and Computer Sciences, 25, pp. 3673–3683, 2017.

[45] M. Sýs and Z. Říha, Faster randomness testing with the nist statistical test suite, pp. 272–284, Security, Privacy, and Applied Cryptography Engineering, Springer International Publishing, 2014.

[46] M. Turan, A. Doğanaksoy, and S. Boztas, On independence and sensitivity of statistical randomness tests, volume 5203, pp. 18–29, 09 2008.

[47] M. S. Turan, E. Barker, J. Kelsey, K. A. McKay, M. L. Baish, and M. Boyle, Recommendation for the entropy sources used for random bit generation, NIST, 2018.

[48] M. Uğuz, A. Doğanaksoy, F. Sulak, and O. Koçak, R-2 composition tests: a family of statistical randomness tests for a collection of binary sequences, Cryptography and Communications, 2019.

[49] J. Walker, Ent. a pseudorandom number sequence test program, Software and documentation, 2008, [online] Available: http://www.fourmilab.ch/random.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Akcengiz, Ziya
Nationality: Turkish (TC)
Date and Place of Birth: 1989, Ankara
Marital Status: Married

EDUCATION

Degree	Institution	Year of Graduation
M.S.	Department of Cryptography, METU	2014
B.S.	Department of Mathematics, METU	2012
High School	Kalaba Anatolian High School	2007

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
12.2017-Continue	TUBITAK / UEKAE	Expert
07.2015-12.2017	METU / IAM	Research Assistant
11.2013-07.2015	METU	Scientific Project Expert

PUBLICATIONS

A. Doğanaksoy, F. Sulak, M. Uğuz, O. Şeker, Z. Akcengiz, *New Statistical Randomness Tests Based on Length of Runs* , Mathematical Problems in Engineering, Hindawi

Publishing Corporation, vol. 2015.

A. Doğanaksoy, F. Sulak, M. Uğuz, O. Şeker, Z. Akcengiz, *Mutual correlation of NIST statistical randomness tests and comparison of their sensitivities on transformed sequences*, Turkish Journal of Electrical Engineering & Computer Sciences, vol. 25, 2017.

Z. Akcengiz, M. Aslan, Ö. Karabayır, A. Doğanaksoy, M. Uğuz, F. Sulak, *Statistical Randomness Tests of Long Sequences by Dynamic Partitioning*, 2020 International Conference on Information Security and Cryptology (ISCTURKEY), Ankara, Turkey, 2020, pp. 68-74, doi: 10.1109/ISCTURKEY51113.2020.9308005.