A UNIFIED EVALUATION OF STATISTICAL RANDOMNESS TESTS AND EXPERIMENTAL ANALYSIS OF THEIR RELATIONS

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

BY

ONUR KOÇAK

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

SEPTEMBER 2016

Approval of the thesis:

A UNIFIED EVALUATION OF STATISTICAL RANDOMNESS TESTS AND EXPERIMENTAL ANALYSIS OF THEIR RELATIONS

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

Prof. Dr. Bülent Karasözen
Director, Graduate School of **Applied Mathematics** _____

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

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

Assist. Prof. Dr. Fatih Sulak
Co-supervisor, **Department of Mathematics, Atılım University** _____

Examining Committee Members:

Prof. Dr. Ersan Akyıldız
Department of Mathematics, METU _____

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

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

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

Assist. Prof. Dr. A. Nurdan Saran
Department of Computer Engineering, Çankaya University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: ONUR KOÇAK

Signature :

ABSTRACT

A UNIFIED EVALUATION OF STATISTICAL RANDOMNESS TESTS AND EXPERIMENTAL ANALYSIS OF THEIR RELATIONS

Koçak, Onur
Ph.D., Department of Cryptography
Supervisor : Assoc. Prof. Dr. Ali Doğanaksoy
Co-Supervisor : Assist. Prof. Dr. Fatih Sulak

September 2016, 82 pages

Random numbers are used in many applications in our daily life. For instance, when your mobile phone is registering a base station, base station sends a random number for authenticating your phone. Moreover, when logging in your e-mail or bank account your browser and the server exchange random numbers while establishing a handshake. Besides, encryption keys and IVs should be random so that no one can predict them without trying all possible values. The number of examples can be increased from many fields including cryptography, information theory, simulation and quantum theory.

Random number sequences are generated by the random number generators (RNG)[1]. Deterministic RNGs should be tested to make sure that the output sequences are indistinguishable from random sequences. Unfortunately, theoretic testing is not possible if the output sequences have very obvious relations which is not a usual case. Therefore, testing process is done statistically by applying randomness tests on the sequences and the results are evaluated to conclude the non-randomness of the generator. For the decision to be more reliable a set of tests called test suites are applied on the sequences.

Nearly all test suites uses the probabilities derived from the approximations of the distribution functions of the tests. As the approximations work for longer sequences,

[1] Throughout this thesis, random number generator name will also cover random sequence generators.

testing short sequences like keys or IVs becomes infeasible. Moreover, the relations among the tests, which affect the decision on the sequence or the generator, are not measured in any suite.

In this thesis, we examine the statistical randomness tests in the literature. We select the tests which are based on mathematical background and are important measures for randomness. Then, we review the distribution functions of these tests to compute the actual probability values. Moreover, we give recursions for the tests whose probability values cannot be computed for longer sequences. Afterwards we find the correlations between the tests and make a classification accordingly. Then, we give some rule of thumbs for designing a test suite and build a test suite consisting of the examined tests.

Keywords : Statistical Randomness Tests, Distribution Functions, Test Suites, Correlation, Classification

ÖZ

ISTATİKSEL RASTGELELİK TESTLERİNİN B UT UNLEŞİK BİR DEĞERLENDİRMESİ VE ARALARINDAKİ İLİŞKİLERİN DENEYSEL ANALİZİ

Koçak, Onur

Doktora, Kriptografi

Tez Yöneticisi : Doç. Dr. Ali Doğanaksoy

Ortak Tez Yöneticisi : Yrd. Doç. Dr. Fatih Sulak

Eylül 2016, 82 sayfa

Rastgele sayılar günlük hayatta birçok uygulamada kullanılmaktadır. Örneğin, cep telefonunuz bir baz istasyonuna bağlanacağı zaman, baz istasyonu telefonunuzu doğrulamak için bir rastgele sayı gönderir. E-postalarını kontrol etmek için internet sitesine giriş yaparken bilgisayarınız ve e-posta sunucusunun el sıkışması sırasında aralarında rastgele sayılar gönderirler. Bunu yanı sıra, şifreleme anahtarları ve ilk değerlerin de üçüncü kişilerin tahmin edememeleri için rastgele olması gerekmektedir. Kriptografi, bilgi teorisi, simülasyon ve kuantum teorisi gibi alanlarda benzer örneklerin sayısı artırılabilir.

Rastgele sayı dizileri rastgele sayı üreteçleri(RSÜ) tarafından üretilirler. Belirsizlikleri düşük olan RSÜlerin çıktılarının rastgele dizilerden ayırt edilemez olduğundan emin olmak için test edilmeleri gerekmektedir. Ancak, bu testler teorik olarak gerçekleştirilemezler. Bu sebeple, testler, çıktılar üzerine istatistiksel rastgellik testleri uygulanarak yapılır ve sonuçları değerlendirilir. Sonucun güvenilir olması için birden çok test içeren test paketleri kullanılmaktadır.

Neredeyse bütün test paketleri testlerin dağılım fonksiyonları için yaklaşımlar kullanırlar. Ancak, yaklaşımlar uzun diziler için doğru sonuç verdikleri için, testler de uzun diziler üzerinde çalışmakta ve şifreleme anahtarları ya da ilk değerler gibi kısa dizilerin test edilmesi mümkün olmamaktadır. Bununla birlikte, testler arasındaki

ilişkiler hiçbir pakette ölçülmemiştir.

Bu tezde literatürdeki istatistiksel rastgelelelik testleri incelenmiş, matematiksel bir tabana oturan ve istatistiki olarak anlamlı olan testler seçilmiştir. Seçilen testlerin dağılım fonksiyonları incelendikten sonra yaklaşım değerleri yerine gerçek değerlerden oluşan olasılık değerleri elde edilmiştir. Bu değerlerden uzun diziler için hesaplanması uygulanabilir olmayanlar için tekrarlamalı formüller verilmiştir. Ardından testler arasındaki ilişkiler incelenmiş ve buna göre testlerin sınıflandırılması yapılmıştır. Son olarak test paketleri oluşturulurken takip edilmesi gereken kurallardan bahsedildikten sonra seçilen testlerden bir test paketi oluşturulmuştur.

Anahtar Kelimeler: İstatistiksel Testler, Dağılım Fonksiyonu, Test Paketleri, Korelasyon, Sınıflandırma

To My Family

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor Assoc. Prof. Dr. Ali Doğanaksoy for his invaluable guidance and enthusiastic encouragement throughout this thesis.

I wish to express my sincere appreciation to my co-supervisor Asst. Prof. Dr. Fatih Sulak for many interesting ideas, advices and guidance. This work would not be completed without his help. My gratitude is extended to Dr. Muhiddin Uğuz for his suggestions and contributions.

My sincere thanks go to all my friends and colleagues for their close friendship and motivation. I would like to thank committee members, academic and administrative staff of the Institute of Applied Mathematics.

Very special thanks to my wife Neşe for her endless support, patience and love, and also for being with me all the way.

I am grateful to my dearest family for their unconditional love and supporting me throughout my life.

A final thank goes to my daughter İdil for waiting patiently to be born until my presentation.

The generous financial support of the Scientific and Technological Research Council of Turkey (TUBITAK) Graduate Scholarship no. 2211 is gratefully acknowledged.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

$\mathbb{R}$	Real Numbers
σ	binary sequence
n	length of the binary sequence
$\tilde{\sigma}$	b-bit integer sequence corresponding to σ
b	integer bit size
l	length of the integer sequence, $l = n/b$
$\hat{\sigma}$	± 1 sequence corresponding to σ
$T(\sigma)$	output of the test T on the sequence σ
$\mathbb{T}$	the set of possible test values
λ	the number of bins for the χ^2 test
$\sigma_{i \to j}$	the subsequence of length $j - i + 1$ that starts from s_i and ends with s_j, inclusive.

CHAPTER 1

INTRODUCTION

Random numbers are used in many applications in our daily life. For instance, when your mobile phone is registering a base station, base station sends a random number. Moreover, when logging in your e-mail or bank account your browser and the server exchanges random numbers while authenticating each other. An encryption key should be random so that no one can predict the key without trying all the possible values. The number of applications can be increased from many fields including cryptography, information theory, simulation and quantum theory.

In this work, we deal with random number sequences. These sequences satisfy should the following criteria.

Let σ be a sequence with terms from the set $\mathbb{T}$ such that $|\mathbb{T}| = m$. Then,

- each element should occur with probability $\frac{1}{m}$,
- even if the first t terms of the sequence is known, the $t+1^{st}$ term can be predicted with probability $\frac{1}{m}$, again,
- the elements of the set $\mathbb{T}$ should be distributed uniformly among the sequence.

The random number sequences are generated by the random number generators (RNG)[1]. These generators produce unpredictable data by using one or more sources of randomness. RNGs can be classified in two groups according to their source of randomness. A RNG that requires an input called **seed** and generates output by algorithmically processing this seed is called a **pseudo-random number generator (PRNG)**. PRNGs are deterministic generators and same input parameters yield the same output. In addition to the PRNGs, generators that use unpredictable events as the source of randomness are called **truly random number generators(TRNG)**. A TRNG uses physical data sources like coin flipping, radioactive decay, stratospheric noise etc. TRNGs are non-deterministic generators. As the input source is not in control of the user and it is not possible to get exactly the same parameters with a previous generation, an output sequence cannot be regenerated. The singularity of the output is an important requirement from the security point of view, but, there are some situations where it is adopted

[1] Throughout this thesis, random number generator name will also cover random sequence generators.

to regenerate the sequence. For instance, instead of transmitting a long random sequence over a secure channel, it is more practical and preferable to send just the seed and generate the sequence in the receiving party. Also, TRNGs are very expensive devices in terms of execution and applicability. For these reasons, PRNGs are preferred to TRNGs.

Since PRNGs are deterministic generators, they should be tested to make sure the output sequences are indistinguishable from random sequences. Unfortunately, theoretic testing is not possible if the output sequence has very obvious relations which is not a usual case. Therefore, testing is done statistically using randomness tests and the results are evaluated to conclude the non-randomness of the generator.

A statistical test examines certain characteristics of the sequence, compares the result to those in random sequences and produces a p-value. ***P*-value**, in the context of this thesis, is the probability of obtaining the observed or more extreme results when the sequence is random and it is a real number between 0 and 1. If the p-value is below a pre-determined limit significance level α, the sequence is concluded to be non-random. Otherwise, the sequence cannot be considered random; the results indicate that the sequence possesses the tested property as expected from a random sequence. Moreover, by definition, sequences with a p-value less then α occurs with probability α. Therefore, one expects to observe $t \times \alpha$ such sequences if t sequences are tested. That is, a single sequence with p-value less than α cannot be regarded as a sign of non-randomness of the generator or the sequence set.

As a statistical test checks a single property, in order to decide on the randomness of a sequence, sequence set or a generator, one needs to run multiple tests on the data. For this purpose, test suites that cover a set of tests are composed. There are many statistical test suites in the literature like Knuth[1], NIST[2], Test-U01[3], Diehard[4], Dieharder[5], Crypt-X[6] etc. Among these suites, one of the mostly considered and used was developed by NIST [1]. NIST Test Suite consists of 15 statistical randomness tests and has been used in many applications. For example, in evaluation of AES candidate algorithms, one of the criteria was the performance of the algorithms as pseudo random number generators. NIST test suite was used to determine the number of rounds at which the algorithms behave like a PRNG, to understand the security level of the algorithms roughly[7].

Designing a test suite is an important task and some rule of thumbs should be followed. On the one hand coverage of a suite should be wide, that is it should compare the sequence under consideration in many different points of view with true random sequences. On the other hand, an over populated suite is not far away from being expensive in terms of running time and computing power. Unfortunately, this trade-off is ignored in most of the suites in use. There are approximately 50 distinct statistical randomness tests in the literature and each test suite collects a number of them without any reasoning and further examination.

With the motivation of designing a proper test suite, we examine the statistical randomness tests in the literature. We select the ones that are based on a mathematical background and important measure for randomness among the ones with calculable distribution functions. Then, we review the distribution functions of these tests and

present the distribution functions with actual probability values instead of approximations. When the actual values are not feasible to compute we present recursions which are computationally feasible, at least, up to 4096 bits. We redefine some tests in order to improve the sensitivity. Besides, we present new statistical randomness tests. We define correlation detection methods and compute the correlations between selected tests. According to the correlations, we classify these tests. We give the essential steps in designing a test suite and build a test suite following these steps.

First we give preliminary information below. In Section 2, we give short and detailed descriptions of the selected tests. Then, in Section 3, we study the correlations and classifications of these tests. In Section 4, we give the important steps in test suite design and construct a test suite using the tests mentioned in this thesis.

1.1 Preliminaries

1.1.1 Test Steps

By "to choose a binary sequence of a given length n randomly", we understand "to choose a binary sequence among all 2^n such sequences with probability $\frac{1}{2^n}$" or equivalently, "to determine each term of the sequence to be either 0 or 1 with equal probabilities, being independent of the other terms of the string". We call a collection which consists of randomly chosen strings a random collection. We discuss how to decide whether a given collection of binary sequences is random or not.

The number of sequences for which the random variable assumes a particular value can be computed theoretically (to obtain the expected value), and also can be counted actually (to obtain the observed value). To decide whether a given collection is random or not, we develop a strategy which is based on the comparison of the expected and observed values.

Let Ω_n be the set of all binary strings $\sigma = s_1 s_2 \cdots s_n$ of length n and let Σ be a collection which consists of N random sequences $\sigma_1, \ldots, \sigma_N$. A statistical test which decides whether the collection Σ is random or not, is constructed by the following six steps.

PHASE 1 - SET-UP

Step 1. Define a test function.

- For the experiment of choosing a string $\sigma \in \Sigma$ randomly, define a random variable $T : \Omega_n \to \mathbb{T}$ where $\mathbb{T}$ is a finite subset of integers.

Step 2. Find the probability distribution function.

- For each fixed n, compute the probability distribution function F_n

$$F_n : \mathbb{T} \to [0, 1]$$

$$F_n(k) = Prob(T \leq k).$$

- In certain cases it may turn out to be infeasible to evaluate the probability distribution function directly at a pair of given integers n and k. In such a case, recursive relations are much practical than computing the explicit expression.

Step 3. Partition the set $\mathbb{T}$.

- Let $T_{\min}$ and $T_{\max}$ be the minimum and maximum values, respectively, that the random variable T can assume.
- Fix an integer$\lambda \geq 2$, put $T_0 = T_{\min}$ and $T_\lambda = T_{\max}$ and choose $T_1, T_2, \ldots, T_{\lambda-1} \in T$ such that $T_0 \leq T_1 < \cdots < T_{\lambda-1} < T_\lambda$.
- Let $B_1 = \{T \in \mathbb{T} | T_0 \leq T \leq T_1\}$ and for $i = 2, \ldots, \lambda$ let $B_i = \{T \in \mathbb{T} | T_i < T \leq T_i\}$. Thus we obtain a partition $B_1, B_2, \ldots, B_\lambda$ of $\mathbb{T}$. We refer to each part of this partition as a *bin*. The minimum and maximum t-values in a bin will be called respectively, the lower and upper bounds of that bin.

Step 4. Compute the expected number of sequences in each bin.

- For each bin B_i, $i = 1, \ldots, \lambda$ compute the probability W_i that the t-value of a random sequence is contained in that bin:

$$\begin{aligned} W_1 &= F(T_1) \\ W_2 &= F(T_2) - F(T_1) \\ &\vdots \\ W_\lambda &= F(T_\lambda) - F(T_{\lambda-1}) \end{aligned}$$

- Compute the expected number E_i of sequences $\sigma \in \Sigma$ such that $T(\sigma) \in B_i$, that is,

$$E_i = N \cdot W_i$$

for $i = 1, \ldots, \lambda$.

- For the χ^2 test (applied at the last step) to be sensitive, minimum of these expected values should be at least 5.

PHASE 2 - APPLICATION
Step 5. Compute the observed values.

- For $i = 1, \ldots, \lambda$ compute O_i, the number of strings $\sigma \in \Sigma$ for which $T(\sigma) \in B_i$, that is

$$O_i = |\{\sigma \in \Sigma | T(\sigma) \in B_i\}|.$$

Step 6. Compute the p-value.

- The random variable χ^2_T defined on Ω_N by setting

$$\chi^2_T = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

  is a χ^2 distributed variable with degree of freedom being $\lambda - 1$.

- We define the p-value of the test to be

$$P_T = \chi^2(\chi^2_T, \lambda - 1).$$

- It may be the case that a test function produces more than one p-value, say $P_{T_1}, \ldots, P_{T_u}$. In such a case we take

$$P_T = 1 - (1 - \min(P_{T_1}, \ldots, P_{T_u}))^{1/u}$$

  as the p-value of the test.

First two steps constitute the theoretical background of the test where we compute the necessary quantities and probability distribution functions. In the third and fourth steps necessary preparations for the χ^2 test are considered. Last two steps describe how the test is to be applied for a given collection.

In the remaining part of this section we will define a number of test functions. For each test function the probability distribution function will be given explicitly together with the theoretical background (Steps 1 and 2). Whenever it is necessary, we will provide recursive relations by which the required figures can be computed efficiently.

The number and bounds of bins can be chosen in many different ways (Steps 3 and 4). For the sequences of length 64 to 4096 we will suggest a reasonable choice for the number of bins and other parameter values if the test uses any. We will also provide bounds for bins corresponding to $n = 128$ bits.

1.1.2 Conversion from Binary to Integer

Some tests given in this thesis requires integer valued sequences. However, if the sequence is binary, one can convert this sequence as follows. Let σ be a binary and b be the required integer size for the test. Then, the converted sequence $\tilde{\sigma}$ will be $\tilde{\sigma} = u_1, \ldots, u_l$ where u_i is the integer whose binary representation is the subsequence

$s_{(i-1)b+1}s_{(i-1)b+2}\cdots s_{ib}$, ie $u_i = \sum_{j=1}^{b} 2^{b-j}s_{(i-1)b+j}$ with $l = \lfloor n/b \rfloor$, $M = 2^b$ and $u_i \in \{0, 1, \ldots M-1\}$, $i = 1, \ldots, l$. For example if $\sigma = 0011011001110$ then $\tilde{\sigma} = (001)_2(101)_2(100)_2(111)_2 = 1, 5, 4, 7$ is the 3-bit representation of σ.

In some cases an *overlapping* conversion is required. In this case, the binary representation of u_i will be $s_i s_{i+1} \cdots s_{i+b-1}$.

If the tested sequence is an integer sequence with a distinct integer bit size, then one can convert this sequence to binary first, and then to the required bit-length integer sequence using the above mentioned conversion.

1.1.3 Conversion from Binary to ± 1Sequence

Random excursion tests operates on the ± 1 correspondence of the binary sequences. For this purpose, the sequences should be converted to ± 1 equivalents. Let σ be a binary sequence st $\sigma = s_1 s_2 \ldots s_n$. Then, $\hat{\sigma}$, the corresponding ± 1 sequence is $\hat{\sigma} = (-1)^{s_1}(-1)^{s_2}\ldots(-1)^{s_n}$.

For example, if $\sigma = 0011011001110$ then $\hat{\sigma} = (-1)^0(-1)^0(-1)^1(-1)^1(-1)^0(-1)^1$ $(-1)^1\,(-1)^0\,(-1)^0(-1)^1\,(-1)^1(-1)^1(-1)^0 = 1, 1, -1, -1, 1, -1, -1, 1, 1, -1, -1, -1, 1$ is the new representation of σ.

1.1.4 Test Setup

In the experiments conducted for this thesis, we follow the below steps:

1. Fix a 128-bit key K and generate 10.000×1000 sequences by $\sigma_i = AES_K(P_i)$ where $(P_i)_{10} = i$.
2. For each set of 1000 sequences apply the tests and get tests value from each test.
3. Using the test values of 1000 sequences compute the *p*-value.
4. Repeat Step 3 10.000 times and produce 10.000 *p*-values for each test.

CHAPTER 2

STATISTICAL TESTS REVISITED

For this work, we examine the statistical randomness tests in the literature. We select the tests that are based on a mathematical background and important measure for randomness among the ones with calculable distribution functions. These tests are frequency test, run test, longest run test, linear complexity and linear complexity profile test, random excursion test, random height test, template matching test, integer frequency test, integer maximum test, saturation test(coupon collector), repetition test and universal test. Then, we review the distribution functions of these tests and present the distribution functions with actual probability values instead of approximations. When the actual values are not feasible to compute we present recursions which are computationally feasible, at least, up to 4096 bits.

We redefine some tests in order to improve the sensitivity and present new statistical randomness tests. For instance, the run test, which checks the number of total runs in a sequence appears in many test suites. However, besides checking the total runs, following the Golomb's approach [8], it is more convenient to check the number of runs of length 1, 2, 3 and so seperately. Therefore, we calculate the probabilities for the number of runs of length r with $r = 1, 2, 3, 4, 5$ and $r > 5$, and define as a test named the number of runs of length r.

Another test derived from the Golomb's postulates is the autocorrelation test. We define autocorrelation function and provide the details in this section.

Linear complexity is an important measure for randomness. Therefore, we include linear complexity and linear complexity profile tests in our test set. Besides, we present two new tests, pseudo-complexity and pseudo-complexity profile, which have the same distribution functions with linear complexity counterparts and simpler to apply.

We review the random excursion and random excursion height tests given in the NIST Test Suite [2] and present the actual probability values using a new approach. Similarly, as the distribution function of overlapping template matching test in the NIST Test Suite is known to be faulty [9][10], we use the probabilities for 4-bit templates from [11].

For the remaining tests the actual probability values are calculated and recursions for efficient computation of the probabilities are given, if necessary.

In this chapter we give the definitions of selected tests in two formats. The first form gives just the necessary information about the tests, what they measure, gives suggested parameters and probability values for 128 bits. Also, a recursion is given if the computation of the probability is infeasible for values up to 4096. The second form is the detailed descriptions of the tests given in the first section. The detailed descriptions present how the distribution functions, and if exist recursions, are calculated.

2.1 Short Definitions

In this section, brief definitions of selected statistical randomness tests are given in an easy to follow format. The aim of this section is to give necessary information to those who needs the only the information to apply the tests. The details of the test, derivation of test functions and useful recursions are given in Section 2.2.

2.1.1 Weight Test

Definition

$T = T(\sigma)$ is the number of 1's in the sequence:

$$T = \sum_{i=1}^{n} s_i,$$

$\mathbb{T} = \{0, 1, \ldots, 2^n\},\ T_{min} = 0\ , T_{max} = 2^n.$

For example, if $\sigma = 0011011001110$ then the weight of the sequence is 7.

Probability distribution function

$$F_n(k) = 2^{-n} \sum_{i=0}^{k} \binom{n}{i} \tag{2.1}$$

Useful recursions

Initial values:

$$F_1(0) = 1,\ F_1(1) = \frac{1}{2}.$$

For $n \geq 2$ and $k = 0$

$$F_n(0) = \frac{1}{2} F_{n-1}(0).$$

For $n \geq 2$ and $k \geq 1$

$$F_n(k) \quad = \quad \frac{1}{2}[F_{n-1}(k) + F_{n-1}(k-1)].$$

we naturally assume that $F_n(k) = 1$ whenever $k \geq n$.

Suggested parameter values

For any $64 \leq n \leq 4096$ we can use $\lambda = 8$ bins.

Bounds and corresponding probabilities of bins

The bin values for $n = 64, 128$ are given in Tables 2.1 and 2.2 respectively.

Table 2.1: Weight test bin probabilities for $n = 64$

i	1	2	3	4	5	6	7	8
T_i	25	27	28	29	31	32	34	64
E_i	0,09966	0,12153	0,08339	0,09489	0,20224	0,09462	0,15123	0,15244

Table 2.2: Weight test bin probabilities for $n = 128$

i	1	2	3	4	5	6	7	8
T_i	55	58	60	61	63	65	68	128
E_i	0,10505	0,13217	0,12326	0,06879	0,14195	0,13319	0,15412	0,14147

2.1.2 Number of Total Runs Test

Definition

A run is defined as an uninterrupted sequence of identical bits[2]. For example, in the sequence $\sigma = 0011011001110$, there are 7 runs: $00 - 11 - 0 - 11 - 00 - 111 - 0$.

$T = T(\sigma)$ is the number of runs in the sequence.

$\mathbb{T} = \{0, 1, \ldots, n-1\}$, $T_{min} = 0$, $T_{max} = n - 1$.

Probability distribution function

The probability distribution function of number of runs is

$$F_n(k) = 2^{-n+1} \sum_{i=0}^{k} \binom{n-1}{i-1}.$$

Useful recursions

Initial values:

$$F_1(0) = 0,\ F_1(1) = 1.$$

For $n \geq 2$ and $k = 1$

$$F_n(0) = \frac{1}{2} F_{n-1}(0).$$

For $n \geq 2$ and $k \geq 2$

$$F_n(k) \quad = \quad \frac{1}{2}[F_{n-1}(k) + F_{n-1}(k-1)].$$

we naturally assume that $F_n(k) = 1$ whenever $k \geq n$.

Suggested parameter values

For any $64 \leq n \leq 4096$ we can use $\lambda = 8$ bins.

Bounds and corresponding probabilities of bins

The bin values for $n = 64$ and $n = 128$ are given in Tables 2.3 and 2.4 respectively.

Table 2.3: Number of total runs test bin values for $n = 64$

i	1	2	3	4	5	6	7	8
T_i	25	27	28	29	31	32	34	64
E_i	0,09966	0,12153	0,08339	0,09489	0,20224	0,09462	0,15123	0,15244

Table 2.4: Number of total runs test bin values for $n = 128$

i	1	2	3	4	5	6	7	8
T_i	56	59	61	62	64	66	69	128
E_i	0,10698	0,13197	0,12243	0,13860	0,13860	0,12243	0,13197	0,10698

2.1.3 Runs of Length r Tests

Definition

This test outputs a test value and a $p-$value for each run length $r \in \{1, 2, 3, 4, 5\}$ and $r > 5$.

$T_r = T_r(\sigma)$ is the number of runs of length r in σ:

$$T_r(\sigma) = \text{ number of runs of length } r$$

$\mathbb{T} = \{0, \ldots, \lfloor \frac{n}{r} \rfloor\}$, $T_{\min} = 0$, $T_{\max} = \lfloor \frac{n}{r} \rfloor$.

In this work we take $r = 1, 2, 3, 4, 5$ and $r > 5$. Considering the sample sequence $\sigma = 0011011001110$, there are 2 runs of length 1, 4 runs of length 2 and a run of length 3. There are no runs of length 4 or 5.

To avoid confusions, we name each test as Run-r wwhich measure the number of runs of length r.

Probability distribution function

Let $C_r(n, k)$ be the number of n bit sequences which include k runs of length r. Then;

$$P(T_r = k) = 2^{-n+1}\left(\sum_{i=1}^{n} C_r(n-i, k) - C_r(n-r, k) + C_r(n-r, k-1)\right) \quad (2.2)$$

Useful recursions

$$C_r(n, k) = 2Cr(n-1, k) - C_r(n-r, k) + C_r(n-r-1, k) + C_r(n-r, k-1) - C_r(n-r-1, k-1) \quad (2.3)$$

Suggested parameter values

No parameters are suggested.

Bounds and corresponding probabilities of bins

The bin values and probabilities for $n = 64, 128$ and 4096 are given below.

2.1.4 Longest Run Test

Definition

$T = T(\sigma)$ is the length of the longest run in the sequence.

$\mathbb{T} = \{0, 1, \ldots, n\}$, $T_{min} = 0$, $T_{max} = n$.

Table 2.5: Bin boundaries and corresponding probabilities of Runs of length $k = 1, 2, 3, 4, 5$ and $k > 5$ for $n = 64$.

		Bin-1	Bin-2	Bin-3	Bin-4	Bin-5	Bin-6	Bin-7	Bin-8	
Run-1	T_i	10	12	14	15	17	18	21	64	
	E_i	0,084674	0,105409	0,152089	0,086789	0,174560	0,079882	0,181243	0,135355	
Run-2	T_i	4	5	6	7	8	9	10	32	
	E_i	0,078765	0,082579	0,118070	0,142894	0,149094	0,135925	0,109362	0,183310	
Run-3	T_i	1	2	3	4	5	6	7	22	
	E_i	0,070597	0,137228	0,204320	0,216732	0,173893	0,109353	0,055106	0,032772	
Run-4	T_i	0	1	2	3	16				
	E_i	0,117560	0,274326	0,294674	0,192801	0,120639				
Run-5	T_i	1	2	12						
	E_i	0,358654	0,389813	0,251532						
Run-5+	T_i	26	29	31	33	35	37	64		
	E_i	0,100698	0,171629	0,169352	0,184659	0,162041	0,112863	0,098758		

Table 2.6: Bin boundaries and corresponding probabilities of Runs of length $k = 1, 2, 3, 4, 5$ and $k > 5$ for $n = 128$.

		Bin-1	Bin-2	Bin-3	Bin-4	Bin-5	Bin-6	Bin-7	Bin-8	
Run-1	T_i	24	27	29	31	33	36	39	128	
	E_i	0,099982	0,119213	0,108184	0,122274	0,124762	0,167098	0,122133	0,136354	
Run-2	T_i	11	13	14	15	16	18	19	64	
	E_i	0,105171	0,141532	0,094448	0,103772	0,106022	0,191194	0,075396	0,182464	
Run-3	T_i	4	5	6	7	8	9	10	43	
	E_i	0,077486	0,085723	0,124208	0,150293	0,154855	0,137851	0,107208	0,162376	
Run-4	T_i	1	2	3	4	5	6	7	32	
	E_i	0,076203	0,141979	0,205089	0,212690	0,168575	0,106134	0,054467	0,034863	
Run-5	T_i	0	1	2	3	25				
	E_i	0,124340	0,274874	0,286809	0,187762	0,126214				
Run-5+	T_i	56	59	61	63	65	67	70	128	
	E_i	0,125678	0,126039	0,112692	0,127158	0,129267	0,118009	0,134873	0,126283	

Probability distribution function

Let $C_r(n, k)$ be the number of n bit sequences which include k runs of length r. Then;

$$P(T = t) = 2^{-n+1}C_t(n, 0) \tag{2.4}$$

Useful recursions

No recursions are required.

Suggested parameter values

No parameters are required.

Bounds and corresponding probabilities of bins

The bin boundries and probabilities are given in the following tables.

Table 2.7: Bin boundaries and corresponding probabilities of Longest Run test for $n = 64$ bits.

i	1	2	3	4	5	6
T_i	4	5	6	7	8	64
E_i	0,253773	0,256119	0,174412	0,099541	0,105145	0,110646

Table 2.8: Bin boundaries and corresponding probabilities of Longest Run test for $n = 128$ bits.

i	1	2	3	4	5	6
T_i	5	6	7	8	9	128
E_i	0,121542	0,244832	0,248288	0,173365	0,101326	0,110646

2.1.5 Linear Complexity Test

Definition

$T = T(\sigma)$ is the length of the shortest LFSR that generates the sequence σ. The linear complexity is calculated via Berlekamp-Massey Algorithm [12] [13].

$$T = \mathcal{L}(\sigma).$$

$\mathbb{T} = \{0, \ldots, n\}$, $T_{\min} = 0$, $T_{\max} = n$.

Probability distribution function

$$Prob(T = l) = \begin{cases} 2^{min(2n-2L,2L-1)-n} & \text{if } n \geq L > 0 \\ 2^{-n} & \text{if } L = 0 \end{cases}$$

Useful recursions

Let $N_n(L)$ be the number of sequences of length n with linear complexity L. Then,

$$N_n(L) = \begin{cases} 2N_{n-1}(L) + N_{n-1}(n-L) & n \geq L > \frac{n}{2} \\ 2N_{n-1}(L) & L = \frac{n}{2} \\ N_{n-1}(L) & \frac{n}{2} > L \leq 0 \end{cases}$$

Suggested parameter values

It is suitable to use $\lambda = 6$ bins.

Bounds and corresponding probabilities of bins

The bin bounds and corresponding probabilities are given in Table 2.10 and Table 2.9.

Table 2.9: Linear complexity test bin probabilities for $n = 64$

i	1	2	3	4	5	6
T_i	30	31	32	33	34	Rest
E_i	0.03125	0.125	0.5	0.25	0.0625	0.03125

Table 2.10: Linear complexity test bin probabilities for $n = 128$

i	1	2	3	4	5	6
T_i	62	63	64	65	66	Rest
E_i	0.03125	0.125	0.5	0.25	0.0625	0.03125

2.1.6 Linear Complexity Profile Test

Definition

Let $\sigma_{i \to j}$ be the subsequence of length $j - i + 1$ that starts from s_i and ends with s_j, inclusive. Then, $T = T(\sigma)$ is the number of distinct terms in the linear complexity profile sequence.

$$T(\sigma) = |\{LP_i | LP_i = LC(\sigma_{1 \to i})\}|.$$

$\mathbb{T} = \{0, \ldots, \frac{n}{2}\}$, $T_{\min} = 0$, $T_{\max} = \frac{n}{2}$.

Probability distribution function

For even n,

$$Prob(T = p) = \begin{cases} 3 \cdot 2^{-n} & \text{if } p = 1 \\ (3 \sum_{i=2}^{n/2} 2^i + 1) 2^{-n} & \text{if } p = 2 \\ (3 \sum_{i=2}^{n/2} 2^i \binom{i-3}{p-2}) 2^{-n} & \text{if } p > 2 \end{cases}$$

Useful recursions

No recursions are needed.

Suggested parameter values

In this test, we suggest to use $\lambda = 10$ bins.

Bounds and corresponding probabilities of bins

Table 2.11: Linear complexity profile test bin probabilities for $n = 128$

i	1	2	3	4	5	6	7	8	9	10
T_i	25-27	28-29	30	31	32	33	34	35	36-37	Rest
E_i	0.0878467	0.1263326	0.0844256	0.0946567	0.0996953	0.0986497	0.0917094	0.0800892	0.1162605	0.1203343

2.1.7 Pseudo Complexity Test

Definition

$T = T(\sigma)$ can be defined as the "1"s complexity of the sequence. The pseudo-complexity is calculated similar to the linear complexity where the i^{th} next discrepancy is equal to 1 if $s_i = 1$, zero otherwise.

$$\Psi_{i+1} = \begin{cases} \Psi_i & \text{if } \Psi_i > \frac{n}{2} \\ \Psi_i & \text{if } \Psi_i \leq \frac{n}{2} \text{ and } s_i = 0 \\ \Psi_i & \text{if } \Psi_i \leq \frac{n}{2} \text{ and } s_i = 1 \end{cases}$$

The distribution function of pseudo-complexity test is same with the linear complexity, however, it is easier to apply this test than linear complexity test.

$$T = \Psi(\sigma).$$

$\mathbb{T} = \{0, \ldots, n\}$, $T_{\min} = 0$, $T_{\max} = n$.

Probability distribution function

$$Prob(T = l) = \begin{cases} 2^{min(2n-2L, 2L-1)-n} & \text{if } n \geq L > 0 \\ 2^{-n} & \text{if } L = 0 \end{cases}$$

Useful recursions

Let $N_n(L)$ be the number of sequences of length n with pseudo complexity L. Then,

$$N_n(L) = \begin{cases} 2N_{n-1}(L) + N_{n-1}(n-L) & n \geq L > \frac{n}{2} \\ 2N_{n-1}(L) & L = \frac{n}{2} \\ N_{n-1}(L) & \frac{n}{2} > L \leq 0 \end{cases}$$

Suggested parameter values

It is suitable to use $\lambda = 6$ bins.

Bounds and corresponding probabilities of bins

The bin bounds and corresponding probabilities are given in Table 2.12 and Table 2.13.

Table 2.12: Pseudo-complexity test bin probabilities for $n = 64$

i	1	2	3	4	5	6
T_i	30	31	32	33	34	Rest
E_i	0.03125	0.125	0.5	0.25	0.0625	0.03125

Table 2.13: Pseudo complexity test bin probabilities for $n = 128$

i	1	2	3	4	5	6
T_i	62	63	64	65	66	Rest
E_i	0.03125	0.125	0.5	0.25	0.0625	0.03125

2.1.8 Pseudo Complexity Profile Test

Definition

Let $\sigma_{i \to j}$ be the subsequence of length $j - i + 1$ that starts from s_i and ends with s_j, inclusive. Then, $T = T(\sigma)$ is the number of distinct terms in the pseudo complexity profile sequence.

$$T(\sigma) = |\{LP_i | LP_i = LC(\sigma_{1 \to i})\}|.$$

$\mathbb{T} = \{0, \ldots, \frac{n}{2}\}$, $T_{\min} = 0$, $T_{\max} = \frac{n}{2}$.

Probability distribution function

For even n,

$$Prob(T = p) = \begin{cases} 3 \cdot 2^{-n} & \text{if } p = 1 \\ (3\sum_{i=2}^{n/2} 2^i + 1)2^{-n} & \text{if } p = 2 \\ (3\sum_{i=2}^{n/2} 2^i \binom{i-3}{p-2})2^{-n} & \text{if } p > 2 \end{cases}$$

Useful recursions

No recursions are needed.

Suggested parameter values

In this test, we suggest to use $\lambda = 10$ bins.

Bounds and corresponding probabilities of bins

Table 2.14: Pseudo complexity profile test bin probabilities for $n = 128$

i	1	2	3	4	5	6	7	8	9	10
T_i	25-27	28-29	30	31	32	33	34	35	36-37	Rest
E_i	0.0878467	0.1263326	0.0844256	0.0946567	0.0996953	0.0986497	0.0917094	0.0800892	0.1162605	0.1203343

2.1.9 Random Excursion Tests

Definition

For random excursion tests, the sequence should be converted to the corresponding ± 1 sequence by following the conversion given in Section 1.1.3. Let the partial sums of the ± 1 sequence $\hat{\sigma}$ sequence be $S_i = \sum_1^i \tilde{s}_i$. Then $T = T(\sigma)$ is the number times the partial sum of the sequence equals c:

$$T(\sigma) = |\{i | S_i = c \text{ where } S_i = \sum^i \tilde{s}_i, i \in \{1, 2, \ldots, n\}\}|,$$

$\mathbb{T} = \{0, \ldots, \frac{n}{2}\}$, $T_{\min} = 0$, $T_{\max} = \frac{n}{2}$.

In this work c is taken to be $0, \pm 1, \pm 2, \pm 3$.

Considering the sample sequence $\sigma = 0011011001110$, the partial sums of corresponding ± 1 sequence will be 1,2,1,0,1,0,-1,0,1,0,-1,-2,-1. The partial sum is equal to 0 four times, 1 three times and so on.

Probability distribution function

The probability distribution is given as a recursion.

Useful recursions

Let $x_t(n, k)$ be the number of strings of length n which have intersect the line $y = t$ exactly at k distinct points. Then

$$x_t(n, k) = x_{t-1}(n + 1, k) - x_{t-2}(n, k).$$

The probability values can be calculated by dividing $x_t(n, k)$ by 2^n.

Suggested parameter values

No parameters are required.

Bounds and corresponding probabilities of bins

The probability values for $n = 128$ and $y = 1, 2, 3$ are given below.

Table 2.15: Bin values and probabilities for $n = 128,\ y = 1$ and $y = -1$

i	1	2	3	4	5	6	7	8
T_i	1	3	5	7	9	12	16	64
E_i	0,140772	0,138555	0,131984	0,121481	0,107849	0,132083	0,119493	0,107782

Table 2.16: Bin values and probabilities for $n = 128,\ y = 2$ and $y = -2$

i	1	2	3	4	5	6	7	8
T_i	0	2	4	6	8	11	15	64
E_i	0,139689	0,137524	0,131103	0,120833	0,107487	0,132098	0,120326	0,110939

2.1.10 Random Excursion Height Test

Definition

Table 2.17: Bin values and probabilities for $n = 128,\ y = 3$ and $y = -3$

i	1	2	3	4	5	6	7	8
T_i	0	2	4	6	9	12	64	64
E_i	0,208993	0,134829	0,126409	0,114484	0,144037	0,107825	0,163423	0,110939

First the sequence σ is converted to the ± 1 sequence using the conversion given in Section 1.1.3

Let $S_i = \sum_1^i \hat{s}_i$. Then $T = T(\sigma)$ is the maximum value of the partial sum of the ± 1 sequence $\hat{\sigma}$:

$$T(\sigma) = max\{S_i\}\ st.\ S_i = \sum^i \hat{s}_i\ i = 1, 2, \ldots, n$$

$\mathbb{T} = \{0, \ldots, n\}$, $T_{\min} = 0$, $T_{\max} = n$.

Considering the sample sequence $\sigma = 0011011001110$, the partial sums of corresponding ± 1 sequence will be 1,2,1,0,1,0,-1,0,1,0,-1,-2,-1. The maximal value of the partial sum sequence is 2.

Probability distribution function

The probability distribution is given as a recursion.

Useful recursions

Let $h_t(n)$ be the number of strings of length n whose excursions intersects $y = t$ line at its highest point. Then

$$h_t(n) = h_{t-1}(n-1) + h_{t+1}(n-1)$$

The probability values can be calculated by dividing $h_t(n)$ by 2^n.

Suggested parameter values

No parameters are required.

Bounds and corresponding probabilities of bins
The probability values for $n = 128$ are given below.

Table 2.18: Random Excursion Height Test bin values and probabilities for $n = 128$

i	1	2	3	4	5	6	7	8
T_i	1	3	5	7	9	12	16	128
E_i	0.13968932	0.13545631	0.12736937	0.11613089	0.10266644	0.12824778	0.11781508	0.13262480

2.1.11 Template Matching Test

Definition

$T = T(\sigma)$ is the frequency of a predefined template $\mathcal{B}$ in σ:

$T(\sigma) = \{(s_i, s_{i+1}, s_{i+2}, s_{i+3})$ st $s_i s_{i+1} s_{i+2} s_{i+3} = \mathcal{B}$ and $i = 1, 2, \ldots, n$ with $s_{n+i} = s_i\}$.

$\mathbb{T} = \{0, 1, \ldots, M\}$, $T_{\min} = 0$, $T_{\max} = M$.

For example, if $\sigma = 0011011001110$ and $\mathcal{B} = 011$ then the number of matches for $\mathcal{B}$ is 3: $0 \mid$ **0 1 1**$||$**0 1 1**$|0|$**0 1 1**$|10$. Note that the counting is done in an overlapping manner. For instance, if the sequence is $\sigma = 101010...$ and $\mathcal{B} = 101$ then the first match is **101**$|010...$ and the next one is $10|$**101**$|0....$

Probability distribution function

The probabilities are given below as recursions.

Useful recursions

Let $T(n, k)$ be the number of binary strings of length n with k overlapping substrings of length 4. Then, the recursions according to the overlapping structure of the templates are given below.

0-overlapping substrings: 0001, 0011, 0111, 1000, 1100, 1110.

$$T(n, 0) = 2T(n-1, 0) - T(n-4, 0)$$

$$T(n, k) = \begin{cases} 0 & n < 4k \\ 1 & n = 4k \\ 2T(n-1, k) - T(n-4, k) + T(n-4, k-1) & n > 4k \end{cases}$$

1-overlapping substrings: 0010, 0100, 1011, 1101, 0110, 1100.

$$T(n,0) = 2T(n-1,0) - T(n-3,0) + T(n-4,0)$$

$$T(n,k) = \begin{cases} 0 & n < 3k+1 \\ 1 & n = 3k+1 \\ 2T(n-1,k) - T(n-3,k) + T(n-4,k) & \\ +T(n-3,k-1) - T(n-4,k-1) & n > 3k+1 \end{cases}$$

2-overlapping substrings: 0101, 1010.

$$T(n,0) = 2T(n-1,0) - T(n-2,0) + 2T(n-3,0) - T(n-4,0)$$

$$T(n,k) = \begin{cases} 0 & n < 2k+2 \\ 1 & n = 2k+2 \\ 2T(n-1,k) - T(n-2,k) + 2T(n-3,k) - T(n-4,k) & \\ +T(n-2,k-1) - 2T(n-3,k-1) + T(n-4,k-1)) & n > 2k+2 \end{cases}$$

3-overlapping substrings: 0000, 1111.

$$T(n,0) = T(n-1,0) + T(n-2,0) + T(n-3,0) + T(n-4,0)$$

$$T(n,k) = \begin{cases} 0 & n < k+3 \\ 1 & n = k+3 \\ T(n-1,k) + T(n-2,k) + T(n-3,k) + T(n-4,k) & \\ +T(n-1,k-1) - T(n-2,k-1) - T(n-3,k-1) & \\ -T(n-4,k-1)) & n > k+3 \end{cases}$$

The probabilities can be computed via dividing $T(n,k)$ by 2^n.

Suggested parameter values

In this test we use templates of 4 bit length.

Bounds and corresponding probabilities of bins

2.1.12 Autocorrelation Test

Definition

Table 2.19: Template matching test bin values for $n = 128$

	i	1	2	3	4	5
0-bit	T_i	6	7	8	9	128
Overlapping	E_i	0,242056	0,170823	0,186299	0,164018	0,236801
1-bit	T_i	5	7	8	10	128
Overlapping	E_i	0,163531	0,274334	0,154664	0,244821	0,162647
2-bit	T_i	5	7	9	11	128
Overlapping	E_i	0,199905	0,2565	0,255663	0,169316	0,118614
3-bit	T_i	4	6	8	10	128
Overlapping	E_i	0,219765	0,187373	0,185726	0,152004	0,255129

Let $\sigma_{i\to j}$ be the subsequence of length $j - i + 1$ that starts from s_i and ends with s_j, inclusive. Then, $T = T(\sigma)$ is the correlation value between $\sigma_{1\to 64}$ $\sigma_{c\to c+64}$.

$$T(\sigma) = weight(\sigma_{1\to 64} \oplus \sigma_{c_i\to c_i+64})$$

where c is the shift value.
$\mathbb{T} = \{0, \ldots, \frac{n}{2}\}$, $T_{\min} = 0$, $T_{\max} = \frac{n}{2}$.
Probability distribution function

The probability distribution function of auto correlation function is equal to the distribution function of weight test for $\frac{n}{2}$, ie.:

$$F_n(k) = 2^{-n} \sum_{i=0}^{k} \binom{n/2}{i} \tag{2.5}$$

Useful recursions

No recursions are required.

Suggested parameter values

It is suggested to find the correlation value for distinct values of c_i on each sequence. In this work we applied autocorrelation test for $c_i = \{$*odd primes less then* $\frac{n}{2}\}$. The number of bins is selected as 9 for a homogeneous distribution among the bins.

Bounds and corresponding probabilities of bins

The bin boundaries and probabilities are given in Table 2.20.

Table 2.20: Bounds for Auto Correlation Test $n = 128$

i	1	2	3	4	5	6	7	8	9	10
T_i	0-27	28-29	30	31	32	33	34	35-36	37-64	255
E_i	0,130218	0,135937	0,087836	0,096336	0,099347	0,096336	0,087836	0,135937	0,130218	0,10135

2.1.13 Integer Tests

These test group is intended to test integer sequences. They check various properties of integer sequences like maximum term, minimum term, the distribution of the integers and so on. Nevertheless, these tests can be applied to both integer and binary sequences. In the integer sequence case, if the integer size is appropriate with the test, then the sequence is taken as is. If the sequence requires a different integer size, then, the sequence is converted first to binary and then to the required integer-length-sequence. The binary sequences are directly converted to the integer sequences as given in Section 1.1.2 In this section, we give the descriptions of integer sequences and 128-bit bin values for each test.

2.1.13.1 Integer Frequency Test

Definition

First each $\sigma \in \Omega_n$ is converted into a b-bit integer sequence and the frequencies of all possible 2^b integers are computed.

This test produces $M = 2^b$ many test values T_a for each $a \in [0, M-1]$.

$T_a = T_a(\sigma)$ is the frequency of the integer a in the sequence $\tilde{\sigma}$:

$$T_a(\sigma) = |\{u_i : u_i = a, i = 1, \ldots, l\}|, \qquad a = 0, 1, \ldots, M-1$$

$\mathbb{T} = \{0, 1, \ldots, M\}$, $T_{\min} = 0$, $T_{\max} = M$.

Let $\sigma = 0011011001110$ then $\tilde{\sigma} = (001)_2(101)_2(100)_2(111)_2 = 1, 5, 4, 7$ is the 3-bit representation of σ. Then, the frequencies of the integers in the sample sequence are 0,1,0,0,1,1,0 and 1 for 0 through 7 respectively.

Probability distribution function

Obviously T_a is a binomial random variable, that is,

$$F_n(k) = M^{-l}\binom{l}{k}(M-1)^{l-k}.$$

Useful recursions

$k \geq 2$

$$F_l(k) = F_l(k-1) + \frac{l+1-k}{k(M-1)}[F_l(k-1) + F_l(k-2)].$$

Suggested parameter values

It is suitable to choose b and λ as given in Table 2.21.

Table 2.21: Integer bit size and number of bins for integer frequency test

Length of the sequence	b	λ
$64 \leq n \leq 83$	3	6
$84 \leq n \leq 119$	3	7
$120 \leq n \leq 263$	3	8
$264 \leq n \leq 799$	4	8
$800 \leq n \leq 1919$	5	8
$1920 \leq n \leq 4096$	6	8

Bounds and corresponding probabilities of bins

The bin bounds and probabilities are given below.

Table 2.22: Integer frequency test bin values and probabilities for $n = 64,\ b = 3$

i	1	2	3	4	5	6
T_i	0	1	2	3	4	21
E_i	0,06056	0,18167	0,25953	0,23482	0,15095	0,11247

Table 2.23: Integer frequency test bin values and probabilities for $n = 128, b = 3$

i	1	2	3	4	5	6	7	8
T_i	2	3	4	5	6	7	8	42
E_i	0,09011	0,12274	0,17096	0,18561	0,16352	0,12013	0,07508	0,07184

2.1.13.2 Integer Maximum Test

Definition

We first convert each $\sigma \in \Omega_n$ into a b-bit integer sequence as mentioned in Section 1.1.2. Then the testidentifies the maximal element in the sequence.

$T = T(\sigma)$ is the greatest integer contained in the sequence $\tilde{\sigma}$:

$$T(\sigma) = \max\{u_1, \ldots, u_l\},$$

$\mathbb{T} = \{0, 1, \ldots, M-1\}$, $T_{\min} = 0$, $T_{\max} = M - 1$.

Let $\sigma = 0011011001110$ then, $\tilde{\sigma} = (001)_2(101)_2(100)_2(111)_2 = 1, 5, 4, 7$ is the 3-bit representation of σ. So, the maximum term in the sample sequence is 7.

Probability distribution function

$$Prob(T \leq k) = \left(\frac{k+1}{M}\right)^l.$$

Useful recursions

No recursions are required.

Suggested parameter values

For any $64 \leq n \leq 4096$ we can use $\lambda = 8$ bins.

The parameter b can be chosen as

$$b = 1 + \lfloor \log_2(n) \rfloor.$$

Bounds and corresponding probabilities of bins

The bin bounds and probabilities are given below for $n = 64, 128$ and 4096.

Table 2.24: Integer maximum test bin values and probabilities for $n = 128,\ b = 8$

i	1	2	3	4	5	6	7	8
T_i	224	234	240	244	248	250	253	255
E_i	0,126789	0,127453	0,126327	0,114674	0,146484	0,08763	0,152707	0,117936

2.1.13.3 Integer Minimum Test

Definition

The test is applied to $\tilde{\sigma}$, the b-bit converted sequence of σ as described in Section 1.1.2.

$T = T(\sigma)$ is the smallest integer contained in the sequence $\tilde{\sigma}$:

$$T(\sigma) = \min\{u_1, \ldots, u_l\},$$

$\mathbb{T} = \{0, 1, \ldots, M-1\}$, $T_{\min} = 0$, $T_{\max} = M - 1$.

Considering the sample sequence $\sigma = 0011011001110$, the corresponding 3-bit integer sequence will be $\tilde{\sigma} = (001)_2(101)_2(100)_2(111)_2 = 1, 5, 4, 7$. The minimum term in $\tilde{\sigma}$ is 1.

Probability distribution function

$$F(k) = Prob(T \leq k) = 1 - \left(\frac{M-k}{M}\right)^l.$$

Useful recursions

No recursions are required.

Suggested parameter values

For any $64 \leq n \leq 4096$ we can use $\lambda = 8$ bins.

The parameter b can be chosen as

$$b = 1 + \lfloor \log_2(n) \rfloor.$$

Bounds and corresponding probabilities of bins

The bin bounds and probabilities are given in the below table for 128.

Table 2.25: Integer minimum test bin values for $n = 128,\ b = 8$

i	1	2	3	4	5	6	7	8
T_i	2	5	7	11	15	21	31	255
E_i	0,117936	0,152707	0,08763	0,146484	0,114674	0,126327	0,127453	0,126789

2.1.13.4 Integer Maximum Minimum Difference Test

Definition

Each sequence $\sigma \in \Omega_n$ is converted into a b-bit integer sequence as in Section 1.1.2.

$T = T(\sigma)$ is the difference of the largest and smallest terms contained in the sequence $\tilde{\sigma}$:

$$T(\sigma) = max\{u_1, \ldots, u_l\} - \min\{u_1, \ldots, u_l\},$$

$\mathbb{T} = \{0, 1, \ldots, M-1\}$, $T_{\min} = 0$, $T_{\max} = M - 1$.

$\sigma = 0011011001110$ then $\tilde{\sigma} = (001)_2(101)_2(100)_2(111)_2 = 1, 5, 4, 7$ is the 3-bit representation of σ. The minimum and maximum elements in the sample sequence are 1 and 7 respectively. Therefore, the maximum-minimum difference is 6.

Probability distribution function

For $k = 0$ we have

$$Prob(T = 0) = l\left(\frac{1}{M}\right)^l.$$

and for $k \geq 1$

$$F_l(k) = (l-k)\left(\frac{k+1}{M}^l\right) + (k+1-l)\left(\frac{k}{M}\right)^l$$

Useful recursions

No recursions are required.

Suggested parameter values

For any $64 \leq n \leq 4096$ we can use $\lambda = 10$ bins. The parameter b can be chosen as

$$b = 1 + \lfloor \log_2(n) \rfloor.$$

Bounds and corresponding probabilities of bins

The bin bounds and probabilities are given in the below tables for $n = 128$.

Table 2.26: Integer maximum difference test bin values and probabilities for $n = 128,\ b = 8$

i	1	2	3	4	5	6	7	8	9	10
T_i	194	208	217	223	228	233	237	242	247	255
E_i	0,078835	0,096564	0,104781	0,093832	0,093925	0,107233	0,09309	0,119482	0,110909	0,10135

2.1.13.5 Integer Coverage Test

Definition

We first convert each $\sigma \in \Omega_n$ into a b-bit integer sequence as mentioned in Section 2.1.13.

$T = T(\sigma)$ is the number of distinct terms contained in the sequence $\tilde{\sigma}$:

$$T(\sigma) = |\{u_1, \ldots, u_l\}|,$$

$\mathbb{T} = \{1, \ldots, M\}$, $T_{\min} = 1$, $T_{\max} = M$.

For example, if $\sigma = 0011011001110$ then $\tilde{\sigma} = (001)_2(101)_2(100)_2(111)_2 = 1, 5, 4, 7$ is the 3-bit representation of σ. There are 4 distinct terms in the sample sequence. Therefore, the coverage of this sequence is 4.

Probability distribution function

Let $P_l(k) = Prob(T = k)$. Let A and B be sets with l and M elements respectively and let $f : A \to B$ be a random mapping. Then $P_l(k) = Prob(|f(A)| = k)$, so

$$P_l(k) = \frac{k!}{M^l} \binom{M}{k} \left\{ {l \atop k} \right\}$$

where $\left\{ {l \atop k} \right\}$ is the Stirling number of the second kind. Consequently

$$F_l(k) = \sum_{i=0}^{k} P_l(k).$$

Useful recursions

We have

$$P_l(1) = \frac{1}{M^{l-1}}.$$

and using the basic recursion for Stirling numbers of the second kind we get

$$P_l(k) \quad = \quad \frac{k}{M} P_{l-1}(k) + \left(1 - \frac{k-1}{M}\right) P_{l-1}(k-1)$$

for $k \geq 2$.

Suggested parameter values

It is suitable to choose b and λ as given in Table 2.27

Table 2.27: Integer Coverage Test suggested values for b and λ

Length of sequence	b	λ
$64 \leq n \leq 192$	5	5
$192 \leq n \leq 245$	6	6
$246 \leq n \leq 1799$	7	7
$1800 \leq n \leq 4096$	8	8

Bounds and corresponding probabilities of bins

The bin bounds and probabilities for integer coverage test are given in the below tables for $n = 128$.

Table 2.28: Integer coverage test bin values for $n = 128,\ b = 5$

i	1	2	3	4	5
T_i	15	16	17	18	256
E_i	0,105771	0,155071	0,227106	0,232804	0,279249

2.1.13.6 Integer Repetition Test

Definition

The sequence σ is converted to an integer sequence $\tilde{\sigma}$ as given in Section 2.1.13. Then, the first index which the corresponding term appeared before is determined.

$T = T(\sigma)$ is the smallest index k st the term $u_k = u_j$ for some $j < k$:

$$T(\sigma) = min_k\{k|u_k = u_j, j = 0, \ldots, k-1\},$$

$\mathbb{T} = \{1, \ldots M+1\}, T_{\min} = 1, T_{\max} = M+1.$

Notice that, by pigeon hole principle, a repetition occurs at most at $(M+1)^{st}$ index. Let $\sigma = 0011011001110$ then $\tilde{\sigma} = (00)_2(11)_2(01)_2(10)_2(01)_2(11)_2 = 0, 3, 1, 2, 1, 3$ is the 2-bit representation of σ. The repetition point in the sample sequence is 5:$u_5 = u_3$.

Probability distribution function

$$Prob(K = k) = M^{-k}\binom{M}{k-1}(k-1)!(k-1).$$

Useful recursions

No recursions are required.

Suggested parameter values

The bit-length b is set to be 6 for $n = 128$.

Bounds and corresponding probabilities of bins

Table 2.29: Integer repetition test bin values for $n = 128,\ b = 6$

i	1	2	3	4	5	6	7	8
T_i	4	6	8	9	11	13	16	21
E_i	0,091087	0,123378	0,151507	0,079253	0,152509	0,131601	0,141654	0,104323

2.1.13.7 Integer Saturation Test

Definition

The sequence σ is converted to integer sequence $\tilde{\sigma}$ using integer size corresponding to the length of σ and the smallest index at which all b-bit integers appeared is selected.

$T = T(\sigma)$ is the length of the shortest subsequence starting from u_1 that covers all M distinct integers:

$$T(\sigma) = min\{i\} \text{ st. } \{\{u_1, \ldots, u_i\} \equiv \{0, 1, \ldots, M-1\}\}.$$

$\mathbb{T} = \{M, M+1, \ldots, l+1\}$, $T_{\min} = M$, $T_{\max} = l+1$.

The output value is $l + 1$ if the saturation point is never achieved.

Let $\sigma = 0011011001110$ then $\tilde{\sigma} = (00)_2(11)_2(01)_2(10)_2(01)_2(11)_2 = 0, 3, 1, 2, 1, 3$ is the 2-bit representation of σ. In the sample sequence, when $m = 2$, all 2-bit integers appear in the first 4 terms of $\tilde{\sigma}$. Therefore, saturation point of the sequence is 4.

Probability distribution function

$$P(SP = k) = M^{-(k-1)} \left\{ \begin{matrix} k-1 \\ M-1 \end{matrix} \right\} (M-1)!.$$

Useful recursions

No recursions are required.

Suggested parameter values

For $n = 128$, it is suggested to set $b = 3$.

Bounds and corresponding probabilities of bins

Bin bounds and probabilities are given in Table 2.30.

Table 2.30: Integer saturation test bin values for $n = 128,\ b = 3$

i	1	2	3	4	5	6	7	8		
T_i	8-13	14-15	16-17	18-20	21-23	24-27	28-33	34-43	247	255
E_i	0,139321	0,108927	0,117315	0,164996	0,134863	0,128835	0,110264	0,095480	0,110909	0,10135

2.1.13.8 Universal Test

Definition

First, the sequence σ is converted to integer sequence with respect to the corresponding integer size. Then, another sequence is generated considering the distance between successive occurrences of the terms and compared to those of random numbers.

This test produces n many t-values. Namely, a t-value T_a for all $a \in \{0, \ldots, M-1\}$.

$T_a = T_a(\sigma)$ is the frequency of a in the distance sequence $\delta(\tilde{\sigma})$ where $\delta(\sigma) = d_1 d_2 .. d_n$ st

$$d_i = \begin{cases} j - i & j \text{ is the smallest index st } j > i \text{ and } u_i = u_j \\ 0 & \text{if } i \text{ is the last index of } u_i \end{cases}$$

$\mathbb{T}_a = \{0, 1, \ldots, n-a\}$ if $a \neq 0$ and $\mathbb{T}_0 = \{1, \ldots, M\}$, $T_{\min} = 0, T_{\max} = max\{n, M\}$.

Probability distribution function

$$Prob(T = \mathbb{T}) = \chi^2(\mu, l-1)$$

where

$$\mu = \sum_{a=0}^{l-1} \frac{(E_a - T_a)^2}{E_a}$$

and $E_a = (l-a)\frac{(M-1)^{a-1}}{M^a}$.

Useful recursions

No recursions are required.

Suggested parameter values

In this test, we suggest $b = 3$ for $n = 128$.

Bounds and corresponding probabilities of bins

Table 2.31: Universal test bin values and probabilities for $n = 128,\ b = 3$

i	1	2	3	4	5	6	7	8
T_i	1	2	3	5	7	10	15	42
E_i"	0,125	0,109375	0,095703	0,157013	0,120213	0,12962	0,128142	0,131267

2.2 Detailed Descriptions

In this section the details of the probability functions and recursions of statistical randomness tests mentioned in the previous section are given. The aim of this chapter is to give all the information used to calculate probability functions and recursions of each test.

2.2.1 Weight Test

This test corresponds to the frequency test that every test suite includes. In this thesis, we calculate the actual values for this test instead of widely used approximations.

Weight test examines the number of 1's in a sequence and outputs a p-value by comparing the weight of the sequence to the expected weight. In an n bit sequence the probability of the sequence to be ω is calculated as follows. Out of n bins for each bit, one chooses ω bins and places 1's in these bins. Then, the remaining $n - \omega$ bins are filled with $n - \omega$ 0's. Therefore, there are

$$\binom{n}{\omega}\binom{n-\omega}{n-\omega} = \binom{n}{\omega} \tag{2.6}$$

sequences which have weight ω. The probability of a random sequence to have weight ω is

$$Pr(W = \omega) = 2^{-n}\binom{n}{\omega}.$$

The recursions given in 2.1.1 is calculated as follows. Initial values:

$$F_1(0) = 1,\ F_1(1) = \frac{1}{2}.$$

For $n \geq 2$ and $k = 0$

$$F_n(0) = \frac{1}{2}F_{n-1}(0).$$

For $n \geq 2$ and $k \geq 1$

$$\begin{aligned} F_n(k) &= 2^{-n}\sum_{i=1}^{k}\left[\binom{n-1}{i} + \binom{n-1}{i-1}\right] + 2^{-n} \\ &= 2^{-n}\sum_{i=0}^{k}\binom{n-1}{i} + 2^{-n}\sum_{i=0}^{k-1}\binom{n-1}{i} \\ &= \frac{1}{2}[F_{n-1}(k) + F_{n-1}(k-1)]. \end{aligned}$$

we naturally assume that $F_n(k) = 1$ whenever $k \geq n$.

2.2.2 Number of Total Runs Test

This test corresponds to the runs test in the NIST Test suite[2]. In this thesis, we calculate the actual probability values.

The total runs test compares the number of runs within a sequence to the expected number of runs and derives a p-value. The distribution function of the test is very similar to the distribution function of the frequency test.

Let ψ be the differential sequence of σ such that

$$\psi_i = \begin{cases} \sigma_i \oplus \sigma_{i+1} & \text{if } i < n \\ 1 & \text{if } i = n \end{cases}$$

Notice that ψ_i is 1 when $\sigma_i \neq \sigma_{i+1}$ which indicates that a run terminates and another run starts. When the last term, 1, is associated with the last run in the sequence, the weight of ψ is equal to the number of runs in σ plus 1. The result of the weight test over ψ will be the result of the number of total runs test.

In this case when calculating the probabilities one should consider that the last term is always 1. Assume that the weight of ψ is r. Out of n bits, one of the 1's are placed to the last bin, and the remaining $r-1$ 1's are placed in $r-1$ bins among $n-1$ bins. Therefore, the probability of the weight of ψ to be r is

$$Pr(W = r) = 2^{-n}\binom{n-1}{r-1}$$

which is equal to the probability of σ to have r runs.

The recursions for number of total runs test can be derived from Weight test by replacing k with $k+1$.
Initial values:

$$F_1(0) = 0,\ F_1(1) = 1.$$

For $n \geq 2$ and $k = 1$

$$F_n(0) = \frac{1}{2}F_{n-1}(0).$$

For $n \geq 2$ and $k \geq 2$

$$\begin{aligned} F_n(k) &= 2^{-n}\sum_{i=1}^{k}\left[\binom{n-1}{i} + \binom{n-1}{i-1}\right] + 2^{-n} \\ &= 2^{-n}\sum_{i=0}^{k}\binom{n-1}{i} + 2^{-n}\sum_{i=0}^{k-1}\binom{n-1}{i} \\ &= \frac{1}{2}[F_{n-1}(k) + F_{n-1}(k-1)]. \end{aligned}$$

2.2.3 Runs of Length r Tests

The test suites in the literature deals with the number of all runs instead of comparing the the number of runs of length k which gives a more refine idea about the randomness of the sequence. Although, there are some studies which corresponds to the number of runs of length 1 and 2, they are not intended for testing randomness and there is no work on the number of longer runs.

Runs of Length r tests are a set of tests that evaluate the number of runs of length k in a sequence for some k and compare the results to the expected number of runs of length k a random sequence. In this work we suggest $k = 0, \pm 1, \pm 2$ and ± 3.

The number of runs of length k can be calculated in a similar way with the number of total runs test. However, the computations become infeasible and one needs to use approximations to apply the test. Therefore, we follow another method to compute the actual values.

Define the run sequence R as the sequence whose i^{th} term is the length of the i^{th} run in the original sequence. It is obvious that the sum of the terms of R will give the length of the sequence, n. For instance consider $\sigma = 0011011001110$. There are 7 runs which are of length 2, 2, 1, 2, 2, 3 and 1 respectively: $00 - 11 - 0 - 11 - 00 - 111 - 0$. The sum of all lengths adds up 13 which is n. Therefore, run sequence R of an n bit sequence is a composition of n. Any binary sequence can be identified uniquely from the corresponding run sequence up to its first bit being 0 or 1. As the compositions cannot distinguish the first bit, there are 2^{n-1} distinct compositions of n.

Let $Q_r(n,k)$ be the set of all compositions of n with k occurrences of positive integer r. This composition is corresponding to a sequence which includes exactly k runs of

length r. Also, let $C_r(n,k)$ be the size of $Q_r(n,k)$. That is, $C_r(n,k)$ is the number of n bit sequences with exactly k runs of length r.

For convenience and in order to give an idea about how the following recursions are derived, we will begin with $C(n)$, the number of all compositions of n. Although, this number is derived above as 2^{n-1} by correlating with the number of n bit sequences, the following will give the necessary viewpoint for the next computations.

A composition of n can be constructed by appending a '1' to the end of all the compositions of $n-1$, or adding a '2' to all the compositions of $n-2$ and so on. For instance consider the compositions of 4 given in Table 2.32.

Table 2.32: Compositions of 4 obtained from smaller integers

3		2		1		0	
3	1	2	2	1	3	$\emptyset$	4
2,1	1	1,1	2				
1,2	1						
1,1,1	1						

Notice that, 4 is a composition of itself. We assume that this composition is derived by appending 4 to the compositions of zero. For completeness, we assume $C(0) = 1$ and let the only composition of 0 be shown by $\emptyset$.

Therefore, the number of compositions of n is

$$C(n) = \sum_{i=0}^{n-1} C(i).$$

This recursion can be simplified for $n > 0$ as follows:

$$\begin{aligned} C(n) &= C(n-1) + C(n-2) + C(n-3) + \cdots \\ C(n-1) &= \qquad\quad C(n-2) + C(n-3) + \cdots \end{aligned}$$

Subtracting the equations we get

$$\begin{aligned} C(n) - C(n-1) &= C(n-1) \\ C(n) &= 2C(n-1). \end{aligned} \tag{2.7}$$

The initial value for this recursion is $C(1) = 1$. So, $C(n) = 2^{n-1}$.

Moving a step forward, now let's consider $C_r(n,k)$.Similar to the previous computation, a composition of n including k runs of length r can be obtained from appending 1 to the end of the sequences in $Q_r(n-1,k)$, appending 2 to the end of the sequences in $Q_r(n-2,k)$ and so on. However, one should consider the case appending r to the end of the sequences in $Q_r(n-r,k)$. In this case the result will be an element of $Q_r(n,k+1)$. Therefore, this case should be excluded from the computation. Finally,

appending r to the end of sequences in $Q_r(n-r,k-1)$ brings sequences in $Q_r(n,k)$. Therefore, the total number of compositions of n with k occurrences of r is

$$C_r(n,k) = \sum_{i=1}^{n} C_r(n-i,k) - C_r(n-r,k) + C_r(n-r,k-1). \tag{2.8}$$

To compute Equation 2.8 for any r, one needs the to know the value of $C_r(n-i,k)$ for all $i < n$. As n gets larger, this computation will be time consuming. In order to have a "computing friendly" calculations, we need to simplify Equation 2.8. If we subtract $C_r(n-1,k)$ from $C_r(n,k)$, we get:

$$\begin{aligned}
C_r(n,k) &= C_r(n-1,k-1) + C_r(n-2,k) + C_r(n-3,k) + \cdots - C_r(n-r,k) + C_r(n-r,k-1) \\
C_r(n-1,k) &= C_r(n-2,k-1) + C_r(n-3,k) + \cdots - C_r(n-1-r,k) + C_r(n-1-r,k-1) \\
\hline
C_r(n,k) &= 2C_r(n-1,k) - C_r(n-r,k) + C_r(n-1-r,k) + C_r(n-r,k-1) - C_r(n-1-r,k-1)
\end{aligned}$$

Therefore, the probability of an n bit sequence to have k runs of length r is

$$P(R=r) = 2^{-n+1}\big(2C_r(n-1,k) - C_r(n-r,k) + C_r(n-1-r,k) + C_r(n-r,k-1) - C_r(n-1-r,k-1)\big). \tag{2.9}$$

Notice that the number of compositions is divided by 2^{n-1} instead of 2^n since a run sequence defines two binary sequences where one starts with 0 bit, and the other starts with 1 bit.

2.2.4 Longest Run Tests

This test is equivalent to the Longest Run of Ones test in the NIST Test Suite[2]. This test, unlike the NIST counterpart, accepts the longest run from both 0 and 1. Similar to the other tests given in this section, the actual values for the longest run are presented.

The longest run test compares the maximal run length to the expected maximal length in a random sequence. The distribution function of longest run test is derived by using the compositions concept mentioned in the Runs of Length r test.

Let $Q_r(n,k)$ be the set of all sequences of length n, including k runs of length r, and let the size of $Q_r(n,k)$ be $C_r(n,k)$.

In order for an n bit sequence to have the maximal run length t, the number of runs longer than t should be zero. Tthis sequence is an element of $\bigcup_{i>t} Q_i(n,0)$. However, this set also includes sequences that the maximal length of the runs are smaller than t. Hence, if we exclude the sequences with no runs longer than $t-1$, ie runs with length smaller than t, we left with the sequences having maximum run length exactly t.

The number of sequences having the maximal run length as t are in

$$\bigcup_{i>t-1}^{n} Q_i(n,0) - \bigcup_{i>t}^{n} Q_i(n,0)$$

which is obviously $Q_t(n,0)$. The number of sequences in $Q_t(n,0)$ is $C_t(n,0)$, therefore, the probability of such a sequence is

$$P(T=t) = 2^{-n+1}C_t(n,0). \tag{2.10}$$

2.2.5 Linear Complexity Test

Linear complexity test is inherited from NIST Test Suite[2]. It is an important measure for the randomness and has been studied extensively for decades. The following details can be found in [14] and [15].

Linear complexity test compares the linear complexity of the sequence to that of a random sequence.

Let $N_n(L)$ denote the number of n-bit sequences with linear complexity L and let the linear complexity value at the index $n-1$ be L'.

- If $L' < \frac{n}{2}$ and $\delta_{n-1} = 0$ then, by Berlekamb-Massey algorithm $L = L'$. So, $N_n(L) = N_{n-1}(L')$ for this case.
- If $\delta_{n-1} = 1$, then the linear complexity will increase, $L = n - L'$, and $L > \frac{n}{2}$.
- If, on the other hand, $L' > \frac{n}{2}$, no matter what the n^{th} bit is, the complexity will not change: $L = L'$. That is the linear complexity will not change if the last bit is 1 or 0.

Therefore, if $n \geq L > \frac{n}{2}$, $N_n(L) = N_{n-1}(n-L) + 2N_{n-1}(L)$ and if $0 \leq L' < \frac{n}{2}$, then $N_n(L) = N_{n-1}(L)$.

The only remaining part is $L = \frac{n}{2}$. If, any $L' < \frac{n}{2}$ lead to $L = \frac{n}{2}$ by an increase in the linear complexity then $L = \frac{n}{2} = n - L'$ ie. $L' = \frac{n}{2}$ which is not the case. Therefore, the only contribution for the case $L = \frac{n}{2}$ is from $L' > \frac{n}{2}$. In this case, the linear complexity will remain the same regardless of δ_{n-1}. Therefore $N_n(L) = 2N_n(L)$ if $L = \frac{n}{2}$. This yields the recursion

$$N_n(L) = \begin{cases} 2N_{n-1}(L) + N_{n-1}(n-L) & n \geq L > \frac{n}{2} \\ 2N_{n-1}(L) & L = \frac{n}{2} \\ N_{n-1}(L) & \frac{n}{2} > L \leq 0. \end{cases}$$

It is easy to show that the given distribution function satisfies the above recursion for the initial values $N_1(0) = 1$ and $N_1(1) = 1$.

2.2.6 Linear Complexity Profile Test

Linear complexity profile test is not a test that is widely included in test suites. However, this test completes the linear complexity test and is a distinguisher. Linear com-

plexity profile test examines if the jumps in the linear complexity values are as expected from a random sequence.

The probability distribution function for the linear complexity profile test is taken from [16] and [19].

From [19] we know that

$$Pr(L = n, LCP = i) = \begin{cases} 2^l 2^{-n} & \text{if } i = 1, l \leq 1 \\ 2^{t+1}\binom{l-2}{i-2}2^{-n} & \text{if } i > 1, l \leq \frac{n}{2} \\ 2^{n+1-l}\binom{n-l-1}{i-2}2^{-n} & \text{if } i > 1, \frac{n}{2} < l \leq n-1 \\ 2^{-n} & \text{if } i = 2, l = n \\ 0 & \text{otherwise} \end{cases}$$

It is trivial to see the probability function given in 2.1.6 hold using the above equation.

2.2.7 Pseudo Complexity Test

Pseudo complexity is a new randomness test that has the same probability distribution with linear complexity test. However, it is easier to compute pseudo complexity then the linear complexity. The test can be interpret as the "ones complexity" as the complexity changes for suitable indexes if the next term is 1.

The derivation of the distribution function and the recursion is the same as liner complexity test given in 2.2.5.

2.2.8 Pseudo Complexity Profile Test

Pseudo complexity profile is a new randomness test that has the same probability distribution with linear complexity profile test. As in the pseudo complexity test case, it is trivial to compute pseudo complexity profile value, therefore, it is faster to apply.

The derivation of the distribution function and the recursion is the same as liner complexity profile test given in 2.1.6, the details of the distribution function can be found in [16].

2.2.9 Random Excursion Tests

Random Excursion tests occur in the NIST Test Suite in the name of Random Excursion and Random Excursion Variant Tests[2]. However, the tests use approximations in the probability calculations and can be applied to sequences of length longer than 10^6*. In this work, we revisit the distribution function of the test and give the probabilities for sequences of size less then, at least, 4096.*

The number of times the partial sum is equal to c is equal to the number of times the graph of the sequence intersects the line $y = c$.We will follow this approach in this section. Let σbe a binary sequence and define the s_k as a balance point if in the subsequence $\sigma_{1\to k}$ the number of 1s and 0s are equal. A sequence is balanced, obviously, if and only if the last term s_n is a balance point.

Let $B(n, k)$ be the set of balanced sequences which contain exactly k balance points and let $b(n, k)$ be the number of such sequences. By definition, if n is odd, then $b(n, k) = 0$ for any $k > 0$.

Proposition 2.1. *For any positive integer* m

$$b(2m, 1) = 2C_{m-1}$$

where C_{m-1} *is the Catalan number* $\frac{1}{m}\binom{2m-2}{m-1}$.

Proof. Any $\sigma = s_1 s_2 \dots s_{2m} \in B(2m)$ is balanced and has only one balanced point which is the last term and none of the terms $s_1, \dots, s_{2m-1}$ is a balance point. Now assume that $m > 1$ and $s_1 = 1$ (hence $s_{2m} = 0$). It easy to see that $s_2 = 1$ and no initial segment of the string $s_2, \dots, s_{2m-1}$ has more 0's than 1's, which means that $s_2, \dots, s_{2m-1}$ is a Dyck word of length $2m - 2$. Thus, to each Dyck word of length 2, there corresponds a $\sigma \in B(2m, 1)$ with $s_1 = 1$. Since the converse relation holds also, the number of strings in $B(2m, 1)$ with the initial term is 1 is the Catalan number $C_(m - 1)$. Including the strings with initial term 0, assertion follows. □

Proposition 2.2. *For any positive integers* m *and* $k > 1$

$$b(2m, k) = \sum_{i=1}^{m-1} b(2i, 1)b(2m - 2i, k - 1)$$

Proof. Let $k > 1$ and consider a string $\sigma \in B(n, k)$. Assume that the first balance point is s_{2t}. Then, σ can be separated into two the substrings $\sigma_1 = s_1 \dots s_{2i}$ and $\sigma_2 = s_{2i+1} \dots c_{2m}$ such that $\sigma_1 \in B(2i, 1)$ and $\sigma_2 \in (2m - 2i, k - 1)$. □

Now, let $X(n, k)$ be the set of strings which contain exactly k balance points and let $x(n, k)$ be the number of such strings. Since a term with an odd index cannot be a balance point, if n is odd, the last term cannot be a balance point and it follows that $x(n, k) = 2x(n - 1, k)$ for all nonnegative integers k. Therefore, it is sufficient to be interested only with strings of an even length.

Lemma 2.3. *The number strings of length* $2m - 1$ *such that no initial segment has more zeros than ones is* $2\binom{2m-1}{m}$.

Proof. It is well known that the number of strings of weight (total number of number of ones in the string) which satisfy the hypothesis of the lemma is $\binom{2m-1}{w} - \binom{2m-1}{w+1}$ for $w \geq m$. Summing up over all possible values of w, we obtain the desired result. □

Proposition 2.4. *For any positive integer* m*,*

$$x(2m,0) = 2\binom{2m-1}{m}.$$

Let $s_1 \dots s_n \in X(2m,0)$ with $s_1 = 1$. Since the given string has no balance point $s_2 = 1$ and in the substring $s_2 \dots s_{2m}$ no initial segment has more zeros than ones. The number such substrings of length $2m-1$ is $\binom{2m-1}{m}$. Considering the strings with $s_1 = 0$ the assertion follows.

Proposition 2.5. *For any positive integers* m *and* $k \geq 1$*,*

$$x(2m,k) = 2\sum_{i=1}^{m} b(2m,i)b(2m-2i,0).$$

Proof. Obvious from the previous results. □

We say that a given string σintersects the line $y = t$ at s_i if $2i-(s_1+\dots+s_i)$. Note that s_i is a balance point if and only if σintersects the line $y = 0$ at s_i. We let $X_t(n,k)$ to be the set of strings of length n which have intersect the line $y = t$ exactly at k distinct terms and let $x_t(n,k) = |X_t(n,k)|$. From the definition it follows that $x_0(n,k) = x(n,k) and x_t(n,k) = x_{-t}(n,k)$ for any $t = 1,\dots,n$. Let σbe a sequence which has no balance point. If $s_1 = 0$ (resp. if $s_1 = 1$), then the sequence $s_2 \dots s_n$ does not intersect the line $y = -1$ (respectively $y = 1$). Thus $x(n,0) = x_1(n-1,0) + x_1(n-1,0)$, so

$$x_1(n,0) = \frac{1}{2}x(n+1,0).$$

Moreover, if σhas no balance point and $s_1 = 0$ (respectively $s_1 = 1$) then necessarily $s_2 = 0$ (respectively, $s_2 = 1$) and the sequence $s_3 \dots s_n$ does not intersect the line $y = -2$ (respectively, $y = 2$), so

$$x_2(n,0) = \frac{1}{2}x(n+2,0).$$

Assume that σhas no balance point. If $s_1 = s_2 = 0$, then either $s_3 = 0$ and $s_4 \dots s_n$ does not intersect $y = -3$ or $s_3 = 1$ and $s_4 \dots s_n$ does not intersect $y = -1$. If $s_1 = s_2 = 1$, then either $s_3 = 0$ and $s_4 \dots s_n$ does not intersect $y = 1$ or $s_3 = 1$ and $s_4 \dots s_n$ does not intersect $y = 3$. It follows that $x(n,0) = 2x_1(n-3,0)+2x_3(n-3,0)$ which gives

$$x_3(n,0) = \frac{1}{2}2x(n+3,0) - \frac{1}{2}x(n+1,0).$$

Now consider a string σwhich intersects the line $y = 0$ exactly k times. If $s_1 = 0$ (resp. if $s_1 = 1$) , then the sequence $s_2 \dots s_n$ intersects the line $y = -1$ (respectively $y = 1$) exactly k times. Thus $x(n,k) = x_1(n-1,k) + x_1(n-1,k)$, so

$$x_1(n,k) = \frac{1}{2}x(n+1,k).$$

For a string σwhich intersects the line $y = 1$ exactly k times, there are two possibilities. Either $s_1 = 0$ and $s_2 \dots s_n \in X(n-1, k-1)$ or $s_1 = 1$ and $s_2 \dots_n \in X_2(n-1, k)$. Then we have $x_1(n, k) = x(n-1, k-1) + x_2(n-1, k)$ which gives

$$x_2(n, k) = x_1(n+1, k) - x(n, k-1).$$

When $t > 1$, for a string σwhich intersects the line $y = t$ exactly k times, there are two possibilities. Either $s_1 = 0$ and $s_2 \dots s_n \in X_(t-1)(n-1, k)$ or $s_1 = 1$ and $s_2 \dots s_n \in X_(t+1)(n-1, k)$. Then we have $x_t(n, k) = x_(t-1)(n-1, k) + x_(t+1)(n-1, k)$ which gives the recursion

$$x_t(n, k) = x_{t-1}(n+1, k) - x_{t-2}(n, k).$$

In particular, for $y = 3$, we have

$$x_3(n, k) = x_2(n+1, k) - x_1(n, k).$$

2.2.10 Random Excursion Height Test

Random Excursion Height test is a companion to Random Excursion tests. This test was previously presented in [17]. The probability distribution function was defined so that the probability values are not feasible to compute for sequences longer than 256 bits. In this thesis we review and refine the distribution function so that it is practical to compute the probability values up to 4096 bits.

Using the same notation as the Random Excursion test let $X(n, k)$ be the set of strings which contain exactly k balance points and $x(n, k)$ be the number of such strings, and $X_t(n)$ be the set of strings of length n which do not intersect the line $y = t$. Also, let $H_t(n)$ is the set of strings of length n for which t is the largest integer that they intersect the line $y = t$ and $h_t(n) = |H_t(n)|$.

In order to prove the recurrence relation we need to show some necessary equations.

Lemma 2.6. *Let n, t and q be positive integers with $t \leq q \leq n$. The number of strings of length n which contain q zeros and which intersect the line $y = t$ at least once is given by*

$$\begin{cases} \binom{n}{q-t} & q \leq \frac{n+t}{2} \\ \binom{n}{q} & q > \frac{n+t}{2} \end{cases}$$

Proof. Given a string $\sigma$of length n which intersects the line $y = t$, depending on q we consider two cases:

- $\frac{n+t}{2} < q \leq n$. In this case σnecessarily intersects the line $y = t$ and number of such strings is $\binom{n}{q}$.

- $t \leq q \leq \frac{n+t}{2}$. Let $\mathcal{A}$ be the set of strings of length n which have q zeros and which intersect the line $y = t$, and let $\mathcal{B}$ be the set of strings of length n which have $q - t$ zeros. We will show that these two sets are equivalent, so that the number of strings in $\mathcal{A}$ is $\binom{n}{q-t}$. Given $\sigma \in \mathcal{A}$. Let i_0 be the smallest integer such that σintersects the line $y = t$ at s_{i_0}. The string $\bar{\sigma} = \bar{s}_1 \dots \bar{s}_{i_0} \dots \bar{s}_n$ where $\bar{s}_i = 1 - s_i$, $i = 1, \dots, i_0$ has $q - t$ zeros, hence $\sigma \in \mathcal{B}$. Thus, to each $\sigma \in \mathcal{A}$, there corresponds a unique string $\tilde{\sigma} \in \mathcal{B}$. Conversely, any string τ in $\mathcal{B}$ has $q - t$ zeros, hence $n - q + t$ ones. On the other hand, the condition $q \leq (n + t)/2$ implies that $n - q + t \leq (n + t)/2$, which means that the string τ intersects the line $y = -t$. Now in the string τ, starting with the first term replace each one with a zero and each zero with a one up to the term at which the string intersects the line $y = -t$ for the first time. The resulting string intersects the line $y = t$ and has q zeros, hence is in $\mathcal{A}$. Then the correspondence given above is one to one and the sets $\mathcal{A}$ and $\mathcal{B}$ are equivalent.

□

By similarity, for any integer t we have $x_{-t} = x_t$ so it is sufficient to compute x_t only for non-negative values of t.

Proposition 2.7. *Let n and t be positive integers. The number of strings of length n which intersect the line $y = t$ at least once is given by*

$$\bar{x}_t(n) = \begin{cases} 2\sum_{i=0}^{\lfloor (n-t)/2 \rfloor} \binom{n}{i} & \text{if } n-t \text{ is odd} \\ 2\sum_{i=0}^{\lfloor (n-t)/2 \rfloor} \binom{n}{i} - \binom{n}{\frac{n-t}{2}} & \text{if } n-t \text{ is even} \end{cases}$$

Proof. In the previous lemma we have obtained the number of strings of length n which has q zeros and which intersect the line $y = t$. For obtaining $\bar{x}_t(n)$ we have to compute the sum of these values over all acceptable values of q: We consider two cases depending on the parity of $n + t$.

If $n + t$ is even

$$\begin{aligned} \bar{x}_t(n) &= \underbrace{\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{\frac{n-t}{2}}}_{t \leq q \leq \frac{n+t}{2}} + \underbrace{\binom{n}{\frac{n+t}{2}+1} + \cdots + \binom{n}{n}}_{q > \frac{n+t}{2}} \\ &= \binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{\frac{n-t}{2}} + \binom{n}{\frac{n-t}{2}-1} + \cdots + \binom{n}{0} \\ &= 2\sum_{i=0}^{\lfloor (n-t)/2 \rfloor} \binom{n}{i} - \binom{n}{\frac{n-t}{2}} \end{aligned}$$

If $n+t$ is odd

$$\begin{aligned}\bar{x}_t(n) &= \underbrace{\binom{n}{0}+\binom{n}{1}+\cdots+\binom{n}{\frac{n-t-1}{2}}}_{t\le q\le\frac{n+t}{2}}+\underbrace{\binom{n}{\frac{n+t+1}{2}}+\cdots+\binom{n}{n}}_{q>\frac{n+t}{2}}\\ &= \binom{n}{0}+\binom{n}{1}+\cdots+\binom{n}{\frac{n-t-1}{2}}+\binom{n}{\frac{n-t-1}{2}}+\cdots+\binom{n}{0}\\ &= 2\sum_{i=0}^{\lfloor(n-t)/2\rfloor}\binom{n}{i}\end{aligned}$$

□

Corollary 2.8. *Let n be a positive integer. The number of strings of length n which do not intersect the line $y=1$ is given by*

$$x_1(n)=\binom{n}{\lfloor\frac{n}{2}\rfloor}$$

and the number of strings which do not intersect the line $y=0$ is given by

$$x_0(n)=2\binom{n-1}{\lfloor\frac{n-1}{2}\rfloor}$$

Proof. The expression for $x_1(n)$ can be obtained directly by putting $t=1$ in Proposition 2.7. Now let $\sigma\in X_0(n)$. Assume that $s_1=1$, then necessarily $s_2=1$ and $s_1\dots s_2\in X_1(n-1)$. It follows that the number of strings in $X_0(n)$ with the first term 1 is $x_1(n-1)=\binom{n-1}{\lfloor\frac{n-1}{2}\rfloor}$. Since the same holds for the strings with the first term 0, we obtain the result.

□

We observe that smallest possible value for the height of a string is -1: $h_t(n)=0$ for all positive integers n and for all negative integers $t<-1$. Next proposition gives the number of strings for all possible values of the height.

Proposition 2.9.

$$\begin{aligned}h_{-1}(n) &= \binom{n-1}{\lfloor\frac{n-1}{2}\rfloor}\\ h_0(n) &= \binom{n}{\lfloor\frac{n}{2}\rfloor}-\binom{n-1}{\lfloor\frac{n-1}{2}\rfloor}\end{aligned}$$

and for all t such that $1\le t\le n$

$$h_t(n) = \binom{n}{\lfloor\frac{n-t}{2}\rfloor}$$

Proof. There are $x_0(n)$ strings of length which do not intersect the line $y = 0$. Given a string $\sigma \in X_0(n)$, if $s_1 = 0$, then σ can never intersect a line $y = t$ for negative values of t and consequently height of σ is necessarily positive. If $s_1 = 1$ then σ can intersect only the lines $y = t$ with $t < 0$ and in such a case height of σ is -1, then $h_{-1}(n) = 1/2x_0(n)$. The number of strings of length n which do not intersect the line $y = 1$ is $x_1(n)$ and height of any such function is at most 0. Since there are $\mathrm{h}_{-1}(n)$ strings with height -1, we get $h_0(n) = x_1(n) - h_{-1}(n)$. If n is even, say $n = 2m$, then

$$\begin{aligned} h_0(n) &= x_1(2m) - h_{-1}(2m) \\ &= \binom{2m}{m} - \binom{2m-1}{m} \\ &= \binom{2m-1}{m} \\ &= \binom{n-1}{\frac{n-2}{2}} \end{aligned}$$

If n is odd, say $n = 2m + 1$, then

$$\begin{aligned} h_0(n) &= x_1(2m+1) - h_{-1}(2m+1) \\ &= \binom{2m+1}{m} - \binom{2m}{m} \\ &= \binom{2m}{m-1} \\ &= \binom{n-1}{\lfloor \frac{n-2}{2} \rfloor} \end{aligned}$$

For any positive integer t, any string which intersects the line $y = t + 1$ intersects the line $y = t$ as well. Hence $X_{t+1}(n, 0) \subset X_t(n, 0)$ and $H_t(n) = X_t(n) \setminus X_{t+1}(n)$. Then $h_t(n) = x_t(n) - x_{t+1}(n)$ and the assertion follows from Proposition 2.7 by straightforward computations. □

There are two strings of length 1: 0 and 1. Heights of these strings are 1 and -1, so that $h_{-1}(1) = 1$, $h_0(1) = 0$, $h_1(1) = 1$. With these initial values and the $h_t(n) = 0$ whenever $t > n$, we can compute $h_t(n)$ recursively as follows:

$$\begin{aligned} h_{-1}(n) &= h_{-1}(n-1) + h_0(n-1) \\ h_0(n) &= h_1(n-1) \\ h_1(n) &= h_{-1}(n) + h_2(n-1) \end{aligned}$$

and for $t = 2, \ldots, n$

$$h_t(n) = h_{t-1}(n-1) + h_{t+1}(n-1)$$

2.2.11 Template Matching Test

Template matching test is given in the NIST Test Suite as "Overlappipng Template Test" and "Nonoverlapping Template Matching Test". However, the computations for the overlapping test is shown to be wrong. In this work we use the calculations from [18] and restate the test.

Let $T_{\mathcal{B}}(n,k)$ be the number of strings of length n which contain the template $\mathcal{B}$exactly k times. For any fixed template $\mathcal{B}$ of length t, the sequence $\{T_{\mathcal{B}}(n,0)\}_n$ satisfies a constant coefficient, linear and homogeneous recursion of order t. This recursion depends only on the overlap number m of $\mathcal{B}$ and t. That is, if $\mathcal{D}$ is another m-overlapping template of length t, then the sequence $\{T_{\mathcal{D}}(n,0)\}_n$ satisfy the same linear recursion with$\{T_{\mathcal{B}}(n,0)\}_n$. Since the recursion depends only on n and m we can drop the template from the notation noting m: $T_(n,k)$.

Lemma 2.10. *Given a m-overlapping template $\mathcal{B}$of length t and an arbitrary binary sequence δ of length not exceeding t. Let u_n denote the number of binary strings of length n starting (or ending) with δ which do not contain $\mathcal{B}$. The sequence u_n satisfy the recursion of $\{T_{\mathcal{B}}(t,m)\}_n$.*

Theorem 2.11. *For any non-negative integers m and t, the sequence $\{T_{\mathcal{B}}(t,m)\}_n$ satisfies a constant coefficient, linear and homogeneous recursion of order 2t. Characteristic polynomial of this recursion is the square of that of $\{T_{\mathcal{B}}(t,0)\}_n$.*

Theorem 2.12. *For any non-negative integers m and t, the sequence $\{T_{\mathcal{B}}(n,k)\}_n$ satisfies a constant coefficient, linear and homogeneous recursion of order $(k+1)t$. Characteristic polynomial of this recursion is the product of those of $\{T_{\mathcal{B}}(n,0)\}_n$ and $\{T_{\mathcal{B}}(n,k-1)\}_n$.*

Let $T(n,k)$ be the number of binary strings of length n with k overlapping substrings of length 4. Using the above theorems and the results given in[REFERANS], then, the recursions of the overlapping structures are given below.

0-overlapping substrings: 0001, 0011, 0111, 1000, 1100, 1110.

$$T(n,0) = 2T(n-1,0) - T(n-4,0)$$

$$T(n,k) = \begin{cases} 0 & n < 4k \\ 1 & n = 4k \\ 2T(n-1,k) - T(n-4,k) + T(n-4,k-1) & n > 4k \end{cases}$$

1-overlapping substrings: 0010, 0100, 1011, 1101, 0110, 1100.

$$T(n,0) = 2T(n-1,0) - T(n-3,0) + T(n-4,0)$$

$$T(n,k) = \begin{cases} 0 & n < 3k+1 \\ 1 & n = 3k+1 \\ 2T(n-1,k) - T(n-3,k) + T(n-4,k) & \\ +T(n-3,k-1) - T(n-4,k-1) & n > 3k+1 \end{cases}$$

2-overlapping substrings: 0101, 1010.

$$T(n,0) = 2T(n-1,0) - T(n-2,0) + 2T(n-3,0) - T(n-4,0)$$

$$T(n,k) = \begin{cases} 0 & n < 2k+2 \\ 1 & n = 2k+2 \\ 2T(n-1,k) - T(n-2,k) + 2T(n-3,k) - T(n-4,k) & \\ +T(n-2,k-1) - 2T(n-3,k-1) + T(n-4,k-1)) & n > 2k+2 \end{cases}$$

3-overlapping substrings: 0000, 1111.

$$T(n,0) = T(n-1,0) + T(n-2,0) + T(n-3,0) + T(n-4,0)$$

$$T(n,k) = \begin{cases} 0 & n < k+3 \\ 1 & n = k+3 \\ T(n-1,k) + T(n-2,k) + T(n-3,k) + T(n-4,k) & \\ +T(n-1,k-1) - T(n-2,k-1) - T(n-3,k-1) & \\ -T(n-4,k-1)) & n > k+3 \end{cases}$$

2.2.12 Autocorrelation Test

Autocorrelation is one of the measures of randomness in Golomb's postulates[8]. Nevertheless, test suites does not include any autocorrelation test or equivalent.

The autocorrelation test examines possible repetitions with small periodicity within the sequence. This test compares the first half of the sequence with $\sigma_{i \to n/2+i}$ at i^{th} iteration. The result is the weight of the $\frac{n}{2}$-bit sequence obtained by XORing the two subsequences. It is expected that there is no periodic patterns in the sequence, therefore, the XOR of the sequences should be a random sequence. The distribution function of auto correlation function, therefore, is equal to the distribution function of weight test for $\frac{n}{2}$:

$$F_n(k) = 2^{-n} \sum_{i=0}^{k} \binom{n/2}{i}. \tag{2.11}$$

Other approaches can be followed like the maximum, or minimum, value of the correlation among many shifts, the distribution of the autocorrelation values and so on.

However, it is important to note that, subsequences generated from consecutive shift values are correlated and it is hard to generate the distribution function for these approaches. Therefore, we choose a simpler way to test the sequence in terms of autocorrelation.

2.2.13 Integer Tests

2.2.13.1 Integer Frequency Test

Integer frequency test appears in various test suites in various names including Knuth[1] and Diehard[4]. In this work, we give the actual probability values and give a recursion for large n values.

Integer frequency test converts the sequence to an integer sequence and checks the frequencies of each integer in the new sequence.

First the sequence is converted to an integer sequence. Then, the frequencies of all possible integers are computed and compared to the expected frequencies to get an idea about the randomness of the sequence.

The conversion from binary sequence to the integer sequence needs careful attention since it affects the sample space size. For an n-bit sequence, if the block size of the integers are chosen to be m there will be $\frac{n}{m}$ terms in the converted sequence and each term will be one of 2^m integers. Then, the expected frequencies of each element will be

$$f_e = 2^{-m}\frac{n}{m}. \tag{2.12}$$

f_e should be large enough in order for the results to be acceptable.

In this work, we choose $m = 3$ for $n = 128$. In this setting, there will be 42 terms in the integer sequence and each integer will have an expected frequency of 5,2. This way one can apply a χ^2 goodness-of-fit test on the frequencies of the integers for a single sequence by computing the expected values from Equation 2.12. However, the recommended application of the test is on a set of sequences.

For an efficient recursion, fix the integers b and l, and let $P(k) = Prob(X_a = k)$. Probability values for $k = 0, 1, \ldots$ can be evaluated recursively by

$$P(0) = \left(1 - \frac{1}{M}\right)^l$$

and for $k = 1, 2, \ldots, M-1$

$$P(k) = \frac{1}{M-1} \cdot \frac{l+1-k}{k} P(k-1).$$

Now we have

$$\begin{aligned} F_l(0) &= \left(1-\frac{1}{M}\right)^l \\ F_l(1) &= \left(l+1-\frac{1}{M}\right)\left(1-\frac{1}{M}\right)^{l-1} \end{aligned}$$

and for $k \geq 2$

$$F_l(k) = F_l(k-1) + \frac{l+1-k}{k(M-1)}[F_l(k-1) + F_l(k-2)].$$

2.2.13.2 Integer Maximum Test

Integer maximum test corresponds to the Max-of-t test given in Knuth test suite. In this work, we give the distribution function with actual probability values.

Integer maximum test examines the maximum integer value in the sequence and compares to the expected maximum value to check the randomness of the sequence.

First the sequence is converted to $m-$bit integer sequence. Then, the maximum term of the sequence is output as the t-value. The process is repeated for many sequences a χ^2 goodness-of-fit test is applied to the observed values.

Let $l = \frac{n}{m}$ be the number of terms of the integer sequence. Then the probability that the maximum term to be smaller than or equal to k can be computed as follows. In a sequence with m-bit terms, there are $M = 2^m$ possible integers for each term where $k+1$ of them are smaller than or equal to k. This means, the probability of a term to be smaller than or equal to k is

$$P(\sigma_i \leq k) = \frac{k+1}{M}.$$

Therefore, probability that every term of the sequence to be smaller than or equal to k, ie maximal term, x, to be smaller than k, is

$$P(x \leq k) = \left(\frac{k+1}{M}\right)^l.$$

In this work, we recommend setting $m = 8$ for $n = 128$ and applying χ^2 goodness-of-fit test on a set of sequences.

2.2.13.3 Integer Minimum Test

Although integer maximum test occurs in various suites, minimum test is not used to measure randomness. In this work, we give the distribution function with actual probability values.

Integer minimum test considers the minimum integer value in the sequence and outputs a p-value regarding the minimum value.

Similar to the integer maximum test, initially the sequence is converted to $m-$bit integer sequence. Then, the minimum term in the sequence is output as the t-value. The test is applied on various sequences and a χ^2 goodness-of-fit test is applied to the observed values.

Let $l = \frac{n}{m}$ be the number of terms of the integer sequence. The probability of a term to be greater than k is $\frac{M-k}{M}$ where $M = 2^m$ and probability that all the terms are greater than k is

$$P(\sigma_i > k) = \left(\frac{M-k}{M}\right)^l.$$

Therefore, the probability that no term is greater than or equal to k, ie minimum term, x, of the sequence is greater than or equal to k, is

$$P(x > k) = 1 - \left(\frac{M-k}{M}\right)^l.$$

Similar to the integer maximum test, it is recommended to take $m = 8$ for $n = 128$ and to apply χ^2 goodness-of-fit test on a set of sequences.

2.2.13.4 Integer Maximum Minimum Difference Test

Max-min difference is an integer test that has not appeared on any test suite yet. The test can be found in

Integer minimum test examines the difference between the maximum and the minimum terms of the sequence.

The test converts the sequence into m-bit integer sequence. Then, finds the maximum, σ_{max}, and minimum, σ_{min}, terms and computes the difference between these two terms: $\sigma_{max} - \sigma_{min}$. Then, a χ^2 goodness-of-fit test is applied to the observed differences of generator output sequences.

The probability of the difference d_{max} being k can be computed as follows.

First, for the maximum-minimum difference to be equal to 0, all the sequence should be equal. The probability of such an event is, therefore:

$$Prob(T = 0) = l\left(\frac{1}{M}\right)^l.$$

For $k \geq 1$, by principle of inclusion-exclusion we write

$$Prob(T = k) = (l - k)\left[\left(\frac{k+1}{M}\right)^l - 2\left(\frac{k}{M}\right)^l + \left(\frac{k-1}{M}\right)^l\right].$$

It follows that

$$F_l(0) = l\left(\frac{1}{M}\right)^l$$

and for $k \geq 1$

$$\begin{aligned} F_l(k) &= \sum_{i=0}^{k} Prob(T = k) \\ &= (l - k)\left(\frac{k+1^l}{M}\right) + (k + 1 - l)\left(\frac{k}{M}\right)^l. \end{aligned}$$

2.2.13.5 Integer Coverage Test

Integer coverage test is derived from the well known coverage concept. This test ha not been appeared in a test suite, however, the details can be found in [19].

Coverage test is an integer test the coverage of the sequence. The coverage of the test is defined as the number of distinct elements in the sequence.

The test is an integer sequence test, therefore, the bit sequence should be transferred into an integer sequence in a non-overlapping fashion. The number of distinct integers is the t-value of the test.

The number of n bit sequences containing exactly k distinct elements is calculated as follows. First, the number of distinct k integer selections out of M integers is $\binom{M}{k}$. Then, since $l \geq k$, some elements can appear more than once in the sequence where $l = \frac{n}{m}$. There are k distinct integers and the sum of the frequencies of these integers are l. The number of such ordered arrangements is equal to the partitions of l into k which is the Stirling number of the second kind l of k, $\left\{ {l \atop k} \right\}$. Finally, considering the $k!$

orderings of k elements within these arrangements the total number of n bit sequences containing exactly k distinct elements is

$$\binom{M}{k} \left\{ {l \atop k} \right\} k!.$$

Then, the probability of such sequences is

$$M^{-l} \binom{M}{k} \left\{ {l \atop k} \right\} k!. \tag{2.13}$$

For the recursions we have

$$P_l(1) = \frac{1}{M^{l-1}}.$$

and using the basic recursion for Stirling numbers of the second kind we get

$$\begin{aligned} P_l(k) &= \frac{k!}{M^l} \binom{M}{k} \left[k \left\{ {l-1 \atop k} \right\} + \left\{ {l-1 \atop k-1} \right\} \right] \\ &= \frac{k}{M} \cdot \frac{k!}{M^{l-1}} \binom{M}{k} \left\{ {l-1 \atop k} \right\} + \frac{M-k+1}{M} \cdot \frac{(k-1)!}{M^{l-1}} \binom{M}{k-1} \left\{ {l-1 \atop k-1} \right\} \\ &= \frac{k}{M} P_{l-1}(k) + \left(1 - \frac{k-1}{M}\right) P_{l-1}(k-1) \end{aligned}$$

for $k = 2, 3, \ldots$.

2.2.13.6 Integer Repetition Test

Integer repetition test examines the first index of the repetition in the sequence.

First the sequence is converted to b-bit integer sequence and then, starting from the first term, each term is compared to the predecessor terms. The index, where the first repetition occurs, is the t-value of the test.

Assume the first repetition occurs at k^{th} point. That is, the first $k-1$ terms are distinct and the k^{th} term is equal to one of the first $k-1$ terms. Then, one can choose $k-1$ distinct integers out of M for the first $k-1$ terms and these terms can be arranged in $(k-1)!$ ways. For the k^{th} element there are $k-1$ possible values. Therefore, there are

$$\binom{M}{k-1} (k-1)!(k-1)$$

and the probability of a repetition to occur at k^{th} index is

$$M^{-k}\binom{M}{k-1}(k-1)!(k-1). \tag{2.14}$$

2.2.13.7 Integer Saturation Test

Integer Saturation test is the equivalent of Knuth's Coupon Collector test. In this work we set the suitable parameters and give a recursion for efficient computation of the probabilities. The test details can be found in [19].

Saturation test investigates the sequence in terms of the length of the shortest subsequence that includes all possible m-bit integers.

After the sequence is converted to m-bit integer sequence, the terms are traced until all m-bit integers are occurred. The index of the last integer, saturation point, is the t-value of the test.

In order to have k as the saturation point, the first $k-1$ terms of the sequence must cover $M-1$ distinct integers, and the k^{th} term must be the missing one in the first $k-1$ terms. Probability of the first $k-1$ terms of the sequence covering $M-1$ distinct integers is

$$\frac{\binom{M}{M-1}\left\{ {k-1 \atop M-1} \right\}(M-1)!}{M^{k-1}}.$$

The last term is the non-appearing integer with probability $\frac{1}{M}$. Then, the probability of a sequence having saturation point k is

$$\begin{aligned} P(SP=k) &= M^{-(k-1)}\binom{M}{k-1}\left\{ {k-1 \atop M-1} \right\}(M-1)!\frac{1}{M} \\ &= M^{-(k-1)}\left\{ {k-1 \atop M-1} \right\}(M-1)!. \end{aligned}$$

2.2.13.8 Universal Test

Universal test[20] deals with the distance between successive occurrences of elements. The test is applied on the integer sequences. Therefore, first, binary sequences are converted to b-bit integer sequences. Then, a distance sequence $\mathcal{D}$ is generated as follows. $\mathcal{D} = d_1 d_2..d_n$ st

$$d_i = \begin{cases} j-i & j \text{ is the smallest index st } j > i \text{ and } u_i = u_j \\ 0 & \text{if } i \text{ is the last index of } u_i \end{cases}$$

Then, the observed frequencies, $\mathcal{F}_i$, in the sequence $\mathcal{D}$ are compared to the expected numbers $\mathcal{E}_i$.

For a term t in $\tilde{\sigma}$, in order for the distance between its next occurrence to be $d \neq 0$, there should be $d-1$ elements which are not equal to t, and the next element should be t. The probability of such a case is

$$P(d) = \left(\frac{M-1}{M}\right)^{d-1} \frac{1}{M}. \tag{2.15}$$

For $d = 0$, the probability can be computed by

$$P(0) = 1 - \sum_{i=1}^{l} \left(\frac{M-1}{M}\right)^{i-1} \frac{1}{M}. \tag{2.16}$$

Notice that, the last term d_n should always be zero as it is the last occurrence of any integer. Also, d_{n-1} is either1 or 0. Continuing this way, the distance d can appear in $l-d$ places. So the expected frequencies of the distances are

$$E_a = (l-a)P(d). \tag{2.17}$$

The χ^2 value can bi computed via

$$\chi^2_\sigma = \sum_{d=0}^{l-1} \frac{(E_d - T_d)^2}{E_d} \tag{2.18}$$

and the $p-$ value can be computed using χ^2 test with χ^2_σ and degree of freedom $n-1$.

CHAPTER 3

CORRELATION OF STATISTICAL RANDOMNESS TESTS

Randomness testing is an expensive process in terms of time and computing power: first a large set of sequences are produced, then, a set of statistical randomness tests are applied on these sequences and the results are evaluated to conclude if the sequence or the generator is non-random. So, it is not feasible to run all known tests in all possible settings. Therefore, user should choose a subset of tests that runs in a practical time. However, this subset of tests should contain as many tests to make the correct decision.

In such a process, if some or all the selected tests are correlated, then the results of these tests will also be similar. This, not only wastes time and computing power, but also may lead to false conclusions on the tested sequences. Therefore, in order to conclude a reliable decision on the randomness, it is important for the tests to be uncorrelated.

Correlation can be investigated experimentally by examining the test values and $p-$ values of tests. One can apply correlation detection methods on the output sets of statistical tests and conclude whether tests are correlated or not. In this section, the correlation detection methods are applied on the data sets and correlations between tests are deduced.

3.1 Pearson Correlation Coefficient

There are various ways to determine the correlation between random variables. The most common method is **the Pearson Product-Moment Correlation** method. The value of the Pearson coefficient is a measure of the linear correlation between two random variable sets. The correlation coefficient of random variables X and Y can be calculated using the Equation 3.1:

$$Corr(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y}, \tag{3.1}$$

where $cov(X,Y)$ is the covariance between X and Y and σ_X is the standard deviation of X.

We calculate the Pearson correlation values for each test pair defined in Chapter 2 and

present the results in Tables A.2, A.3, A.4 and A.5. In order for the tables to fit we rename the tests and the corresponding test names are given in Table A.1. In theory, for the tests to be uncorrelated all the values should be 0. On the other hand, In practice there may be some deviations from 0 even if the tests are uncorrelated which can be caused by the data selection. However, in the tables there are correlation values higher than 0.1 which cannot be ignored.

The tables suggest the following statements.

- There is a correlation among the Random Excursion(1), Random Excursion(2) and Random Excursion(3) tests, and Random Excursion Height test.
- There is a correlation among the Random Excursion(-1), Random Excursion(-2) and Random Excursion(-3) tests.
- Random Excursion(1) and Random Excursion(-1) tests are correlated to Random Excursion Test.
- Weight test and Height test have a higher correlation value than expected.
- Weight test and Integer Frequency tests are highly correlated.
- Number of Total Runs test is correlated to Runs of Length 1 and Integer Frequency tests.

However, Pearson correlation values are not a strong measure and should be supported with other instruments. Therefore, we need to define new measures for determining the correlation between tests. For this reason we compare the Fail-Fail Ratio, the reactions of the tests on the changes and deterministic structures in the sequence.

3.2 *Fail-Fail* Ratio

The primary result of a randomness test is whether the sequence passed the test or failed. Therefore, it is convenient to base the correlation on the failed sequences. The behaviour of a test on the failed sequences from another test can give some idea about the correlation between these two tests. For instance, if tests $T1$ and $T2$ are uncorrelated, then it is expected that any sequence failed from $T1$ fails from $T2$ with the same fail probability of any sequence in the sequence space. That is, failing from $T1$ should not affect the result of the test $T2$. Any other result can indicate a relation between the mentioned tests. This way, the fail probability of $T1$ on the sequences that fail from $T2$ can be used as a correlation measure. We call this probability *Fail-Fail Ratio (FFR)*[21]. Also, *Pass-Fail(PFR)* and *Pass-Pass (PPR)* ratios can be defined in a similar manner. However, our experiments show that these two ratios are not as reliable as *FFR*.

We applied statistical randomness tests on 10000 sets, each consisting of 1000 sequences, and checked the *FFR*. The sequences are the output of *AES* algorithm, with a

fixed key ad-bd incremental plaintexts. The plaintexts start from all zero 128-bits and at each iteration we increment the integer value of the plaintext by one. The results are given in Table B.1. As the tables is too large to fit here, we present only the rows with correlation values higher than 0,4 for *FFR*.

Table B.1 indicates the following.

- Random Excursion tests for $y = 1, 2, 3$ are mutually highly correlated, and correlated to weight test.
- Random Excursion tests for $y = -1, -2, -3$ are mutually highly correlated and highly correlated to Pseudo Complexity Profile and Weight tests.
- Random Height and Random Excursion for $y = 0$ are mutually highly correlated.
- Runs of Length 5 and Runs of Length > 5 tests are mutually correlated to each other and highly correlated to Longest Run and Random Excursion tests for $y = 1, 2, 3$.
- Minimum test is highly correlated to Pseudo Complexity, Pseudo Complexity Profile, Weight and Number of Total Runs tests. Moreover, Minimum test is correlated to Maximum and Maximum-Minimum Difference tests.

Besides the above inferences, there are some other results that can be derived from the Table B.1. However, these results are weaker and need to be justified by increasing the number of tests.

In order to check the reliability of this method, we apply *Fail-Fail Ratio* on the short sequence tests in the NIST Statistical Test Suite. As our primary aim is to test 128 bit sequence sets, we choose 9 short sequence tests from this suite and applied on the same data sets with previous experiment. The results are given in Table C.1.

According to Table C.1, Frequency and Block Frequency tests, and Approximate Entropy, Serial-1 and Serial-2 tests are correlated. This result is compliant with the previous results[22]. This conformity encourages the use of *Fail-Fail Ratio* as a correlation detection tool.

3.3 Transformations

Another method for observing the correlation between two tests is analysing the reactions of the tests to the changes in the sequence[23]. For this purpose, first a set of sequences are tested, and a transformation is applied on the sequences. Then, the correlation between the original sequence results and the transformed sequence results are computed. This way, the characteristic similarities between tests can be detected. This similarities can also be used for classification of the tests. For instance, if, for a large space of sequences test T_1 detects the complementation of a sequence but test

T_2 does not, then these two tests are probably in distinct classes and the correlation between these two tests is not expected to be significant. Conversely, if two tests show similar behaviour on the same transformations then these tests can be considered to be in the same class and there may be a notable correlation between these tests. By increasing the number of experiments and transformations, a reliable classification can be achieved.

For this purpose, we define some transformations π_i, and calculate the correlation value between $T(\sigma)$ and $T(\pi_i(\sigma))$ for each transformation π_i. Then, according to the correlation values, we classify the tests. During this process we define a number of transformations and select the ones that contribute to classification. For instance, if all tests can detect a transformation (or none of them cannot detect) then we exclude this transformation. The transformations used in the classification are given below.

1. **Complement:** This transformation complements all the sequence: $C(\sigma) = \tilde{\sigma}$ such that $\hat{s}_i = 1 \oplus s_i$.

2. **Swap Bits:** The *Swap Bits* transformation swaps two successive bits in the sequence: $B(\sigma) = \tilde{\sigma}$ such that $\tilde{\sigma} = s_2 s_1 s_4 s_3 s_5 s_6 \ldots$.

3. **Swap Halves:** Swaps the first half of the sequence with the last half: $\sigma = A||B,\ H(\sigma) = B||A$.

4. **Reverse:** *Reverse* transformation reverses the sequence: $R(\sigma) = s_n s_{n-1} \ldots s_1$.

5. **Swap Half-Reverse Last:** This transformation first swaps the halves of the sequence and then reverses the last half of the swapped sequence: $\sigma = A||B$ then $HR(\sigma) = B||R(A)$.

6. **Reverse Halves:** *Reverse Halves* transformation reverses each half of the sequence: if $\sigma = A||B$ then $RH(\sigma) = R(A)||R(B)$.

7. **Complement Reverse:** This transformation reverses the complement of the sequence $CR(\sigma) = C(R(\sigma))$.

To clarify the process, consider the Total Number of Runs test. This test cannot detect the complementation of the sequence as the number of runs in the sequence σ and its complement is the same. Therefore, the output values for the σ and $C(\sigma)$ are equal. So, the correlation value of the Total Number of Runs test corresponding to complement transformation is 1.

We apply all the tests with the defined transformations above and compute the correlation values between the outputs of the original sequences and the outputs of the transformed sequences. The results are given in Table D.1. According to this table, the tests can be classified as in Table 3.1.

Table 3.1: Classification of Tests.

Complexity Tests
Autocorrelation Test
Linear Complexity Test
Linear Complexity Profile Test
Pseudo Complexity Test
Integer Complexity Tests
Universal Test

Max-Min Tests
Maximum Test
Minimum Test
Max-Min Difference Test
Excursion Tests
Random Excursion Tests[2]

Frequency Tests
Integer Frequency Test
Weight Test
Coverage Tests
Coverage Test
Repetition Test
Saturation Test

Run Tests
Random Height Test
Template Matching Test
Longest Run Test
Run of Length k Tests[1]
Total Runs
Pseudo Complexity Profile Test

[1]This set includes $k = 1, 2, 3, 4, 5$ and $k > 5$.

[2]This class includes Random Excursion Tests(0),(± 1),(± 2),(± 3)

CHAPTER 4

BUILDING A TEST SUITE

Designing a test suite is not a trivial task. On the contrary, one should follow some certain rules in order to build a reliable suite. Piling up all the known tests makes an impractical test suite which will be useless. The tests in the suite should be selected carefully considering the correlation of the tests, coverage of the test suite, classes of the tests and so on. Important criteria in the test suite design process are given below.

Test Subject
The tests should be selected among the ones that are based on mathematical backgrounds and meaningful in terms of randomness and statistics. The tests should give feedback about the quantity and quality of the tested feature.

Coverage
The coverage of the suit is defined as the ratio between the number of sequences that fail from at least one of the test and the sample set size, $\frac{|F|}{|\omega|}$. Then, the coverage of the suit should be evaluated after the tests are selected and it should be close to the expected coverage. If the coverage value is smaller then the expected value, then is a sign of correlated or weak tests in the suite.

Correlation
The tests in the suite should be uncorrelated. Correlated tests not only wastes time and computing power, but also may lead to wrong decisions for the randomness of the sequence or the generator. Besides, the correlation of the tests affect the coverage of the suite. The lower the correlation, the higher the coverage. Moreover, it is a good practice to inform the users about the correlations of the tests in the suite and the coverage of the overall test suite. This way, if a smaller subset of the suite is intended to be applied, the user can select the tests with lower correlation and higher coverage.

Marginal Coverage
The marginal benefits of the tests should be considered when including a new test to the suite. For instance, assume there are two tests $T1$ and $T2$ in a suit with coverage value c_1. When the test $T3$ is added to the suite, assume the new coverage becomes c_2. For uncorrelated tests, $\Delta(c) = c_1 - c_2$ should be close to the significance level. If $\Delta(c)$ is smaller than the significance level, then either $T3$ is a weak test, or it is correlated to the tests in the suite. In any case, $T3$ should be excluded from the suite.

Diversity
For a comprehensive test suite, there should be tests from various classes. For instance, besides the tests examining the frequencies of terms, there should be tests checking the complexity of the sequence. The suite, unless it was aimed to measure a specific property, should include at least one test from each class.

Aim
The aim of the suite should be reflected on the test choice. If, for instance, short sequences are aimed to be tested, then the tests in the suite should be able to test short sequences. Similarly, if the aim is to test random number generators of long sequences, the included tests should be suitable for testing bunches of long sequences. Besides, the type of the sequences, integer or binary, should be considered and the tests should be selected accordingly.

4.1 A Sample Test Suite

In this section, a sample selection process is applied on the tests mentioned in Section 2. The purpose of the selection is to design a test suite with high coverage and low correlation by including as least tests as possible.

First, it is convenient to go class-by-class in order to have at least one test from each class.

- **Complexity Tests:** All 4 tests in this class have coverage very close to $0,01$. However, among them, Linear Complexity Profile test has the least correlation values with the tests from other classes. So, it is a good practice to select Linear Complexity Profile test. Besides, running this test also implies running Linear Complexity Test. Therefore, one can select these two tests from the complexity class.

- **Integer Complexity Tests:** In this class there is only one test, Universal Test, exists. Therefore, selection of Universal Test should be left after the selection of other classes according to the correlations with the selected tests.

- **Max-Min Tests:** In this set the Integer Maximum Minimum Difference test has a higher correlation value than Maximum and Minimum tests. Considering the correlation values where p-value< 0.05, the Minimum test has higher correlation values than the Maximum test. Therefore, Maximum test can be selected from this class.

- **Random Excursion Tests:** The Random Excursion(1), Random Excursion(2) and Random Excursion(3) tests are highly correlated to each other and to Random Excursion Height test. Therefore, if selected, one needs to select only one of these tests. Similarly, The Random Excursion(-1), Random Excursion(-2) and Random Excursion(-3) tests are highly correlated and it is enough to choose one of them. Also there are correlations between Random Excursion-Random Excursion(1) and Random Excursion-Random Excursion(-1). Considering the cor-

relations of Random Excursion-Random Excursion(2) and Random Excursion-Random Excursion(3), Random Excursion test is a good choice. For the remaining tests, Random Excursion(-2) and Random Excursion(-3), the former has higher correlation values with other tests. Therefore, the selected tests from this class are Random Excursion and Random Excursion(-3).

- **Frequency Tests:** The frequency distribution of the terms in the sequence is a very basic condition on the randomness of the sequence and affects the properties that other tests control. Although the correlation of weight and Integer Frequency tests are medium, they are correlated to many tests in the set. Therefore, it is a better choice to reserve the frequency tests for the later decisions and select according to the correlation values with the selected tests.

- **Coverage Tests:** The correlation values and coverages of all three tests are very close to each other. In this class, selection can be made according to the priority of the test suite. Otherwise, Repetition Test is faster than the other two tests and can be selected primarily.

- **Run Tests:** In this class Total Runs and Pseudo Complexity Profile are correlated to Maximum test. Random Height Test and Random Excursion Tests are correlated to each other. Also, Run Length> 5, Runs of Length 5 and Longest Run tests are correlated. Template Matching Test can be selected among the remaining tests. If a second test is to be chosen, it is one of the Run-2, Run-3 and Run-4 tests. Among these three tests, if we check the correlation where p-value < 0.05, the most uncorrelated test is Run-2 test. That is, the Template Matching and Run-2 tests can be selected from this class.

At this point, when the correlation values of Universal test and the selected tests are computed, and the marginal benefit of the Universal test is evaluated, the results suggest that the Universal Test should be omitted. The marginal benefit of the Universal Test is less than 0.005 and if selected, it will increase the correlation of the suite.

From the above discussion, an example suite consists of the following tests: Linear Complexity Profile, Linear Complexity, Maximum, Random Excursion, Random Excursion(-3), Repetition, Template Matching and Run-2 Tests. The coverage of the suite is 0.0762 in contrast to the expected 0.077255306 which is very close. Moreover, the correlation among the tests are considerably low. Thanks to the selection process, the suite includes at least one test from each class, except the frequency class. Actually, our experiments show that, the probability of a sequence that fails from frequency class tests but passes the selected suite tests is about 0.0001 which can be ignored.

CHAPTER 5

CONCLUSION

In this thesis, we examine the statistical randomness tests in the literature. We go over 51 tests and select the tests with reasonable measure and calculable distribution functions. We review the selected tests and compute the distribution functions so that actual probability values can be computed for sequences shorter than 4096 bits. We, also, define new tests for more sensitive results. At the end, we left with 20 tests and 30 p-values.

By using correlation detection methods, we find the correlations between each test and using the correlation values we classify the selected tests. Finally, we stress some important points in building a test suite and present an example test suite using the tests mentioned in this thesis.

The focus of this work is testing the set of sequences, like encryption keys, that cannot be tested using the present test suites. As a future work, we are aiming to extend our results for longer sequences and contribute to the effectiveness and efficiency of the statistical randomness testing.

REFERENCES

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

[2] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical report, NIST, 2001.

[3] P. L'Ecuyer and R. Simard. Testu01: A c library for empirical testing of random number generators. *ACM Trans. Math. Softw.*, 33(4):22, 2007.

[4] G. Marsaglia. The Marsaglia random number CDROM including the DIEHARD battery of tests of randomness, 1996.

[5] R. G. Brown. Dieharder: A random number test suite, 2013.

[6] L. Nielsen W. Caelli, E. Dawson and H. Gustafson. Crypt–x statistical package manual, measuring the strength of stream and block ciphers, 1992.

[7] Juan Soto and Lawrence Bassham. Randomness testing of the advanced encryption standard finalist candidates. In *NIST IR 6483, National Institute of Standards and Technology*, 1999.

[8] Solomon W. Golomb. *Shift Register Sequences – A Retrospective Account*, pages 1–4. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[9] Kenji Hamano and Toshinobu Kaneko. Correction of overlapping template matching test included in nist randomness test suite. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, E90-A(9):1788–1792, September 2007.

[10] Yamaguchi Kenji Nakamuro Katsuhiro Okutomi Hidetoshi, Kaneda Manabu. Study on the randomness evaluation method using nist randomness test. *Symposium on Cryptography and Information Security*, 2006.

[11] Fatih Sulak Muhiddin Uğuz, Ali Doğanaksoy. A new randomness test based on the overlapping blocks. *Combinatorics 2016*, 2016.

[12] Elwyn R. Berlekamp. *Algebraic coding theory*. McGraw-Hill series in systems science. McGraw-Hill, 1968, New York.

[13] James L. Massey. Shift-register synthesis and BCH decoding. *IEEE Trans. Information Theory*, 15(1):122–127, 1969.

[14] Rainer A. Rueppel. *Analysis and Design of Stream Ciphers*. Springer-Verlag, Berlin. An optional note.

[15] Hu Qi. Stream ciphers and linear complexity, 2007.

[16] M.Z. Wang. Linear complexity profiles and jump complexity. *Information Processing Letters*, 61(3):165 – 168, 1997.

[17] Fatih Sulak Meltem Sönmez Turan Ali Doğanaksoy, Cagdas Calık. New randomness tests using random walk. *National Cryptology Symposium II, TURKEY*, 2006.

[18] Fatih Sulak. New statistical randomness tests: 4-bit template matching tests. *Turkish Journal of Mathematics*, pages –, 2016. to appear.

[19] Fatih Sulak. *STATISTICAL ANALYSIS OF BLOCK CIPHERS AND HASH FUNCTIONS*. PhD thesis, MIDDLE EAST TECHNICAL UNIVERSITY, Ankara, 2011.

[20] Cihangir Tezcan Ali Doğanaksoy. An alternative approach to maurer's universal statistical test. *ISCTurkey 2006 Conference Proceedings*, pages 25–27, 2006.

[21] Onur Koçak Ali Doğanaksoy Fatih Sulak, Muhiddin Uğuz. On the independence of statistical randomness tests included in the nist test suite. *Turkish Journal of Electrical Engineering & Computer Sciences*, pages –, 2016. to appear.

[22] Meltem Sönmez Turan, Ali DoĞanaksoy, and Serdar Boztaş. *On Independence and Sensitivity of Statistical Randomness Tests*, pages 18–29. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[23] Muhiddin Uğuz Okan Şeker Ziya Akcengiz Ali Doğanaksoy, Fatih Sulak. Mutual correlation of nist statistical randomness tests and comparison of their sensitivities on transformed sequences. *Turkish Journal of Electrical Engineering & Computer Sciences*, pages –, 2015. to appear.

APPENDIX A

Pearson Correlation Value of Tests

Table A.1: Test naming in the Pearson correlation tables

Autocorrelation	T1	Pseudo Comp.	T11	Run 1	T21
Coverage	T2	Random Ex.	T12	Run 2	T22
Integer Frequency	T3	Random Ex. 1	T13	Run 3	T23
Linear Comp. Pr.	T4	Random Ex. -1	T14	Run 4	T24
Linear Comp.	T5	Random Ex. 2	T15	Run 5	T25
Longest Run	T6	Random Ex. -2	T16	Saturation	T26
Maximum	T7	Random Ex. 3	T17	Template Mat.	T27
Max-Min Diff	T8	Random Ex. -3	T18	Total Runs	T28
Minimum	T9	Random Height	T19	Universal	T29
Pseudo Comp. Pr.	T10	Repetition	T20	Weight	T30

Table A.2: Pearson correlation values between tests - 1

	T1	T2	T3	T4	T5	T6	T7	T8
T1		0,00985	-0,01470	0,01260	-0,00629	-0,01164	-0,00794	-0,00780
T2	0,00985		-0,02078	0,00316	0,00671	-0,01047	0,00260	-0,00147
T3	-0,01470	-0,02078		-0,00037	-0,00097	0,03649	0,01853	0,01074
T4	0,01260	0,00316	-0,00037		0,00811	0,00616	0,01505	0,00712
T5	-0,00629	0,00671	-0,00097	0,00811		0,01645	-0,00646	-0,01392
T6	-0,01164	-0,01047	0,03649	0,00616	0,01645		0,01245	0,02328
T7	-0,00794	0,00260	0,01853	0,01505	-0,00646	0,01245		0,06928
T8	-0,00780	-0,00147	0,01074	0,00712	-0,01392	0,02328	0,06928	
T9	-0,00266	-0,00130	0,04343	-0,00957	0,00439	0,02087	0,00893	0,08128
T10	0,00520	0,01834	0,02629	-0,00252	-0,00451	0,02764	0,00297	-0,00362
T11	-0,01472	0,00371	0,01513	-0,00176	-0,00468	0,00840	-0,00195	-0,00584
T12	0,00835	0,00350	0,00352	0,00775	-0,01747	-0,00007	-0,00521	0,00902
T13	-0,00035	0,02085	0,01434	-0,00695	0,00206	0,00974	0,02076	0,01076
T14	0,00833	0,00010	-0,01311	0,00886	0,00565	0,00904	0,00680	0,00812
T15	0,01236	0,00654	0,03393	-0,00319	-0,01024	0,00985	0,01495	0,00625
T16	0,02002	-0,00064	0,02354	0,02236	-0,00001	0,01613	0,00382	-0,00940
T17	0,01051	-0,00603	0,04049	0,00089	0,00274	-0,00109	0,00727	0,00260
T18	0,00074	-0,00526	0,04515	0,01006	-0,00833	-0,01270	0,01918	-0,00366
T19	0,00393	0,00459	0,07775	0,01680	0,00054	0,01161	0,02290	-0,00453
T20	0,00703	0,01907	0,00310	-0,00977	-0,01546	0,00796	-0,00815	0,00931
T21	0,00939	-0,00316	0,06815	0,01560	-0,01214	-0,00333	0,01908	0,00877
T22	0,01785	0,01421	0,01754	0,00288	0,00618	0,00245	-0,00498	-0,00110
T23	0,00255	-0,00105	0,00919	0,00653	-0,00635	0,01188	-0,01960	0,01095
T24	-0,00257	0,00746	0,00963	0,00141	-0,01162	0,00291	-0,00794	-0,02144
T25	0,00685	0,00565	0,02025	0,01189	-0,01449	0,00708	0,00690	-0,00145
T26	0,00294	0,00164	-0,00070	0,00724	-0,00234	-0,00633	0,00004	0,00472
T27	-0,01338	-0,00612	0,07379	-0,01659	0,00639	0,01235	0,00016	0,01054
T28	-0,01753	-0,00383	0,07379	-0,00104	0,01882	0,02311	0,00841	0,01267
T29	0,00246	-0,00492	-0,00357	0,00319	-0,00228	0,00018	0,00100	-0,00784
T30	0,00143	0,00821	0,10906	0,00660	-0,01105	0,01017	0,01353	-0,01358

Table A.3: Pearson correlation values between tests - 2

	T10	T11	T12	T13	T14	T15	T16
T1	-0,00266	0,00520	-0,01472	0,00835	-0,00035	0,00833	0,01236
T2	-0,00130	0,01834	0,00371	0,00350	0,02085	0,00010	0,00654
T3	0,04343	0,02629	0,01513	0,00352	0,01434	-0,01311	0,03393
T4	-0,00957	-0,00252	-0,00176	0,00775	-0,00695	0,00886	-0,00319
T5	0,00439	-0,00451	-0,00468	-0,01747	0,00206	0,00565	-0,01024
T6	0,02087	0,02764	0,00840	-0,00007	0,00974	0,00904	0,00985
T7	0,00893	0,00297	-0,00195	-0,00521	0,02076	0,00680	0,01495
T8	0,08128	-0,00362	-0,00584	0,00902	0,01076	0,00812	0,00625
T9		0,02396	-0,00052	-0,00278	-0,01429	-0,00219	0,00812
T10	0,02396		-0,01894	0,00043	-0,00736	-0,00501	0,01793
T11	-0,00052	-0,01894		0,00158	0,00200	0,00478	-0,01208
T12	-0,00278	0,00043	0,00158		0,13186	0,14480	0,04600
T13	-0,01429	-0,00736	0,00200	0,13186		0,02657	0,16873
T14	-0,00219	-0,00501	0,00478	0,14480	0,02657		0,01558
T15	0,00812	0,01793	-0,01208	0,04600	0,16873	0,01558	
T16	0,00112	0,01598	-0,00405	0,05321	0,01060	0,17916	0,01366
T17	0,01346	0,03017	-0,00203	0,02239	0,07569	0,00099	0,19857
T18	-0,01622	0,00879	-0,00792	0,03733	0,00722	0,07826	0,01648
T19	0,01774	0,03412	-0,01000	0,05803	0,09380	0,03045	0,18073
T20	0,00283	0,00673	-0,00202	-0,01616	-0,00409	0,01224	0,01111
T21	0,00969	0,00869	-0,01009	-0,02609	-0,01388	-0,00946	-0,00309
T22	0,01170	0,01787	-0,00184	0,00531	0,00663	0,00841	0,00419
T23	0,02317	-0,00234	0,01062	0,00288	0,00070	-0,00447	0,00056
T24	-0,01005	0,00979	0,00081	0,00048	0,01336	-0,00317	-0,01352
T25	-0,00187	-0,01182	0,02333	0,00730	0,00841	0,00329	0,01990
T26	0,00259	0,01508	-0,01715	-0,00056	-0,02406	-0,00643	-0,00399
T27	0,01880	0,03306	0,01150	0,01643	0,00232	0,02018	0,03427
T28	0,00530	0,00761	-0,00712	-0,01840	0,00038	0,00994	0,01356
T29	-0,01200	-0,00049	0,00892	0,00207	0,00558	-0,01484	0,01652
T30	0,03461	0,05660	0,00934	0,03901	0,02513	0,01224	0,05713

Table A.4: Pearson correlation values between tests - 3

	T17	T18	T19	T20	T21	T22	T23
T1	0,02002	0,00074	0,00393	0,00703	0,00939	0,01785	0,00255
T2	-0,00064	-0,00526	0,00459	0,01907	-0,00316	0,01421	-0,00105
T3	0,02354	0,04515	0,07775	0,00310	0,06815	0,01754	0,00919
T4	0,02236	0,01006	0,01680	-0,00977	0,01560	0,00288	0,00653
T5	-0,00001	-0,00833	0,00054	-0,01546	-0,01214	0,00618	-0,00635
T6	0,01613	-0,01270	0,01161	0,00796	-0,00333	0,00245	0,01188
T7	0,00382	0,01918	0,02290	-0,00815	0,01908	-0,00498	-0,01960
T8	-0,00940	-0,00366	-0,00453	0,00931	0,00877	-0,00110	0,01095
T9	0,00112	-0,01622	0,01774	0,00283	0,00969	0,01170	0,02317
T10	0,01598	0,00879	0,03412	0,00673	0,00869	0,01787	-0,00234
T11	-0,00405	-0,00792	-0,01000	-0,00202	-0,01009	-0,00184	0,01062
T12	0,05321	0,03733	0,05803	-0,01616	-0,02609	0,00531	0,00288
T13	0,01060	0,00722	0,09380	-0,00409	-0,01388	0,00663	0,00070
T14	0,17916	0,07826	0,03045	0,01224	-0,00946	0,00841	-0,00447
T15	0,01366	0,01648	0,18073	0,01111	-0,00309	0,00419	0,00056
T16		0,20876	0,04374	-0,00048	-0,01708	0,01601	0,00428
T17	0,02939	0,01865	0,14469	0,00725	-0,01351	0,00687	-0,01726
T18	0,20876		0,06301	-0,01106	0,00739	0,00862	-0,01546
T19	0,04374	0,06301		0,01268	-0,02244	0,01950	-0,01198
T20	-0,00048	-0,01106	0,01268		-0,01155	0,01434	0,01390
T21	-0,01708	0,00739	-0,02244	-0,01155		-0,00512	0,00934
T22	0,01601	0,00862	0,01950	0,01434	-0,00512		0,01628
T23	0,00428	-0,01546	-0,01198	0,01390	0,00934	0,01628	
T24	0,01231	-0,00879	0,01046	-0,01126	0,00672	0,00439	0,00553
T25	0,01060	0,01236	0,00632	-0,00078	0,00329	-0,00178	0,01442
T26	-0,00779	-0,00529	-0,00601	-0,00064	0,00693	0,02130	0,00217
T27	0,02412	0,02352	0,05967	0,00071	0,01699	0,03403	0,02075
T28	0,00599	0,01703	-0,00381	0,01571	0,20141	-0,00045	0,00306
T29	0,00557	0,01062	0,00375	0,01499	-0,00102	0,00463	0,00707
T30	0,03878	0,06737	0,17660	0,01123	-0,01660	0,00230	-0,00910

Table A.5: Pearson correlation values between tests - 4

	T24	T25	T26	T27	T28	T29	T30
T1	-0,00257	0,00685	0,00294	-0,01338	-0,01753	0,00246	0,00143
T2	0,00746	0,00565	0,00164	-0,00612	-0,00383	-0,00492	0,00821
T3	0,00963	0,02025	-0,00070	0,07379	0,07379	-0,00357	0,10906
T4	0,00141	0,01189	0,00724	-0,01659	-0,00104	0,00319	0,00660
T5	-0,01162	-0,01449	-0,00234	0,00639	0,01882	-0,00228	-0,01105
T6	0,00291	0,00708	-0,00633	0,01235	0,02311	0,00018	0,01017
T7	-0,00794	0,00690	0,00004	0,00016	0,00841	0,00100	0,01353
T8	-0,02144	-0,00145	0,00472	0,01054	0,01267	-0,00784	-0,01358
T9	-0,01005	-0,00187	0,00259	0,01880	0,00530	-0,01200	0,03461
T10	0,00979	-0,01182	0,01508	0,03306	0,00761	-0,00049	0,05660
T11	0,00081	0,02333	-0,01715	0,01150	-0,00712	0,00892	0,00934
T12	0,00048	0,00730	-0,00056	0,01643	-0,01840	0,00207	0,03901
T13	0,01336	0,00841	-0,02406	0,00232	0,00038	0,00558	0,02513
T14	-0,00317	0,00329	-0,00643	0,02018	0,00994	-0,01484	0,01224
T15	-0,01352	0,01990	-0,00399	0,03427	0,01356	0,01652	0,05713
T16	0,01231	0,01060	-0,00779	0,02412	0,00599	0,00557	0,03878
T17	-0,00458	0,00532	-0,02637	0,02518	-0,00085	-0,00602	0,06962
T18	-0,00879	0,01236	-0,00529	0,02352	0,01703	0,01062	0,06737
T19	0,01046	0,00632	-0,00601	0,05967	-0,00381	0,00375	0,17660
T20	-0,01126	-0,00078	-0,00064	0,00071	0,01571	0,01499	0,01123
T21	0,00672	0,00329	0,00693	0,01699	0,20141	-0,00102	-0,01660
T22	0,00439	-0,00178	0,02130	0,03403	-0,00045	0,00463	0,00230
T23	0,00553	0,01442	0,00217	0,02075	0,00306	0,00707	-0,00910
T24		0,02724	0,00509	-0,01464	0,01087	0,00873	0,00689
T25	0,02724		0,00595	-0,00053	0,00479	0,01589	0,02054
T26	0,00509	0,00595		0,00332	-0,01159	-0,00673	-0,00479
T27	-0,01464	-0,00053	0,00332		0,02347	0,00983	0,09429
T28	0,01087	0,00479	-0,01159	0,02347		-0,00547	0,01196
T29	0,00873	0,01589	-0,00673	0,00983	-0,00547		0,00047
T30	0,00689	0,02054	-0,00479	0,09429	0,01196	0,00047	

APPENDIX B

Fail-Fail Ratio of the Tests

Table B.1: *Fail-Fail Ratio* of the tests with correlation value $c > 0.4$

Test-1	Test-2	Correlation Ratio
Minimum	Pseudo Complexity	1
Random Height	Random Excursion_y=1	1
Random Excursion_y=1	Random Height	1
Random Excursion_y=2	Random Height	1
Random Excursion_y=2	Random Excursion_y=1	1
Random Excursion_y=3	Random Height	1
Random Excursion_y=3	Random Excursion_y=1	1
Random Excursion_y=3	Random Excursion_y=2	1
Random Excursion_y=-2	Random Excursion_y=-1	1
Random Excursion_y=-3	Random Excursion_y=-1	1
Random Excursion_y=-3	Random Excursion_y=-2	1
Run>5	Longest Run	0,971291866
Run-5	Longest Run	0,968778696
Random Excursion_y=-3	Weight	0,928571429
Random Excursion_y=-2	Weight	0,875
Random Excursion_y=-3	Pseudo Complexity Profile	0,862068966
Random Excursion_y=-2	Pseudo Complexity Profile	0,857142857
Random Excursion_y=-2	Maximum	0,785714286
Random Excursion_y=-3	Maximum	0,785714286
Random Excursion_y=2	Random Excursion_y=3	0,673299531
Random Excursion_y=-2	Random Excursion_y=-3	0,672817907
Minimum	Pseudo Complexity Profile	0,6
Minimum	Weight	0,571428571
Random Excursion_y=-1	Weight	0,541666667
Random Excursion	Random Excursion_y=-1	0,515960452
Random Height	Random Excursion_y=2	0,511131994
Random Excursion_y=1	Random Excursion_y=2	0,511131994
Random Excursion_y=-1	Random Excursion	0,509412913
Random Excursion_y=-2	Random Excursion	0,509412913

Continued on next page

Table B.1 – *Continued from previous page*

Test-1	**Test-2**	**Correlation Ratio**
Random Excursion_y=-3	Random Excursion	0,509412913
Random Excursion_y=-1	Random Excursion_y=-2	0,508390706
Random Excursion	Random Height	0,507426099
Random Excursion	Random Excursion_y=1	0,507426099
Random Height	Random Excursion	0,504371506
Random Excursion_y=1	Random Excursion	0,504371506
Random Excursion_y=2	Random Excursion	0,504371506
Random Excursion_y=3	Random Excursion	0,504371506
Minimum	Total Runs	0,470588235
Random Excursion_y=3	Linear Complexity	0,4375
Run>5	Max-Min Diff	0,416666667
Maximum	Max-Min Diff	0,416666667
Random Excursion	Weight	0,416666667
Random Excursion_y=-2	Max-Min Diff	0,416666667
Random Excursion_y=-3	Max-Min Diff	0,416666667

APPENDIX C

Fail-Fail Ratio of NIST Tests Suite

Table C.1: NIST Short Sequences Fail-Fail Ratio Table

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
Frequency (T_1)		**0.070122**	0.00813	0.006098	0.010163	0.011179	0.009146	0.010163	0.011179
Block Frequency(T_2)	**0.072025**		0.004175	0.011482	0.008351	0.012526	0.016701	0.009395	0.005219
Runs(T_3)	0.007663	0.003831		0.017241	0.011494	0.025862	0.020115	0.023946	0.006705
Longest Run(T_4)	0.005848	0.010721	0.017544		0.009747	0.016569	0.035088	0.019493	0.008772
Binary Matrix Rank(T_5)	0.012034	0.009627	0.014440	0.012034		0.014440	0.009627	0.015644	0.012034
Approximate Entropy(T_6)	0.009821	0.010714	0.024107	0.015179	0.010714		0.022321	**0.306250**	**0.137500**
Cumulative Sum(T_7)	0.008964	0.015936	0.020916	0.035857	0.007968	0.024900		0.029880	0.004980
Serial-1 (T_8)	0.009398	0.008459	0.023496	0.018797	0.012218	**0.322368**	0.028195		**0.281955**
Serial-2 (T_9)	0.011168	0.005076	0.007107	0.009137	0.010152	**0.156345**	0.005076	**0.304569**	

APPENDIX D

Correlations of the Tests with Corresponding to Transformed Sequences

Table D.1: Correlation of tests with respect to the transformations

Test Name	$C(\sigma)$	$B(\sigma)$	$HR(\sigma)$	$C(R(\sigma))$	Reverse	$H(\sigma)$	$RH(\sigma)$
Autocorrelation	1,000000	0,616492	0,059230	0,224906	0,224906	0,005444	0,241093
Coverage	1,000000	0,204551	0,175537	0,245642	0,245642	0,183341	0,162245
Integer Frequency	0,143314	0,004453	0,138369	0,284798	0,558945	0,279168	0,055682
Linear Complexity	0,683198	0,000978	0,000282	0,028148	0,046480	0,005047	0,001701
Linear Complexity Profile	0,314211	0,051909	0,004756	0,006846	0,006028	0,005804	0,028192
Longest Run	1,000000	0,831832	0,931950	0,979056	0,979056	0,919052	0,917548
Maximum	0,062450	0,856820	0,381123	0,037794	0,057852	1,000000	0,057852
Max-Min Diff	1,000000	0,848187	0,358943	0,027538	0,027538	1,000000	0,027538
Minimum	0,062450	0,857644	0,389347	0,033628	0,065103	1,000000	0,065103
Pseudo Complexity	0,024608	0,673146	0,002345	0,002840	0,002293	0,003891	0,002937
Pseudo Complexity Profile	0,703245	0,796995	0,918352	0,678248	0,866998	0,974521	0,867314
Template Matching	-0,326734	0,445557	0,977526	-0,326734	1,000000	1,000000	1,000000
Random Excursion	1,000000	1,000000	0,130502	0,103905	0,103905	0,148510	0,364410
Random Excursion(1)	0,460839	0,895269	0,121609	0,077389	0,097195	0,142718	0,352743
Random Excursion(-1)	0,460839	0,895284	0,120362	0,080167	0,096781	0,143221	0,357816
Random Excursion(2)	0,007184	1,000000	0,135515	0,053653	0,113766	0,157979	0,375711
Random Excursion(-2)	0,007184	1,000000	0,130510	0,044295	0,111067	0,157406	0,376769
Random Excursion(3)	0,279692	0,912234	0,135703	0,016603	0,112201	0,158821	0,387904
Random Excursion(-3)	0,279692	0,911844	0,134499	0,003498	0,114029	0,160278	0,386232
Random Height	0,690247	0,996327	0,711939	0,376123	0,685753	0,746442	0,865111
Repetition	1,000000	1,000000	0,012933	0,019101	0,019101	0,022164	0,129598
Run>5	1,000000	0,703733	0,968436	1,000000	1,000000	0,962647	0,962647
Run-1	1,000000	0,611412	0,989284	1,000000	1,000000	0,987657	0,987657
Run-2	1,000000	0,564285	0,976437	1,000000	1,000000	0,969664	0,969664
Run-3	1,000000	0,361314	0,966900	1,000000	1,000000	0,958981	0,958981
Run-4	1,000000	0,020842	0,960177	1,000000	1,000000	0,952106	0,952106
Run-5	1,000000	0,420643	0,953486	1,000000	1,000000	0,945842	0,945842
Saturation	1,000000	0,057627	0,062293	0,092427	0,092427	0,050854	0,108281
Number of Total Runs	1,000000	0,505481	0,992205	1,000000	1,000000	0,992167	0,992167
Universal	1,000000	0,481727	0,540447	0,571712	0,571712	0,543717	0,731666
Weight	1,000000	1,000000	1,000000	1,000000	1,000000	1,000000	1,000000

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Koçak, Onur
Nationality: Turkish
Date and Place of Birth: 1985, Ankara
Marital Status: Married

EDUCATION

Degree	Institution	Year of Graduation
M.S.	Department of Cryptography, METU	2009
B.S.	Department of Mathematics, METU	2007
High School	Süleyman Demirel Anatolian High School	2002

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
06.2015-	TUBITAK UEKAE	Researcher
06.2012-06.2015	TURKTRUST Inc.	Project Engineer

PUBLICATIONS

F. Sulak, O. Koçak, E. Saygı, M. Öğünç, B. Bozdemir, *A second pre-image attack and a collision attack to cryptographic hash function LUX*, Communications, Faculty of Sciences University of Ankara

F. Sulak, M. Uğuz, O. Koçak, A. Doğanaksoy, *On the independence of statistical randomness tests included in the NIST test suite*, Turkish Journal of Electrical Engineering & Computer Sciences, to appear

N. Koçak, O. Koçak, F. Özbudak, Z.Saygı, *Characterisation and Enumeration of a Class of Semi-Bent Quadratic Boolean Functions*, Int. J. Information and Coding Theory, vol.3, no 2, 39-57, 2015.

A. Doğanaksoy, B. Ege, O. Koçak, F. Sulak, *Statistical Analysis of Reduced Round Compression Functions of SHA-3 Second Round Candidates*, Int. J. Research and Reviews in Applied Sciences, April 2013

O. Koçak, O. Kurt, N. Öztop, Z.Saygı, *Notes on Bent Functions in Polynomial Forms*, Int. J. Information Security Science, vol.1, no 2, 43-48, 2012.

O. Koçak, N. Öztop, *Cryptanalysis of TWIS Block Cipher*, WEWORC 2011, LNCS 7242, 109-121. Springer, Heidelberg, 2012.

F. Sulak, A. Doğanaksoy, B. Ege, O. Koçak, *Evaluation of Randomness Test Results for Short Sequences*, Sequences and Their Applications - SETA 2010 Lecture Notes in Computer Science, 2010, Volume 6338/2010, 309-319