STATISTICAL ANALYSIS OF BLOCK CIPHERS AND HASH FUNCTIONS

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

BY

FATİH SULAK

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

FEBRUARY 2011

Approval of the thesis:

STATISTICAL ANALYSIS OF BLOCK CIPHERS AND HASH FUNCTIONS

submitted by **FATİH SULAK** in partial fulfillment of the requirements for the degree of **Philosophy of Doctorate in Department of Cryptography, Middle East Technical University** by,

Prof. Dr. Ersan Akyıldız _____
Director, Graduate School of **Applied Mathematics**

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

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

Examining Committee Members:

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

Prof. Dr. Ferruh Özbudak _____
Department of Mathematics, METU

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

Assis. Prof. Dr. Ali Aydın Selçuk _____
Department of Computer Engineering, Bilkent University

Assis. Prof. Dr. Şahin Emrah _____
Department of Computer Engineering, Ankara 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: FATİH SULAK

Signature :

ABSTRACT

STATISTICAL ANALYSIS OF BLOCK CIPHERS AND HASH FUNCTIONS

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

February 2011, 64 pages

One of the most basic properties expected from block ciphers and hash functions is passing statistical randomness testing, as they are supposed to behave like random mappings. Previously, testing of AES candidate block ciphers was done by using the statistical tests defined in the NIST Test Suite. As some of the tests in this suite require long sequences, data sets are formed by concatenating the outputs of the algorithms obtained from various input types. However, the nature of block cipher and hash function algorithms necessitates devising tests and test parameters focused particularly on short sequences, therefore we propose a package of statistical randomness tests which produce reliable results for short sequences and test the outputs of the algorithms directly rather than concatenations. Moreover, we propose an alternative method to evaluate the test results and state the required computations of related probabilities for the new evaluation method.

We also propose another package of statistical tests which are designed basing on certain cryptographic properties of block ciphers and hash functions to evaluate their randomness, namely the cryptographic randomness testing. The packages are applied to the AES finalists, and produced more precise results than those obtained in similar applications. Moreover, the

packages are also applied to SHA-3 second round candidate algorithms.

Keywords: Statistical Randomness Tests, Cryptographic Randomness Tests, Block Ciphers, Hash Functions, AES Finalists

ÖZ

BLOK TİPİ ALGORİTMALARIN VE ÖZET FONKSİYONLARIN İSTATİSTİKSEL ANALİZİ

Sulak, Fatih
Doktora, Kriptografi Bölümü
Tez Yöneticisi : Ali Doğanaksoy

Şubat 2011, 64 sayfa

Blok tipi algoritmaların ve özet fonksiyonların rastgele fonksiyonlar gibi davranmaları beklendiği için bu algoritmaların sağlaması gereken temel özelliklerden birisi de istatistiksel rassallık testlerinden geçmeleridir. AES yarışmasında aday olan blok tipi algoritmaların istatistiksel analizi NIST test paketi kullanılarak yapılmıştı. Fakat bu paketteki bazı testler uzun dizilere uygulanabildiği için algoritmaların çıktıları arka arkaya eklenerek uzun diziler oluşturulmuştu. Ancak blok tipi algoritmaların ve özet fonksiyonların doğası gereği kısa dizilere uygun testlerden oluşan bir paket daha sağlıklı sonuçlar vereceği için, bu çalışmada kısa diziler için sağlıklı sonuçlar üreten testlerden oluşan yeni bir istatistiksel test paketi önerdik ve bu paketi algoritmaların çıktılarına doğrudan uyguladık. Ayrıca yeni bir değerlendirme metodu önerdik ve bu metod için gerekli olasılık hesaplamalarını yaptık.

Bunun dışında blok tipi algoritmaların ve özet fonksiyonların belirli kriptografik özelliklerine dayanan yeni bir test paketi, kriptografik test paketi, önerdik. Bu paketleri AES yarışmasında finale kalan algoritmalara uyguladık ve önceki çalışmalardan daha iyi sonuçlar elde ettik. Ayrıca paketleri SHA-3 yarışmasında ikinci aşamaya kalan algoritmalara da uyguladık.

Anahtar Kelimeler: İstatistiksel rassallık testleri, Kriptografik rassallık testleri, Blok Tipi Algoritmalar, Özet Fonksiyonlar, AES finalistleri

ACKNOWLEDGMENTS

First of all, I owe my sincerest gratitude to my supervisor Ali Doğanaksoy who patiently guides, motivates, and encourages me throughout not only this thesis but also all my academic studies. Without his patience and guidance, it would be impossible to finish this work.

I am heartily thankful to Onur Koçak and Barış Ege for being so cooperative during the studies we have done together. I am very lucky to have colleagues and friends like them.

I thank all the members of the SAKDAT group at IAM, especially to Çağdaş Çalık and Meltem Sönmez Turan for their valuable comments. I also wish to thank all the colleagues and friends I have met over many years at IAM, that I could not mention individually.

Special thanks should be given to Köksal Muş for his support, and most importantly for his friendship.

Last but not the least, I would like to thank my family for their love and support during my whole life.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1 Random Sequences

A sequence whose terms are chosen from a finite set in such a manner that, each term takes any value from the set with an equal probability, is said to be random. If the set is $\{0, 1\}$, then the sequence is called a binary random sequence, or in other words, $\{a_n\}$ is called a binary random sequence if for any i, $Pr(a_i = 0) = Pr(a_i = 1) = 1/2$.

Binary random sequences have the following properties [1]:

- Independence: The terms (bits) of a binary random sequence are independent, that is for any $i \neq j$, $Pr(a_i = a_j) = 1/2$.
- Unpredictability: If t bits of the sequence are known, it is impossible to predict another bit.
- Uniformity: Zeroes and ones are uniformly distributed among the entire sequence.

An integer which is taken from the interval $[0, n-1]$ with every integer having probability $1/n$ to be chosen is called a random number. Binary random sequences can be used to generate random numbers.

Random sequences and random numbers are used in a large variety of areas, such as quantum mechanics, game theory, statistics, cryptography, and so on. In cryptography, random sequences are needed for several applications, such as the generation of primes in RSA encryption, secret keys in symmetric encryption, challenges in challenge-response protocols,

initialization vectors, salts in hash functions and the like, but the most common application is the generation of secret keys.

Secret keys should be generated randomly so that the best option of the attacker should not be better than trying all possible elements in the set from which the key is chosen. If an attacker narrows down the number of possible keys, then the protocol is assumed to be broken. In 1996, Goldberg and Wagner [2] showed that the "random numbers" used to generate the keys in Netscape SSL protocol were based on the time of the processor and therefore predictable, which helped them to find a major weakness in the protocol. Thus, it is vital to use an algorithm which produces random numbers properly.

1.2 Random Number Generators

Generation of random numbers for cryptographic purposes is a difficult task. Random numbers are ideally generated using true random sources, called true random number generators (TRNGs), which use a nondeterministic source to produce random numbers. One method to produce random numbers is measuring the elapsed time between emission of particles, as the radioactive decay time is unpredictable. The other methods also typically consist of some physical quantity, such as atmospheric noise, thermal noise from a semiconductor diode, sound from a microphone, noise in an electrical circuit and so on.

On the one hand generation of random numbers using TRNGs is inefficient and on the other hand it is impractical to store and transmit large number of random bits. Therefore, deterministic algorithms, which are called pseudorandom number generators (PRNGs), are preferred to TRNGs. PRNGs take a truly random binary sequence (seed) of length k and produce a periodic "random looking" binary sequence of length $l >> k$ [3]. In most applications, the seed is produced by TRNGs. The length of the seed should be sufficiently large and the output should be unpredictable to an adversary with a limited computational power.

The characteristics of PRNGs are different from TRNGs. First, PRNGs are efficient compared to TRNGs, taking shorter time to produce numbers. They are also deterministic, meaning that a given sequence of numbers is reproducible. PRNGs are periodic while TRNGs have no period.

An example of a PRNG is the linear congruential generator [3] which produces a pseudorandom sequence of numbers $x_1, x_2, x_3, \ldots$ according to the linear recurrence

$$x_n = a \cdot x_{n-1} + b \ (mod\ m)$$

where x_0 is the seed and a, b, and m are parameters of the generator.

Another method to generate pseudorandom sequences is to utilize one way functions by selecting a random seed s, and applying the function to the sequence $s, s+1, s+2, \ldots$ thus obtaining the pseudorandom sequence $f(s), f(s+1), f(s+2), \ldots$.

1.3 Statistical Randomness Tests

The output sequences of PRNGs should be statistically indistinguishable from truly random sequences, therefore statistical analysis of PRNGs are crucial. Analysis of PRNGs is performed by producing a sample sequence, and evaluating this sequence by statistical randomness tests.

A statistical randomness test is developed to test a null hypothesis (H_0) which states the input sequence is random. The test takes a binary sequence as an input and "accepts" or "rejects" the hypothesis. Randomness tests are probabilistic and there are two types of errors. If the data is random and H_0 is rejected type I error is occured and if the data is nonrandom and H_0 is accepted type II error is occured. The probability of a type I error is called the level of significance of the test and denoted by α. A statistical test produces a real number between 0 and 1 which is called p-value. If $p - value > \alpha$ then H_0 is accepted, otherwise rejected. Therefore, level of significance varies according to application, however for cryptographic applications it is usually set to 0,01.

One of the first attempts to measure randomness of binary sequences is the postulates proposed by Golomb, which state three necessary conditions for randomness of a periodic binary sequence. Let $a_1, a_2, a_3, \ldots$ be a periodic binary sequence with period N, and let a "run" be an uninterrupted maximal sequence of identical bits. Then for the subsequence $a_1, a_2, \ldots, a_N$:

1. The number of ones differs from the number of zeroes by at most 1.

2. At least $1/2^n$ of the runs have length n, and for each length, the runs of zeroes and the

runs of ones are equal.

3. The autocorrelation function is two-valued.

The outputs of linear feedback shift registers satisfy all three postulates, therefore the postulates are not sufficient to measure randomness. Golomb's postulates are used to define randomness tests. The first postulate states that the weight of an n-bit sequence is approximately $n/2$ and Frequency Test takes n-bit binary sequence as input, calculates the weight of the sequence and using normal distribution produces a p-value. Similarly some other randomness tests are defined using the second and the third postulates.

1.4 Test Suites

A test suite is a collection of statistical randomness test that are designed to test the randomness properties of sequences. There are several test suites in the literature:

- The first collection of randomness tests were presented by Knuth in his famous book [4]. The randomness tests mentioned in this book are Frequency Test, Serial Test, Gap Test, Poker Test, Coupon Collector's Test, Permutation Test, Run Test, Maximum of t Test, Collision Test, Birthday Spacings Test, and Serial Correlation Test.

- CRYPT-X [5], a test suite developed in Queensland University of Technology, consists of Frequency Test, Binary Derivative Test, Change Point Test, Runs Test, Sequence Complexity Test, Linear Complexity Test, Sub-blocks Test, and Entropy Test.

- DIEHARD Test Suite [6] was developed by Marsaglia and published in 1995 on a CD-ROM. The tests in this suite are Birthday Spacings Test, The Overlapping 5 Permutation Test, Binary Matrix Rank Test, Bitstream Test, Overlapping Pairs Sparse Occupancy Test, Overlapping Quadruples Sparse Occupancy Test, DNA Test, Count the Ones Test, Parking Lot Test, Minimum Distance Test, 3D Spheres Test, Squeeze Test, The Overlapping Sums Test, Run Test, and Craps Test. The randomness tests in the suite are designed to evaluate 32-bit integers.

- NIST Test Suite [7] (published in 2001) originally consisted of 16 tests but the Fast Fourier Transform Test was later discarded due to a problem discovered by NIST in

2009. The randomness tests in the suite are Frequency Test, Frequency Test within a Block, Runs Test, Test for the Longest Run of Ones in a Block, Binary Matrix Rank Test, Discrete Fourier Transform Test, Non-overlapping Template Matching Test, Overlapping Template Matching Test, Maurer's Universal Statistical Test, Lempel-Ziv Compression Test, Linear Complexity Test, Approximate Entropy Test, Cumulative Sums Test, Random Excursions Test, and Random Excursions Variant Test. This suite is used by Soto and Bassham [8] to evaluate the AES finalists block ciphers.

- TESTU01 [9] is a recently designed test suite, which has two categories: Those that apply to a sequence of real numbers in (0, 1) and those designed for a sequence of bits.

1.5 Motivation of the Thesis

The test suites in the literature are designed to evaluate the randomness of PRNGs and sequences. Our motivation is to test block ciphers and hash functions which produce short sequences. To attain this, we determine the statistical randomness tests which produce reliable results for short sequences and propose a new evaluation method.

The outline of the thesis is as follows: In chapter 2, we give the definitions of the statistical randomness tests, which produce reliable results for short sequences, with the required computations of related probabilities for the new evaluation method. We also present a method to classify these statistical randomness tests. In chapter 3, we propose a package of cryptographic randomness tests, designed based on certain cryptographic properties of block ciphers and hash functions to evaluate their randomness. In chapter 4, we apply the packages defined in chapter 2 and chapter 3 to the finalist algorithms in AES competition and the second round candidate algorithms in SHA-3 competition. To reach this goal, first we define how to produce data sets from the algorithms for statistical randomness testing, then we give the number of rounds where the algorithms achieve randomness. Finally we conclude the thesis in chapter 5.

CHAPTER 2

STATISTICAL RANDOMNESS TESTING

The outputs of cryptographic primitives such as block ciphers and hash functions should be random looking so that when the outputs are analyzed, predicting the algorithm should not be possible. Such an algorithm is said to be indistinguishable from a random permutation (or a random mapping). Therefore, the evaluation of the outputs of the algorithms by statistical randomness tests is of great importance. This process consists of two parts: taking a sample sequence from the algorithm, and analyzing this sample by statistical randomness tests.

During Advanced Encryption Standard (AES) competition, statistical testing of the candidate block ciphers is done by J. Soto[8], using the statistical tests defined in the NIST Test Suite [7]. However, as approximations and asymptotic approaches used in the distribution functions of statistical randomness tests force the user to use long sequences, some of the tests in the suite require sequences of length at least 10^6 bits for testing, where the block size of the AES candidates were 128 bits. Soto proposed a solution to this problem by forming data sets as concatenation of the outputs of the candidate algorithms, which correspond to various input types like key avalanche, high weight, low weight and the like.

However, the nature of block cipher and hash function algorithms necessitates devising tests and test parameters focused particularly on sequences of lengths between 128 bits and 512 bits, so called "*short sequences*". Our aim is to test block ciphers and hash functions which produce short sequences, by testing the outputs directly instead of concatenation. To attain this, we determine the statistical randomness tests, which produce reliable results for short sequences and propose an alternative evaluation method.

2.1 Statistical Randomness Testing

In this section, first we give the definition of the Frequency Test, a basic test for binary sequences, and the application of this test to block ciphers and hash functions with an alternative method for evaluation. Then, we give the definitions of the statistical randomness tests, which produce reliable results for short sequences, with the required computations of related probabilities for the new evaluation method. The probabilities are presented in Appendix A. In all the tests it is assumed that the sequence subject to the test is an n-bit sequence.

2.1.1 Frequency Test

Frequency Test [7] compares the Hamming weight W of the sequence with the expected weight of a random sequence. For a given weight w, there are $\binom{n}{w}$ many n-bit sequences, and $Pr(W = w) = \dfrac{\binom{n}{w}}{2^n}$, therefore the probability distribution function of this test is the binomial distribution.

When $n > 30$, the binomial distribution can be approximated by the normal distribution function. If we use the normal distribution function to produce p-value, we assign different p-values to the sequences with $W = n - t$ and $W = n + t$. However, for cryptographic purposes these sequences are identical in terms of randomness. Therefore, we use the two tailed distribution function instead of the standard normal distribution to produce p-value.

In order to test block ciphers and hash functions, a set of n-bit sequences are evaluated using randomness tests, and a set of p-values are obtained. Two approaches are suggested in the NIST Test Suite [7] to evaluate test results: examination of the proportion of sequences with a p-value greater than a certain bound; and the distribution of p-values. The latter one assumes that p-values are uniformly distributed. A goodness of fit distribution test is performed to measure whether the p-values are uniform or not by dividing the interval $[0, 1]$ into 10 equal subintervals.

Let m be the number of sequences tested, and F_i be the number of p-values in subinterval i for $i = 1, 2, \ldots, 10$. Then the χ^2 value and the corresponding p-value are calculated as

$$\chi^2 = \sum_{i=1}^{10} \frac{(F_i - \frac{m}{10})^2}{\frac{m}{10}} \quad \text{and} \quad p\text{-value} = \texttt{igamc}\left(\frac{9}{2}, \frac{\chi^2}{2}\right)$$

where igamc is the incomplete gamma function. If p-value$\geq$0.0001, the test results are considered to be uniformly distributed.

However, when short sequences are in question, two main problems arise. First problem is noted in [7], *"the asymptotic reference distributions would be inappropriate and would need to be replaced by exact distributions that would commonly be difficult to compute"*. The second problem arises from the fact that, for short sequences, the probability of a p-value to be in a subinterval is not the same for all subintervals, since the exact distribution is discrete where the asymptotic reference distribution is continuous.

In our approach, we choose the difficult way and compute the probability of a p-value to be in a given subinterval, for each test. NIST assumes that this probability is $\frac{1}{10}$ for 10 subintervals but this is not the case for short sequences. Hence, we compute the subinterval probabilities for each test.

By making a table consisting of all possible test variables and the corresponding p-values, together with the probabilities, we find the distribution of p-values. Thus, for each $i = 0, 1, \ldots, 9$, we need to compute

$$Pr\left(\frac{i}{10} \leq p\text{-value} < \frac{i+1}{10}\right)^{1}.$$

Using these probabilities for subintervals we propose an improved method for evaluation of the test results. Let p_i be the probability of a p-value to be in subinterval i, then

$$\chi^2 = \sum_{i=1}^{10} \frac{(F_i - m \cdot p_i)^2}{m \cdot p_i} \quad \text{and} \quad p\text{-value} = \texttt{igamc}\left(\frac{9}{2}, \frac{\chi^2}{2}\right).$$

For specific sequence lengths, p_i might be zero for some i. In that case, degree of freedom should be modified accordingly.

For the Frequency Test, the p-value of the sequence depends only on the weight W of the sequence as a variable. As for a given weight w, there are $\binom{n}{w}$ many n-bit sequences, the probability $Pr(W = w)$ is

$$Pr(W = w) = \frac{\binom{n}{w}}{2^n}.$$

For example, for $n = 256$, the p-value of a sequence is in the interval $[0.8, 0.9)$ if and

[1] In the last subinterval we need to compute $Pr(0,9 \leq p\text{-value} \leq 1\ )$

only if the weight of the sequence is 126 or 130. In that case $Pr(0.8 \leq p\text{-value} < 0.9) = \frac{\binom{256}{126} + \binom{256}{130}}{2^{256}} \approx 0.0966$.

2.1.2 Frequency Test within a Block

Frequency Test within a Block [7] separates the sequence into m-bit blocks and compares the proportion of 'one's in each block with the expected values for a random sequence. The variables used for computing p-values are the weights W_i of each block. Then

$$Pr(W_i = w_i) = \frac{\prod_{i=1}^{\lfloor \frac{n}{m} \rfloor} \binom{m}{w_i}}{2^n}. \tag{2.1}$$

We choose $m = 32$ and calculate the corresponding subinterval probabilities. However, as there are 8 subsequences, it is not feasible to use the equation 2.1. But the variable subject to χ^2 is an integer and the complexity of equation 2.1 is reduced by partitioning that integer as sums of perfect squares.

2.1.3 Periodic Frequency Test

Periodic Frequency Test, a similar test to the Frequency Test within a Block, separates the sequence into m-bit blocks and compares the weights of the subsequences $\{a_{n_i}\}$ formed by taking every i^{th} bits of each block with the expected weights of random sequences. The variables used for computing p-values are the weights W_i of each subsequence. We choose $m = 8$, and the calculation of the corresponding subinterval probabilities is given in equation 2.1.

Algorithm 2.1.1: PERIODIC FREQUENCY TEST($\{a_1, a_2, \ldots, a_n\}, k$)

for $i \leftarrow 1$ **to** N
  do
$\{b_i = 1 - 2a_i;$
for $j \leftarrow 1$ **to** $\left\lfloor \frac{N}{B} \right\rfloor$
  do
$\begin{cases} e_j = 0; \\ \textbf{for } i \leftarrow 1 \textbf{ to } B \\ \quad \textbf{do} \\ \quad \{s_i = \sum_{k=1}^{i} b_{(j-1)B+k}; \\ \textbf{for } i \leftarrow 1 \textbf{ to } B \\ \quad \textbf{do} \\ \quad \begin{cases} \textbf{if } \ s_i = 0 \\ \quad \textbf{then } \ e_j = e_j + 1; \end{cases} \end{cases}$
Apply χ^2 *of Goodness of Fit test to* e_j *values*;
return $(p - value)$

2.1.4 Runs Test

A run is an uninterrupted maximal sequence of identical bits. In the Runs Test [7], the number of runs in the sequence is compared with the expected number of runs in a random sequence. The p-value is determined by the variables W and V, which denote the weight of the sequence and the number of runs in the sequence respectively. Now we need to compute $Pr(W = w_1, V = v_1)$ for a given sequence, and calculate the probabilities of subintervals. We consider the question in two cases:

1. $v_1 = 2a$: Since there is an even number of runs, the number of runs of 'zero's and 'one's are equal to each other. First we write 'one's and 'zero's consecutively to define $2a$ runs. Now, we find the distribution of $(w_1 - a)$ many 'one's and $(n - w_1 - a)$ many 'zero's so that the number of runs remains the same. The number of such distributions

is equal to the number of nonnegative integer solutions of the system

$$\begin{aligned} x_1 + x_2 + \cdots + x_a &= w_1 - a \\ y_1 + y_2 + \cdots + y_a &= n - w_1 - a. \end{aligned}$$

The first bit can be zero or one, then the probability is computed as;

$$Pr(W = w_1, V = 2a) = \frac{2\binom{w_1-1}{a-1}\binom{n-w_1-1}{a-1}}{2^n}.$$

2. $v_1 = 2a + 1$: If $v_1 = 1$, the only possibilities are all one and all zero sequences. Hence the probability for this case is $Pr(W = w_1, V = 1) = \frac{2}{2^n}$. Now, assume that $a > 0$. Then, considering the first bit, the problem can be handled in two parts using the previous method. If the first bit is one, there are $(a+1)$ runs of 'one's and a runs of 'zero's; if the first bit is zero, there are a runs of 'one's and $(a+1)$ runs of 'zero's. The number of such distributions is equal to the number of nonnegative integer solutions of the systems

$$\begin{aligned} x_1 + x_2 + \cdots + x_{a+1} &= w_1 - a - 1 \\ y_1 + y_2 + \cdots + y_a &= n - w_1 - a \end{aligned}$$

$$\begin{aligned} x_1 + x_2 + \cdots + x_a &= w_1 - a \\ y_1 + y_2 + \cdots + y_{a+1} &= n - w_1 - a - 1. \end{aligned}$$

Therefore, the probability is computed as;

$$Pr(W = w_1, V = 2a + 1) = \begin{cases} \dfrac{\binom{w_1-1}{a}\binom{n-w_1-1}{a-1} + \binom{w_1-1}{a-1}\binom{n-w_1-1}{a}}{2^n} & \text{if } a \neq 0 \\ \dfrac{2}{2^n} & \text{if } a = 0. \end{cases}$$

As stated in NIST test suite, if $\left|\frac{W}{n} - \frac{1}{2}\right| \geq \frac{2}{\sqrt{n}}$, the p-value is set to 0. Hence, the subinterval probabilities are calculated regarding this.

2.1.5 Test for the Longest Run of Ones in a Block

Test for the Longest Run of Ones in a Block [7] separates the sequence into m-bit blocks, and compares whether the length of the longest run of 'one's of the blocks are consistent with the expected number of those for a random sequence. We choose $m = 8$ and use the same categories, as NIST suggested [7]. Therefore, denoting the length of the longest run of

'one's within a block as V, we get $q_0 = Pr(V \leq 1) = 55/256$, $q_1 = Pr(V = 2) = 94/256$, $q_2 = Pr(V = 3) = 59/256$ and $q_3 = Pr(V \geq 4) = 48/256$. Here, the only variables for the calculation of p-values are the frequencies of the longest runs of 'one's in each category (V_i). In that case, $d = \lfloor n/8 \rfloor$ implies that, for each $i = 0, \ldots, 3$ we have

$$Pr(V_i = x_i) = \frac{\binom{d}{x_0}\binom{d-x_0}{x_1}\binom{d-x_0-x_1}{x_2}\binom{d-x_0-x_1-x_2}{x_3} {q_0}^{x_0} {q_1}^{x_1} {q_2}^{x_2} {q_3}^{x_3}}{2^n}.$$

2.1.6 Serial Test

The subject of the Serial Test [7] is the frequency of all possible overlapping m and $(m-1)$-bit blocks of a sequence. Two p-values are produced for this test. We take $m = 3$, and the p-values are determined by the frequencies of 1-bit, 2-bit and 3-bit blocks. We first calculate the probability of frequencies for $m = 2$, then $m = 3$.

Let X denote the number of 'zero's, and Y denote the number of 'one's in a sequence. Similarly, let A, B, C and D denote the number of 00, 01, 10 and 11 blocks respectively. In the case of $m = 2$, the p-value is computed depending on variables X, Y, A, B, C and D. Therefore, our aim is to compute $Pr(X = X_1, Y = Y_1, A = A_1, B = B_1, C = C_1, D =_1)$. We first note some immediate facts:

Fact 2.1.1 *For any sequence the number of 01 blocks is equal to the number of 10 blocks (that is, $B = C$).*

Proof. Each run in the sequence defines either a 01 or a 10 block, if the sequence contains more than one runs. Also, in two consecutive runs, there will be a 01 and a 10 block. Considering the overlapping blocks, the number of runs will be even, which implies $B = C$. ■

Fact 2.1.2 *X=A+B.*

Proof. Any 00 or 10 block is defined by a 0 bit. Therefore, the total number of 00 and 10 blocks sum up to the number of 'zero's. ■

Since $A+B+C+D = n$, if n, X, and B are known, the values of the other parameters can easily be computed using the facts above. Hence, to produce the probabilities for each subinterval,

one needs a probability table which covers $Pr(B = B_1, X = X_1)$ values for all (B_1, X_1) tuples and the p-value derived from the sequence.

Our aim is to compute the probability $Pr(B = B_1, X = X_1)$. Now, if $B_1 = 0$, the sequence is either an all one or an all zero sequence. Thus the probability $Pr(B = 0, X = X_1) = 2/2^n$. If $B_1 > 0$, the probability $Pr(B = B_1, X = X_1)$ is computed in three steps assuming the bits are arranged on a circle to visualize the overlapping characteristic of the test. First, locate B_1 many 01 blocks on the circle. Then place $(X_1 - B_1)$ many 'zero's between 01 blocks. The number of possible arrangements is equal to the number of nonnegative integer solutions of the equation $x_1 + x_2 + \cdots + x_{B_1} = X_1 - B_1$ which is $\binom{X_1-1}{B_1-1}$. Afterwards, locate the remaining $(n - X_1 - B_1)$ many 'one's. These bits can not be placed between 'zero's, otherwise the number of 01 blocks would increase. Therefore, these 'one's should be placed adjacent to other 'one's on the circle. The number of such arrangements is equal to the number of nonnegative integer solutions of the equation $x_1 + x_2 + \cdots + x_{B_1} = n - X_1 - B_1$ which is equal to $\binom{n-X_1-1}{B_1-1}$.

Here, each arrangement on the circle gives n sequences. However, since there are b many 01 blocks, B_1 of these sequences are identical. Therefore, the probability $Pr(B = B_1, X = X_1)$ is equal to the number of possible arrangements divided by all possible n bit sequences which gives,

$$Pr(B = B_1, X = X_1) = \begin{cases} \dfrac{\binom{X_1-1}{B_1-1}\binom{n-X_1-1}{B_1-1}\frac{n}{B_1}}{2^n} & \text{if } B_1 > 0 \\ \dfrac{2}{2^n} & \text{otherwise.} \end{cases}$$

In the case of $m = 3$, the probability can be computed similarly. Let E_0 be the event that the number of 01 blocks is B_1 and the numbers of 000, 001, 010, 011, 100, 101, 110, 111 blocks are a, b, c, d, e, f, g, h respectively. Assume that $b > 0$ and locate b many 001 blocks on the circle. Then place a many 'zero's to form 000 blocks. The number of such arrangements is equal to the number of nonnegative solutions of the equation $x_1 + x_2 + \cdots + x_b = a$ which is equal to $\binom{a+b-1}{a}$.

All possible 00 blocks are located. The remaining 'zero's should be located such that the number of 00 blocks remains same. This can be accomplished by locating $B_1 - b$ many 01 blocks between b many $00\cdots01$ blocks. The number of such arrangements is equal to the number of nonnegative solutions of the equation $x_1 + x_2 + \cdots + x_b = B_1 - b$ which is equal to $\binom{B_1-1}{b-1}$.

All ‘zero’s are located. Remaining ‘one’s will be located in two steps. First locate d many 011 blocks. Since all 01 blocks are located, d of these blocks are selected to form a 011 block. The number of such arrangements is equal to $\binom{B_1}{d}$. Then locate h many 111 blocks. This is accomplished by placing h ‘one’s adjacent to 011 blocks. The number of such arrangements is equal to the number of nonnegative solutions of the equation $x_1 + x_2 + \cdots + x_d = h$ which is equal to $\binom{h+d-1}{h}$.

Finally, each circular solution gives n arrangements. However, since there are b many 001 blocks in the sequence, s of these arrangements are identical. Thus if $b > 0$,

$$Pr(E_0) = \frac{\binom{a+b-1}{a}\binom{B_1-1}{b-1}\binom{B_1}{d}\binom{h+d-1}{h}\frac{n}{b}}{2^n}$$

is obtained.

2.1.7 Approximate Entropy Test

Similar to the Serial Test, Approximate Entropy Test [7] compares the frequencies of overlapping m and $(m + 1)$-bit blocks of a sequence with the expected frequencies of those in a random sequence. We take $m = 2$, and the p-values are determined by the frequencies of 1-bit, 2-bit and 3-bit blocks. Therefore the calculation of probabilities is the same with the Serial Test.

2.1.8 Non-overlapping Template Matching Test

The subject of the Non-overlapping Template Matching Test [7] is the frequency of a pre-specified block in the sequence. An m-bit window is used to search the m-bit overlapping blocks of the sequence. The pre-specified block is chosen in a manner that if the pattern is observed somewhere in the sequence, then it should not be seen before the template is completed, therefore the window slides m bits in that case.

We choose $m = 4$ and the pre-specified block as 0001. Let K denote the number of 0001 blocks in the sequence and assume that we know the weight W and the number of runs V of the sequence. If the sequence is not all zero or all one sequence the number of runs is even. If we compute $Pr(K = k|W = w, V = 2r)$ then by summing over all possible weights and runs, we obtain $Pr(K = k)$. Similar to the Serial Test, the bits are arranged on a circle and we write

'one's and 'zero's consecutively to define $2r$ runs. As all the 0001 blocks contain 01 blocks, if a run of 'zero's has more than 2 'zero's, it produces one 0001 block.

Now, we find the distribution of $n - w - r$ many 'zero's and $w - r$ many 'one's so that the number of 0001 blocks is k and $V = 2r$. The number of such arrangements is equal to the number of integer solutions of the system

$$\begin{aligned} x_1 + x_2 + \cdots + x_r &= \quad n - w - r, \quad 0 \le y_i \\ y_1 + y_2 + \cdots + y_r &= \qquad w - r, \qquad x_{\alpha_j} \ge 2, \;\; 0 \le x_{\alpha_s} \le 1 \end{aligned}$$

where $(\alpha_1, \alpha_2, \ldots, \alpha_r)$ is a permutation of $(1, 2, \ldots, r)$, and $1 \le i \le r$, $1 \le j \le k$, $k+1 \le s \le r$.

The second equation has $\binom{w-1}{r-1}$ solutions. In order to find the number of solutions of the first equation, inclusion-exclusion principle can be applied but assuming that a of x_{α_s}, are one and the remaining are zero, the number of integer solutions of the first equation is the same with the number of non-negative integer solutions of the equation

$$z_1 + z_2 + \ldots + z_k = n - w - r - 2k - a,$$

which is $\binom{n-w-r-a-k-1}{k-1}$.

Considering the circular symmetry, other than all zero or all one sequence, we have

$$Pr(K = k) = \frac{n}{r} \sum_{w=1}^{n-1} \sum_{r=1}^{n/2} \binom{w-1}{r-1} \binom{r}{k} \sum_{a=0}^{r-k} \binom{r-k}{a} \binom{n-w-r-a-k-1}{k-1}.$$

2.1.9 Overlapping Template Matching Test

The subject of the Overlapping Template Matching Test [7] is the frequency of a pre-specified block in the sequence, and similar to the Non-overlapping Template Matching Test, an m-bit window is used to search the m-bit overlapping blocks of the sequence. However, if the pattern is observed then the window slides only one bit.

We choose $m = 4$ and the pre-specified block as 1111. Similar to the Non-overlapping Template Test, let K denote the number of 1111 blocks in the sequence. Assuming that we know the weight W and the number of runs V of the sequence, our aim is to find $Pr(K = k | W = w, V = 2r)$. First, the bits are arranged on a circle and we write 'one's and 'zero's consecutively to define $2r$ runs. Each run of 'one's with length $l \ge 4$ defines $l - 3$ 1111 blocks. Now,

we find the distribution of $n - w - r$ many 'zero's and $w - r$ many 'one's so that the number of 1111 blocks is k and $V = 2r$. The number of such distributions is equal to the number of integer solutions of the system

$$\begin{aligned} x_1 + x_2 + \cdots + x_r &= n - w - r \\ y_1 + y_2 + \cdots + y_r &= w - r \end{aligned}$$

with t of $y_i \geq 3$, $1 \leq i \leq r$.

The first equation has $\binom{n-w-1}{r-1}$ solutions. Without loss of generality (or multiplying by $\binom{r}{t}$), assume $y_j \geq 3$, $1 \leq j \leq t$, and $0 \leq y_s \leq 2$, $t + 1 \leq s \leq r$. We find the number of solutions of the second equation in two parts:

In order to have k many 1111 blocks, we should have

$$\begin{aligned} y_1 - 2 + y_2 - 2 + \cdots + y_t - 2 &= k, \;\; y_j \geq 3 \\ y_1 + y_2 + \cdots + y_t &= k + 2t, \;\; y_j \geq 3 \\ z_1 + z_2 + \cdots + z_t &= k - t, \;\; z_j \geq 0; \end{aligned}$$

therefore, the number of solutions are $\binom{k-1}{t-1}$ (and $t = 0 \Leftrightarrow k = 0$).

As the weight of the sequence is w, we have:

$$y_{t+1} + y_{t+2} + \cdots + y_r = w - r - k - 2t$$

with $0 \leq y_i \leq 2$, $t + a \leq i \leq r$. We apply the Inclusion Exclusion principle to find the number of the solutions and obtain:

$$\sum_{i=0}^{\lfloor \frac{w-r-k-2t}{3} \rfloor} \binom{w - k - 3t - 3i - 1}{r - t - 1} \binom{r - t}{i} (-1)^i.$$

Considering the circular symmetry, other than all zero or all one sequence, we have

$$Pr(K = k) = \frac{n}{r} \sum_{w=1}^{n-1} \sum_{r=1}^{n/2} \binom{n - w - 1}{r - 1} \sum_{t=0}^{r} \binom{r}{t} \sum_{i=0}^{\lfloor \frac{w-r-k-2t}{3} \rfloor} \binom{w - k - 3t - 3i - 1}{r - t - 1} \binom{r - t}{i} (-1)^i.$$

2.1.10 Linear Complexity Test

The linear complexity of a finite binary sequence of length n, denoted by L, is the length of the shortest linear feedback shift register (LFSR) that generates a sequence whose first n terms are the given sequence.

Definition 2.1.3 *Let $a_0, a_1, \ldots, a_n$ be a binary sequence. For $C(D) = 1 + c_1 D + \cdots + c_L D^L$, let '$L, C(D)$'be an LFSR that generates $a_0, a_1, \ldots, a_n$. The next discrepancy d_n is the difference between a_n and $(n+1)^{th}$ term generated by the LFSR [3].*

Linear complexity profile of a sequence can be computed using the following algorithm [3]:

$$L_{n+1} = \begin{cases} L_n & if\ L_n > n/2, \\ L_n & if\ L_n \leq n/2\ and\ d_n = 0, \\ n + 1 - L_n & if\ L_n \leq n/2\ and\ d_n = 1, \end{cases} \tag{2.2}$$

where d_n is the next discrepancy.

Berlekamp Massey algorithm [3] is widely used to find the linear complexity of a sequence. However, as the complexity of this algorithm is $O(n^2)$ bit operations, it is not feasible for large values of n. Therefore, in NIST Test Suite the Linear Complexity Test is applied by dividing the sequence into M-bit subsequences (where $500 \leq M \leq 5000$), obtaining the linear complexity of each subsequence and applying χ^2 Goodness of Fit Test [7].

In [10], the number of sequences of length n and having linear complexity L, denoted by $N_n(L)$ is given as

$$N_n(L) = \begin{cases} 2^{min(2n-2L, 2L-1)} & if\ n \geq L > 0, \\ 1 & if\ L = 0. \end{cases}$$

As we apply Linear Complexity Test to 256-bit sequences, we use this equation to determine the p-value and calculate the subinterval probabilities. Let l be the linear complexity of the sequence. If $l \leq EV$, then p-value $= 2 \cdot \sum_{i=0}^{l} p_i$, where EV is the expected value of the linear complexity and p_i is the probability $Pr(LC = i)$. Similarly if $l > EV$, then p-value $= 2 \cdot \sum_{i=l}^{256} p_i$. But since $Pr(LC = n/2) = 1/2$, we get p-value> 1 for $l = n/2$, therefore the p-value is set to 1 in this case. p-values of Linear Complexity Test for $n = 256$ is presented in Table 2.1.

2.1.11 Linear Complexity Profile Test

Let $a_1, a_2, \ldots, a_n$ be a binary sequence and L_t denote the linear complexity of the subsequence $a_1, a_2, \ldots, a_t$. The sequence $L_1, L_2, \ldots, L_n$ is called the linear complexity profile of the sequence. The subject of Linear Complexity Profile Test is the number of the distinct elements in the linear complexity profile of a sequence, which is denoted by LCP.

Table 2.1: p-values of Linear Complexity Test for $n = 256$

Linear Complexity	Probability	p-value
123	0,00048828	0,001302
124	0,00195313	0,005208
125	0,0078125	0,020833
126	0,03125	0,083333
127	0,125	0,333333
128	0,5	1
129	0,25	0,666667
130	0,0625	0,166667
131	0,015625	0,041667
132	0,00390625	0,010417
133	0,00097656	0,002604

Example 2.1.4 *Let $a_n = \{1, 0, 0, 1, 1, 0, 1, 0\}$, then the linear complexity profile of a_n is 1, 1, 1, 3, 3, 3, 4, 4 and $LCP = 3$.*

We use equation 2.2 to find $Pr(LCP = s)$. First, we prove a theorem to find this probability.

Theorem 2.1.5

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

Proof. If $LCP = 1$ then $LC = 0$ and the sequence is $\{0, 0, \ldots, 0\}$ or $LC = 1$ and the sequence is $\{1, 0, \ldots, 0\}$ or $LC = 1$ and the sequence is $\{1, 1, \ldots, 1\}$. If $LC = n$ then the sequence is $\{0, 0, \ldots, 1\}$ and $LCP = 2$. If $LCP > 1$ and $LC \neq n$ we prove the theorem, using induction on n. All possible LC and LCPs for $n = 4$ is given in table 2.2 and the claim is true for $n = 4$. Assume, the claim is true for $n = N - 1$. For $n = N$, there are three cases.

If $t < \frac{n}{2}$ the claim is true by Berlekamp Massey algorithm.

Table 2.2: LC and LCP values for $n = 4$

LC	LCP	#
0	1	1
1	1	2
2	2	8
3	2	4
4	2	1

If $t = \frac{n}{2}$, which is valid for even n, we have

$$\begin{aligned} Pr(L = n/2, LCP = i, l = n) &= 2 \cdot Pr(L = n/2, LCP = i, l = n-1) \\ &= 2 \cdot 2^{n-\frac{n}{2}} \binom{n - \frac{n}{2} - 2}{i-2} = 2^{\frac{n}{2}+1} \binom{\frac{n}{2} - 2}{i-2}, \end{aligned}$$

as L_{n-1} should be $n/2$.

If $t > \frac{n}{2}$, we have

$$\begin{aligned} & Pr \quad (L = t, LCP = i, l = n) \\ &= 2 \cdot Pr(L = t, LCP = i, l = n-1) + Pr(L = n-t, LCP = i-1, l = n-1) \\ &= 2 \cdot 2^{n-t} \binom{n-t-2}{i-2} + 2^{n-t+1} \binom{n-t-2}{i-3} = 2^{n-t+1} \binom{n-t-1}{i-2}. \end{aligned}$$

■

Theorem 2.1.6 *For even n,*

$$Pr(LCP = s) = \begin{cases} 3/2^n & if\ s = 1, \\ \dfrac{3 \cdot \sum_{i=2}^{n/2} 2^i + 1}{2^n} & if\ s = 2, \\ \dfrac{3 \cdot \sum_{i=2}^{n/2} 2^i \binom{i-2}{s-2}}{2^n} & if\ s > 2. \end{cases}$$

Proof. We use theorem 2.1.5 to calculate $Pr(LCP = s)$. If $s = 1$ we have $Pr(LCP = 1) = 3/2^n$. If $s > 2$ we have

$$\begin{aligned}
Pr(LCP = s) &= \sum_{i=0}^{n} Pr(L = i, LCP = s) \\
&= \frac{\sum_{i=0}^{n/2} 2^{i+1}\binom{i-2}{s-2}}{2^n} + \frac{\sum_{i=n/2+1}^{n-1} 2^{n+1-i}\binom{n-1-i}{s-2}}{2^n} \\
&= \frac{2 \cdot \sum_{i=2}^{n/2} 2^{i}\binom{i-2}{s-2}}{2^n} + \frac{\sum_{i=1}^{n/2-1} 2^{n/2+1-i}\binom{n/2-1-i}{s-2}}{2^n} \\
&= \frac{2 \cdot \sum_{i=2}^{n/2} 2^{i}\binom{i-2}{s-2}}{2^n} + \frac{\sum_{i=-n/2+1}^{-1} 2^{n/2+1+i}\binom{n/2-1+i}{s-2}}{2^n} \\
&= \frac{2 \cdot \sum_{i=2}^{n/2} 2^{i}\binom{i-2}{s-2}}{2^n} + \frac{\sum_{i=2}^{n/2} 2^{i}\binom{i-2}{s-2}}{2^n} \\
&= \frac{3 \cdot \sum_{i=2}^{n/2} 2^{i}\binom{i-2}{s-2}}{2^n}.
\end{aligned}$$

If $s = 2$ we also have $Pr(L = n, LCP = 2) = 1/2^n$, thus we add $1/2^n$ to this probability. ■

We use theorem 2.1.6 to determine the p-values and find the sunbinterval probabilities.

Algorithm 2.1.2: Linear Complexity Profile Test($\{a_1, a_2, \ldots, a_n\}$)

for $i \leftarrow 1$ **to** n
  do
$\{L_i = BerlekampMassey(a_1, a_2, \ldots, a_i);$
$LCP = 1$
for $j \leftarrow 2$ **to** n
  do
$\begin{cases} \textbf{if } L_i \neq L_{i-1} \\ \quad \textbf{then } LCP = LCP + 1; \end{cases}$
Use table B.1 to determine the $p - value$;
return $(p - value)$

2.1.12 Rotation Test

The derivative of a finite binary sequence $a_1, a_2, \ldots, a_n$ is defined as $b_i = a_i \oplus a_{i+1}$ for $1 \leq i \leq n-1$ and $b_n = a_n \oplus a_0$. The subject of the Binary Derivative Test [5] is the weight of

the derivative of a sequence. The weight of the first derivative is the number of runs in the sequence.

As 2^k th derivative is $b_n = a_n \oplus a_{n+2^k}$, we propose Rotation Test, where the sequence is rotated t bits and the new sequence is XORed to the original sequence. In other words $b_n = a_n \oplus a_{n+t}$ and the Frequency Test is applied to b_n. We choose $t = 8$, and the subinterval probabilities are the same with the Frequency Test.

Algorithm 2.1.3: ROTATION TEST($\{a_1, a_2, \ldots, a_n\}, t$)

for $i \leftarrow 1$ **to** n

do

$\{ b_i = a_i \oplus a_{i+t(mod\ n)}$;

$weight = \sum_{i=1}^{n} b_i$;

$p - value = erfc(\frac{2 \cdot weight - n}{\sqrt{2}})$

return $(p - value)$

2.1.13 Lempel-Ziv Compression Test

Lempel-Ziv complexity, which is a basis of LZ77 compression algorithm, is an important measure used in cryptography. Lempel-Ziv complexity is the subject of the Lempel-Ziv Compression Test [7], which is excluded from the NIST Test suite.

Doğanaksoy et. al [11] calculated the subinterval probabilities for short sequences. Let $A(n, r)$ denote the set of binary sequences of length n, and Lempel-Ziv complexity r, $C(n, r)$ denote the set of closed sequences (the sequences with the property that if n is increased by 1 then r also increases). Then in [11] it is found that:

$$\begin{aligned} a(n,r) &= 2c(n-1,r-1) + 2\left(a(n-1,r) - c(n-1,r)\right) \\ c(n,r) &= 2c(n-r,r-1) + \sum_{0 \le a \le n} \sum_{1 \le p < r} \binom{r}{p} c(a-p,p-1)c(n-a-r+p,r-p-1) \end{aligned}$$

where $a(n, r) = |A(n, r)|$ and $c(n, r) = |C(n, r)|$. Using these equations, the subinterval probabilities are presented in [11].

2.1.14 Test Related to Random Walk

Consider a binary sequence $a_1, a_2, \ldots a_n$ and let $b_i = 1 - 2a_i$ with $b_i \in \{-1, 1\}$, for $i = 1, 2, \ldots, n$ and $s_j = \sum_{i=1}^{j} b_i$ for $j = 1, 2, \ldots, n$. Then in $x - y$ plane, the path formed by $(1, s_1), (2, s_2), \ldots, (n, s_n)$ represents a random walk if $Pr(a_i = 0) = Pr(a_i = 1) = 1/2$ for all $1 \leq i \leq n$.

The points at which the path intersects the x-axis correspond to $s_j = 0$. The part of the sequence between two such successive points is referred to an *excursion*. *Length* of an excursion starting at b_i and ending at b_j is $j - i$. The maximum value of $|s_n|$ is defined as the *height* of a random walk.

2.1.14.1 Random Walk Excursion Test

Random Walk Excursion Test, defined by Doğanaksoy et. al. [12], evaluates the sequence as a random walk and the subject of the test is the number of excursions. The probability distribution for the length of the random excursions, $P(l)$, is given by

$$P(l) = \begin{cases} \frac{C_{k-1}}{2^{2k-1}} & if\ l = 2k \\ 0 & otherwise \end{cases}$$

where C_i is the Catalan number defined as $\frac{1}{i+1}\binom{2i}{i}$ [12].

Let P_k be the probability of a random excursion to be of length greater than k. Obviously,

$$P_{2k} = 1 - \sum_{i=1}^{k} P(2i)$$

and $P_{2k} = P_{2k+1}$.

Given a sequence of even length N, to compute the probability $P(N, k)$ of having exactly k excursions in the sequence, two cases are considered:

- The sequence is balanced. This means that the sequence consists of k complete excursions. If $N/2 = p_1 + p_2 + \cdots + p_k$ is a partition of $N/2$ into k positive integers $p_1, \ldots, p_k$ we have $N = 2p_1 + 2p_2 + \cdots + 2p_k$. Then the probability of having k successive excursions of respective lengths $2p_1, \ldots, 2p_k$ is $P(2p_1).P(2p_2) \ldots P(2p_k)$. Consequently,

the required probability is given by the sum $\sum P(2p_1) \wedge P(2p_k)$ where the summation ranges over all ordered partitions $(p_1, \ldots, p_k)$ of $N/2$ into k positive integers.

- The sequence is not balanced. This means that the sequence consists of $k-1$ (not excluding the possibility that $k-1=0$) complete excursions of total length $N-2m$ and an incomplete (last) one of length $2m$ ($m>0$). In this case the required probability is $\sum_{m=1}^{N/2} P_{N-m} \sum P(2p_1) \wedge P(2p_{k-1})$ where the inner summation runs over all ordered partitions $p_1, \ldots, p_{k-1}$ of $N/2-m$ into $k-1$ positive integers.

The subinterval probabilities are presented in [12].

2.1.14.2 Cumulative Sums Test

Cumulative Sums Test [7] evaluates the sequence as a random walk and compares the height of the random walk to the expected value for a random sequence. The only variable for computing the p-value is the variable z, which is the maximum distance of random walk from the x-axis. Note that the test produces two p-values; one from producing the random walk from left to right, and one from right to left. Assume we know the weight W of the sequence, we need to calculate $Pr(z = r|W = w)$ for a given sequence of length n.

We represent random walk in a $w \times (n-w)$ grid as a path starting from the bottom left corner, (stepping one unit up for a 0 term and stepping one unit right for a 1 in the sequence) and ending at the top right corner. A path which intersects the upper inclined line shown in the figure corresponds to a random walk with $S_{max} \geq r$. The number of such paths is known to be equal to $\binom{n}{w-r}$.

Let us investigate a more general case. We want to find the number of paths starting from A, finishing at B and not cutting the two given lines, that is a random walk with weight w and taking values between $[-r+1, r-1]$. Using inclusion-exclusion principle the probability is given as:

$$Pr(z < r|W = w) = \begin{cases} \dfrac{\binom{n}{w} - \sum_{i=1}^{\infty} \left(\left[\binom{n}{w-ir} + \binom{n}{n-w-ir} \right] (-1)^{i+1} \right)}{2^n} & \text{if } r > |n-2w| \\ 0 & \text{otherwise.} \end{cases}$$

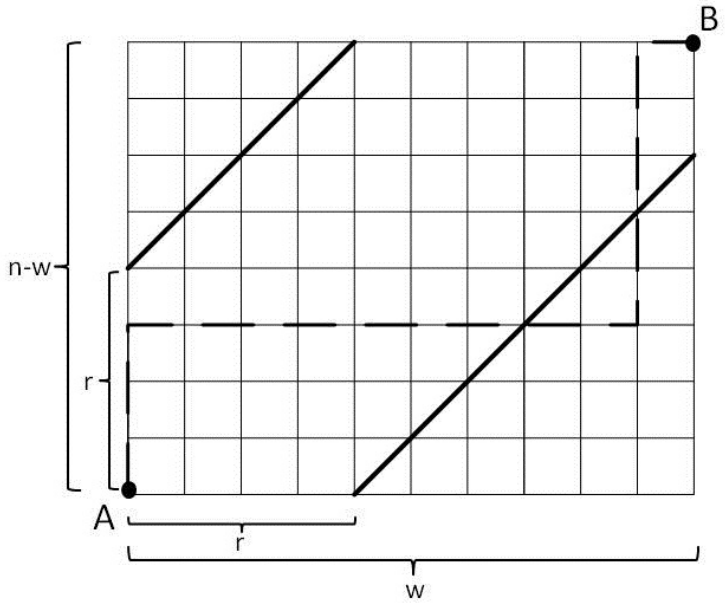

Figure 2.1: Random walk represented in a grid

Considering all possible weights of an n-bit sequence, we get

$$Pr(z < r) = \sum_{w=0}^{n} Pr(z < r|W = w).$$

Finally, we have $Pr(z = r) = Pr(z < r + 1) - Pr(z < r)$.

2.1.15 Test Related to Integers

For all the tests in this subsection, first the binary sequence $\{a_1, a_2, \ldots, a_n\}$ is divided into k-bit blocks and the corresponding integer values of the subsequences evaluated (the remaining bits are discarded). Then, an integer sequence of $\{t_1, t_2, \ldots, t_{\lfloor n/k \rfloor}\}$ with $0 \leq t_i \leq 2^k - 1$ is tested.

2.1.15.1 Integer Frequency Test

This test is a frequency test on the integers and the subject of this test is the weights of each integer. Let F_i denote the frequency of integer i, then

$$Pr(F_i = t) = \binom{\lfloor n/k \rfloor}{t} \left(\frac{1}{2^k}\right)^t \left(1 - \frac{1}{2^k}\right)^{\lfloor n/k \rfloor - t}.$$

A χ^2 Goodness of Fit Test with k^2 bins is applied to produce p-value. For $n = 256$, we choose $k = 3$ and calculate the subinterval probabilities.

Algorithm 2.1.4: INTEGER FREQUENCY TEST($\{a_1, a_2, \ldots, a_n\}, k$)

for $i \leftarrow 1$ **to** $\left\lfloor \frac{n}{k} \right\rfloor$
do
$t_i = \sum_{j=1}^{k} 2^{k-j} a_{j+(i-1)k}$;

for $i \leftarrow 0$ **to** $2^k - 1$
do
$e_i = 0$;
for $j \leftarrow 1$ **to** $\left\lfloor \frac{n}{k} \right\rfloor$
do
if $t_j = i$
then $e_i = e_i + 1$;
Apply χ^2 *of Goodness of Fit test to* e_i *values*;
return ($p - value$)

2.1.15.2 Maximum of t Test

Maximum of t Test is first defined by Knuth [4]. The subject of the test is the maximum integer in the sequence, denoted by Max. First we calculate $Pr(Max \leq t)$. This means that all the elements are chosen from the subset $\{0, 1, \ldots t\}$, and the probability is given as $Pr(Max \leq t) = \left(\frac{t+1}{2^k}\right)^{\lfloor n/k \rfloor}$. Therefore,

$$\begin{aligned} Pr(Max = t) &= Pr(Max \leq t+1) - Pr(Max \leq t) \\ &= \left(\frac{t+1}{2^k}\right)^{\lfloor n/k \rfloor} - \left(\frac{t}{2^k}\right)^{\lfloor n/k \rfloor} \\ &= \frac{(t+1)^{\lfloor n/k \rfloor} - t^{\lfloor n/k \rfloor}}{2^n} \end{aligned}$$

For $n = 256$ we choose $k = 10$, and calculate the subinterval probabilities. p-values are calculated similar to the method mentioned in the Linear Complexity Test and given in Table B.2.

2.1.15.3 Maximum Difference Test

The subjects of the test are the maximum and the minimum integers in the sequence, denoted by Max and Min respectively. $Pr(Max = t)$ is calculated in the Max of t Test and $Pr(Min = t)$ is calculated similarly. We need to calculate $Pr(Max - Min = t)$, and this probability is given as

$$Pr(Max - Min = t) = \sum_{i=0}^{n-1-t} Pr(Max = t + i) \cdot Pr(Min = i).$$

For $n = 256$ we choose $k = 8$, and calculate the subinterval probabilities.

Algorithm 2.1.5: MAXIMUM DIFFERENCE TEST($\{a_1, a_2, \ldots, a_n\}, k$)

for $i \leftarrow 1$ **to** $\left\lfloor \frac{n}{k} \right\rfloor$
do
$t_i = \sum_{j=1}^{k} 2^{k-j} a_{j+(i-1)k}$;

$Max = 0$;
for $i \leftarrow 0$ **to** $\left\lfloor \frac{n}{k} \right\rfloor$
do
if $t_i > Max$
then $Max = t_i$;
$Min = 0$;
for $i \leftarrow 0$ **to** $\left\lfloor \frac{n}{k} \right\rfloor$
do
if $t_i < Min$
then $Min = t_i$;
$Difference = Max - Min$;
Use table B.3 to determine the $p - value$;
return $(p - value)$

2.1.15.4 Coupon Collector Test

The subject of the test is the different number of integers, denoted by Cov. We know that [4]

$$Pr(Cov = t) = \frac{2^k \cdot (2^k - 1) \cdots (2^k - (t-1))}{(2^k)^{\lfloor n/k \rfloor}} \left\{ \begin{matrix} \lfloor n/k \rfloor \\ t \end{matrix} \right\},$$

where $\left\{ {\lfloor n/k \rfloor \atop t} \right\}$ denote is the Stirling number of the second kind.

For $n = 256$ we choose $k = 6$, and calculate the subinterval probabilities. p-values are calculated similar to the method mentioned in the Linear Complexity Test and given in Table B.4.

2.1.15.5 Repeating Point Test

The subject of this test is the index of integer, denoted by RP, where the first repeating integer occurs. In order to calculate the subinterval probabilities, we need to find $Pr(RP = t)$. In that case the first $t - 1$ integer should be different and the t^{th} integer should be the same with one of the first $t - 1$ integers. Then

$$Pr(RP = t) = \frac{P(2^k, t-1) \cdot (t-1)}{2^{kt}}.$$

For $n = 256$ we choose $k = 6$, and calculate the subinterval probabilities. If there is no repeating integer in the sequence, we represent it by > 42 in Table A.6.

Algorithm 2.1.6: REPEATING POINT TEST($\{a_1, a_2, \ldots, a_n\}, k$)

for $i \leftarrow 1$ **to** $\left\lfloor \frac{n}{k} \right\rfloor$
  do
$\left\{ t_i = \sum_{j=1}^{k} 2^{k-j} a_{j+(i-1)k}; \right.$
$RP = 0;$
for $i \leftarrow 2$ **to** $\left\lfloor \frac{n}{k} \right\rfloor$
  do
$\left\{ \begin{array}{l} \textbf{for } j \leftarrow 1 \textbf{ to } i-1 \\ \quad \textbf{do} \\ \quad \left\{ \begin{array}{l} \textbf{if } t_i = t_j \\ \quad \textbf{then } RP = i; \textbf{ break} \end{array} \right. \end{array} \right.$
Use table B.5 to determine the $p - value$;
return $(p - value)$

2.1.15.6 Saturation Point Test

The subject of this test is the index of integer, denoted by SP, where all possible integers occur in the given sequence. In order to have $SP = t$, the coverage of the sequence for first $t-1$ integers should be $2^k - 1$ and the integer with index t should be the remaining integer. Following the same notation with Coupon Collector Test,

$$Pr(SP = t) = Pr(Cov = 2^k - 1 | K = t-1) \cdot \frac{1}{2^k} = \frac{2^k! \left\{ {t-1 \atop 2^k-1} \right\}}{2^{kt}},$$

where K denotes the number of integers in the sequence, is obtained. For $n = 256$ we choose $k = 4$, and calculate the subinterval probabilities. If all the integers do not occur in the sequence we represent it by > 64 in Table A.7.

Algorithm 2.1.7: SATURATION POINT TEST($\{a_1, a_2, \ldots, a_n\}, k$)

for $i \leftarrow 1$ **to** $\left\lfloor \frac{n}{k} \right\rfloor$
  do
$\{ t_i = \sum_{j=1}^{k} 2^{k-j} a_{j+(i-1)k}$;
for $i \leftarrow 1$ **to** 2^k
  do
$\begin{cases} index[i] = \left\lfloor \frac{n}{k} \right\rfloor + 1; \\ \textbf{comment: } initialization\ of\ index\ array \end{cases}$
$SP = 1$;
for $i \leftarrow 1$ **to** 2^k
  do
$\begin{cases} \textbf{for } j \leftarrow 1 \textbf{ to } \left\lfloor \frac{n}{k} \right\rfloor \\ \quad \textbf{do} \\ \quad \begin{cases} \textbf{if } t_j = i \\ \quad \textbf{then } index[i] = j; \textbf{break} \end{cases} \end{cases}$
for $i \leftarrow 1$ **to** 2^k
  do
$\begin{cases} \textbf{if } index[i] > SP \\ \quad \textbf{then } SP = index[i]; \end{cases}$
Use table B.6 to determine the $p - value$;
return $(p - value)$

Table 2.3: Correlation analysis results of p-values obtained from random data and transformations

	Comp 8	Comp 16	Comp 32	Swap 8	Swap 16	Rot 1 256	Rot 1 8
Fre	0,790	0,633	0,416	1,000	1,000	1,000	1,000
CuSum	0,802	0,653	0,432	0,952	0,906	0,989	0,992
BlFre	0,869	0,744	0,536	0,880	0,772	0,929	1,000
ApE	0,695	0,491	0,239	0,715	0,529	1,000	0,463
Ser1	0,624	0,384	0,156	0,636	0,409	1,000	0,337
Ser2	0,586	0,318	0,083	0,577	0,350	1,000	0,262
Run	0,660	0,443	0,216	0,653	0,458	0,983	0,435
LC	0,002	-0,003	0,005	0,001	-0,002	0,001	0,002
LCP	0,004	0,000	-0,002	-0,002	0,000	0,003	0,006
LonRun	0,694	0,487	0,237	0,717	0,525	0,469	0,517
Max	0,809	0,670	0,452	0,825	0,684	0,424	0,342
IntFre	0,782	0,608	0,361	0,807	0,652	0,544	0,394

2.2 Classification of Statistical Randomness Tests

Classification of statistical randomness tests is important to form a reasonable test suite. In this section we present just an introductory work. In order to classify the statistical randomness tests, we define certain transformations and observe how the tests are affected with respect to these transformations. We choose 10^5 random 256-bit data taken from [6] and produce new data sets defined by the transformations. Then we apply the statistical randomness tests to these data sets and perform a linear correlation analysis to p-values obtained from the original data and the new data, to measure the effect of the transformation. The results are presented in Table 2.3 where *Comp X* represents complementing random *X* bits, *Swap X* represents swapping random *X* bits, and *Rot X Y* represents rotating left *X* bits for each *Y* bits.

We define four classes by the correlation analysis results presented in Table 2.3:

1. Uniformity Tests: Frequency Test, Cumulative Sums Test, Frequency Test within a Block
2. Entropy Tests: Approximate Entropy Test, Serial Test, Runs Test
3. Complexity Tests: Linear Complexity Test, Linear Complexity Profile Test,
4. Other Tests: Test for the Longest Run of Ones in a Block, Maximum of t Test, Integer Frequency Test

CHAPTER 3

CRYPTOGRAPHIC RANDOMNESS TESTING

In chapter 2, we propose a new test package for block ciphers and hash functions, which evaluates the algorithms as PRNGs. In this chapter, we propose a package of cryptographic randomness tests, designed based on certain cryptographic properties of block ciphers and hash functions to evaluate their randomness.

3.1 Cryptographic Randomness Testing

In addition to the randomness of their outputs, block ciphers and hash functions should satisfy certain cryptographic properties as well. When block ciphers are considered, diffusion and confusion are among the most important properties, while for hash functions one of the basic design criteria is collision resistance. The cryptographic properties mentioned in this chapter are as follows:

- Whenever one input bit is changed, every output bit should change with probability a half to achieve ideal diffusion. This criterion is called the strict avalanche criterion (SAC).
- The distance of a boolean function to the set of all affine functions should be large. This property is measured in terms of nonlinearity, and it is a concept related to confusion.
- Finding two inputs that have the same output should be hard, which is called the collision resistance property.
- Block ciphers with a fixed plaintext and hash functions are one way functions and they are required to behave like a random mapping.

Corresponding to these four cryptographic criteria, we consider four randomness tests for block ciphers and hash functions:

- The aim of the SAC Test is to measure if an algorithm satisfies the SAC property.
- The Linear Span Test evaluates an algorithm by examining the linear dependence of the outputs formed from a highly linearly dependent set of inputs.
- The subject of the Collision Test is the number of collisions in a portion of the output corresponding to a random subset of the input set.
- Coverage Test takes a subset of the input set and examines the size of the corresponding output set.

In each section, the general idea, the mathematical background and application details of individual tests are given. For a better understanding of the concepts, it is assumed that the function under test is a function $f : F_2^n \times F_2^m \mapsto F_2^n$, where in the case of block cipher, n stands for the block size, and m stands for the key size. Similarly for the case of hash function, m and n stand for the message block size and the chaining variable size respectively.

3.2 SAC Test

The abbreviation SAC stands for strict avalanche criterion, which was originally proposed for S-boxes by Webster and Tavares in 1986 [13]. SAC states that for a particular S-box, whenever one input bit is changed, every output bit must change with probability $\frac{1}{2}$. Therefore, extending this idea to the round functions of block ciphers (or the compression functions of hash functions) as previously done for stream ciphers [14], this test evaluates the given function by examining the effect of a single bit flip on the output bits. To achieve this, an $m \times n$ matrix called the *SAC Matrix* is formed in the following way: first all entries of the *SAC Matrix* are set to 0, a random input is taken and the output is computed. Then, after flipping the i^{th} bit of the input, the corresponding output is XORed to the original output. Afterwards, for each non-zero bit j of the output, $(i, j)^{\text{th}}$ entry of the *SAC Matrix* is incremented by 1. This is done for each input bit i, and the whole process is repeated for 2^{20} different random inputs[1].

[1] This process should be repeated as many times as possible to detect small biases. If the performance of the function under testing is not good, the process is repeated for a less number of random inputs. Also, if the process is repeated more than 2^{20} times, the probabilities should be calculated with more than 6 digit precision.

Table 3.1: Ranges and probabilities of SAC Test for 2^{20} trials

Bin	Range	Probability
1	0-523857	0.200224
2	523858-524158	0.199937
3	524159-524417	0.199677
4	524418-524718	0.199937
5	524719-1048576	0.200224

Here, let K be the number of hits that an entry get, then

$$Pr(K = k) = \frac{\binom{n}{k}}{2^n} .$$

Therefore the expected value of each entry of the matrix is 2^{19}, and the distribution of the values of the matrix should follow a binomial distribution.

Following the construction of the *SAC Matrix*, χ^2 Goodness of Fit Test with the probabilities derived from Table 3.1 is used to evaluate the distribution of the values of the entire matrix. If a matrix produces a p-value less than 0.01, then it is considered non-random [14].

This method may fail to catch the correlation between a particular input and a particular output bit since the matrix is evaluated as a whole, therefore another method is proposed to evaluate each entry in the matrix. The entries outside a specific interval are flagged. The expected interval is taken as $[2^{19} - 5009, 2^{19} + 5009]$, and is computed so that a 2^{20}-bit sequence with a weight out of this interval would be assigned a p-value less than 10^{-6} from the Frequency Test. Since in the *SAC Matrix*, the number of entries is close to 10^6, it is not much unexpected to observe a term smaller than 10^{-6}. Therefore, the test is applied once more to check whether such a case is coincidental or not. If a flagged entry deviates from the expected value once more significantly, it is evident that a specific input bit and a specific output bit are correlated, which is a major cryptographic weakness, so the matrix is considered to be non-random.

3.3 Linear Span Test

Nonlinearity is one of the basic design criteria for cryptographic primitives. In order to test block ciphers and hash functions for randomness based on nonlinearity, the outputs of a highly linearly dependent set of inputs are examined. For this purpose, similar to the Linear Span

Test proposed for stream ciphers[1], linearly independent t plaintexts are chosen and an input set of size $m = 2^t$ is obtained by computing all linear combinations of these plaintexts. An $m \times m$ matrix is formed using the corresponding ciphertexts and the rank of this matrix is compared to the rank of a random binary matrix. After determining the rank of the output matrix, the corresponding bin value, which is initially set to 0, is incremented by one. After the test is repeated as many times as possible, the resulting bin values are put through a χ^2 Goodness of Fit Test with the probabilities given in Table 3.2 to produce the p-value. A p-value less than 0.01 is considered to indicate a non-random mapping.

Table 3.2: Probabilities used in Linear Span Test ($m > 19$)

Rank	$\leq m-2$	$m-1$	m
Probability	0.133636	0.577576	0.288788

The computation of the probability of a random binary matrix to have rank R for arbitrary R is not straightforward. However, an $m \times m$ random binary matrix has either rank m or $m-1$ over 85% of the time, therefore this test is applied with only three bins for the χ^2 Goodness of Fit Test and the probabilities for $Pr(R = m)$ and $Pr(R = m-1)$ cases are needed. In the case $Pr(R = m)$, all ciphertexts are linearly independent. There are $2^m - 1$ choices for the first plaintext, $2^m - 2$ choices for second plaintext, ..., $2^m - 2^{i-1}$ choices for i^{th} plaintext, ..., $2^m - 2^{m-1}$ choices for the last plaintext, therefore

$$Pr(R = m) = \frac{\prod_{i=1}^{m}(2^m - 2^{i-1})}{2^{m^2}}$$

is obtained.

In the case $Pr(R = m-1)$, first $m-1$ linearly independent ciphertexts are chosen similar to the first case. The last ciphertext should be chosen so that, it is linearly dependent with the previously selected set. If the linearly dependent ciphertext is the i^{th} one, there are 2^{i-1} choices, thus there are $1 + 2 + 2^2 + \cdots + 2^{m-1} = 2^m - 1$ choices for it, therefore

$$Pr(R = m-1) = \frac{\prod_{i=1}^{m-1}(2^m - 2^{i-1})}{2^{m^2}} \cdot (2^m - 1)$$

is obtained.

3.4 Collision Test

Collision resistance is an important design criterion for hash functions, which means that it should be hard to find two messages with the same hash value, and the Collision Test is designed to evaluate the randomness based on collision resistance. The subject of this test is the number of collisions in specific bits of the output, which can be considered as near collision. In other words, an input set of size n is evaluated through f, and the number of collisions (C) in t bits of the output is evaluated.

The same method with Knuth is used to calculate the probability that c collisions occur when n distinct random inputs are mapped into an output set of size $m = 2^t$ in [4]. The probability of c collisions occur is given as,

$$Pr(C = c) = \frac{m(m-1)\cdots(m-n+c+1)}{m^n} \left\{ \begin{matrix} n \\ n-c \end{matrix} \right\},$$

where $\left\{ \begin{matrix} n \\ n-c \end{matrix} \right\}$ is the Stirling number of the second kind. This probability is used to obtain the results given in Table 3.3. Here, as it affects directly the running time of the test, the selection of the parameter n should be done carefully. If it is chosen to be too large, it may not be possible to repeat the testing steps enough times to be able to detect subtle evidences of non-randomness. Therefore, 2^{12} and 2^{14} is chosen for the values of n in the calculations to have reasonable running times for the tests. The parameter m is chosen depending on the parameter n such that the probabilities are close enough to each other to be used in a 5-bin χ^2 Goodness of Fit Test.

Table 3.3: Ranges and probabilities of Collision Test for 16 and 20 bits

	$n = 2^{12}$, $m = 2^{16}$		$n = 2^{14}$, $m = 2^{20}$	
Bin	Range	Probability	Range	Probability
1	0-116	0.206246	0-117	0.190231
2	117-122	0.194005	118-124	0.215008
3	123-128	0.219834	125-130	0.211585
4	129-134	0.183968	131-137	0.202689
5	135-4096	0.195947	138-16384	0.180487

The test is applied as follows: first, a random input is taken and then an input set of size 2^{12} (or 2^{14}) is formed by assigning all possible values to its first 12-bits (or 14-bits respectively). After the outputs of these inputs are computed through f, number of collisions is obtained with

the help of an array or a hash table. Afterwards, the corresponding bin value, is incremented. Above steps are repeated for 2^{20} random inputs and the resulting bin values are evaluated through a χ^2 Goodness of Fit Test with the expected values derived from Table 3.3 to produce the p-value. A p-value less than 0.01 is considered to indicate a non-random mapping.

3.5 Coverage Test

The Coverage Test evaluates a given function f through examining the size of the output set (coverage) formed from a subset of its domain. For a random mapping, the output set size is expected to be about 63% ($1 - 1/e \cong 0.63212$) of the input set. Block ciphers loaded with a fixed plaintext and hash functions are one-way functions and required to behave like a random mapping. In the case of block ciphers, if the key is fixed, the function f becomes a random permutation and this case should be carefully investigated before moving on to the details of the test.

For a random permutation, the coverage is equal to the size of the input set, when all output bits are considered. For example, assume that the function f under test has block size n. Then, the coverage of an input set formed by 2^l distinct random values is obviously 2^l if all output bits are taken as reference. But, if $n - k$ output bits are considered when forming the output set, the maximum possible number of hits that an $(n - k)$-bit output can have is 2^k. Therefore, whenever $l > k$, f is expected to behave like a random mapping.

Hence, when applying this test, functions used in both block ciphers and hash functions are expected to behave like a random mapping, which implies that the expected coverage is about 63% of the size of the input set.

The calculations for the intervals and their probabilities are the same with the Coverage Test proposed by Turan et al. in [14] for stream ciphers. Let A_k be the number of mappings from an n-element set to an n-element set. Here, A_k is defined recursively as

$$A_k = \binom{n}{k}\left[k^n - \sum_{i=1}^{k-1}\binom{k}{i}\frac{A_{k-i}}{\binom{n}{k-i}}\right] .$$

Therefore, the probability of the coverage being k is

$$Pr(C = k) = \frac{A_k}{n^n} ,$$

for $k = 1, 2, \ldots, n$. The probabilities and intervals for the bins to be used for the test are given in Table 3.4. This table is slightly different than the one given in [14], since a typo is spotted and corrected when verifying the results given in that paper.

Table 3.4: Ranges and probabilities of Coverage Test for 12 and 14 bits

	12 - bits		14 - bits	
Bin	Range	Probability	Range	Probability
1	1-2572	0.199176	1-10323	0.201674
2	2573-2584	0.204681	10324-10346	0.195976
3	2585-2594	0.197862	10347-10367	0.207530
4	2595-2606	0.203232	10368-10390	0.195266
5	2607-4096	0.195049	10391-16384	0.199554

The test is applied as follows: first, a random input is taken and then an input set of size 2^{12} (or 2^{14}) is formed by assigning all possible values to its first 12-bits (or 14-bits respectively). After applying f to this input set, the coverage is computed and the corresponding bin value is incremented.

The resulting bin values are evaluated through a χ^2 Goodness of Fit Test with the expected values derived from Table 3.4 to produce the p-value. A p-value less than 0.01 is considered to indicate a non-random mapping.

CHAPTER 4

APPLICATION to BLOCK CIPHERS and HASH FUNCTIONS

In this chapter we apply the packages defined in chapter 2 and chapter 3 to the finalist algorithms in AES competition and the second round candidate algorithms in SHA-3 competition. First, we define how to produce data sets from the algorithms for statistical randomness testing, then we give the number of rounds where the algorithms achieve randomness.

4.1 Data Sets

Any cryptographic module must have the property that the redundancies at the input should not leak to the output. Depending on this understanding, we construct data sets having certain structures. Then we try to observe whether this structure is inherited by the module. In all of the tests, it is assumed that the algorithm subject to the test is a mapping $F_2^m \times F_2^n \longmapsto F_2^n$, where in the case of hash functions, m is the size of the message block and n is the size of the chaining variable, and in the case of block ciphers m is the size of the key and n is the size of the plaintext.

4.1.1 Low Density Message

Low Density Message (LW) data set is formed by binary strings of low weight. The message length of SHA-3 candidate algorithms are 32, 256, 512, 1088 and 1536 bits, and the plaintext length of AES finalist algoritms is 128 bits. In the 32-bit case, the data set consists of 32-bit binary strings of weight not exceeding 5. In 256-bit case, this bound is 3 and in the remaining cases the strings of weights 1 and 2 are taken. The number of sequences for different message lengths are given in Table 4.1.

Table 4.1: The number of sequences for different message lengths

m	weight	# Sequences
32	≤ 5	242 824
256	≤ 3	2 796 416
512	≤ 2	131 328
1088	≤ 2	592 416
1536	≤ 2	1 242 676

Table 4.2: Selection of k for 1-Bit Message Avalanche and Message Rotation

m	k	mk
32	32768	1 048 576
256	4096	1 048 576
512	2048	1 048 576
1088	963	1 047 744
1536	682	1 047 552

4.1.2 High Density Message

High Density Message (HW) data set is formed by choosing high density messages (or plaintexts) and they are formed similar to the low density message case. In other words, the inputs of the low density messages are complemented bitwise.

4.1.3 1-Bit Message Avalanche

In order to form 1-Bit Message Avalanche (Av1) data set, first a random message (or plaintext) R of length m is chosen. Then each time by flipping another bit of R, a set of m messages (or plaintexts) is formed and the corresponding hash values (or ciphertexts) are obtained. The same procedure is applied to k different messages (or plaintexts) to get a set of mk sequences. The values of k for different input lengths are given in Table 4.2.

4.1.4 8-Bit Message Avalanche

In order to form 8-Bit Message Avalanche (Av8) data set, first a random message (or plaintext) $R = b_0 b_1 \dots b_s$ of length $m = 8s$ is chosen, where b_i is a 8-bit word for $i = 0, 1, \dots, s$. Then,

Table 4.3: Selection of k for 8-Bit Message Avalanche

m	k	mk
32	1028	1 048 560
256	128	1 044 480
512	64	1 044 480
1088	30	1 040 400
1536	21	1 028 160

by assigning all possible 256 values to each word, $255 \cdot \frac{m}{8}$ different messages (or plaintexts) are formed and the corresponding hash values (or ciphertexts) are obtained. The same procedure is applied to k different messages (or plaintexts) and the values of k for different input lengths are given in Table 4.3.

4.1.5 Message Rotation

A random message (or plaintext) R of length m is chosen to form Message Rotation (Rot) data set, and a set of m messages is formed by consecutive 1-bit rotations of R and the corresponding hash values (or ciphertexts) are obtained. The same procedure is applied to k different messages (or plaintexts) to get a set of mk sequences.

4.2 Evaluation Method

Each data set produces a large set of p-values. Our first task is to obtain a single p-value associated with the data set under consideration. We apply χ^2 Goodness of Fit Test by partitioning the interval [0,1] into 10 equal subintervals as mentioned in chapter 2.

If we assume that the tests are independent, then the probability of Type I error (the data is random, but the null hypothesis is rejected) is computed as $1 - (1 - \alpha)^s$, where α is the level of level of significance, and s is the number of tests. We choose $\alpha = 0,0001$, in other words, if p-value$\geq 0,0001$ we conclude the data is random, and 0,0799% random data will be concluded as nonrandom.

In the case of $\alpha = 0,001$, 0,797% of random data will be concluded as nonrandom. Since we test several reduced versions of the algorithms, it is not much surprising to obtain a p-

value$< 0,001$. In that case we put a flag and repeat the test with a different data set (In the case of LW and HW, as it is not possible to change the data set, we change the weights of the strings to obtain new data sets.). If p-value$< 0,001$ is obtained for the same test, we conclude that the data is non-random.

4.3 Application to AES Finalist Algorithms

In this section, application of the statistical and cryptographic randomness tests to the AES finalist algorithms and their results are given. Brief descriptions are followed by a comparison with the previous work on the subject. The p-values are presented in Appendix C and Appendix D.

4.3.1 MARS

The block cipher MARS[15] uses three main function when encrypting a block of plaintext. First, an unkeyed mixing operation called the forward mixing is applied following a key whitening operation. Then, the keyed transformation called the cryptographic core is applied to the state. Finally, another unkeyed mixing called the backward mixing is applied and the ciphertext is obtained after another key whitening operation. In this work, as it is desired to test the cryptographic randomness properties of the selected block ciphers, only the cryptographic core of the algorithm is tested. In other words, the forward and backward mixing operations are excluded when applying the tests.

As Table 4.4 suggests, the cryptographic core of the algorithm satisfies the SAC property after 6 rounds. Since the cryptographic core has a type-3 Feistel network structure, this result is not unexpected. On the other hand, the suggested number of rounds for the cryptographic core is 16, which seems like a safe enough security margin.

4.3.2 RC6

RC6[16] is a 128 bit block cipher which uses 128,192 or 256 bit key sizes. The algorithm can be parametrized for other word and key sizes. The encryption process stars with a key whitening. Then, the round function is applied 20 times. The round function consists of

modular multiplication, modular addition, XOR and rotations operations. 20 rounds is followed by another key whitening which concludes the encryption process.

The test results given in Table 4.4 show that at least 5 rounds of the round function is enough to achieve randomness.

4.3.3 Rijndael

Rijndael[17] consists of four main operations: SubBytes, ShiftRows, MixColumns and AddRoundkey. The SubBytes operation is the confusion step, which uses an 8×8 s-box with very good cryptographic properties. The ShiftRows and MixColumns steps are mainly for satisfying the diffusion property. Finally, AddRoundkey is simply a key XOR at the end of each round. The MixColumn operation is skipped in the final round of encryption.

The results given in Table 4.4 suggests that randomness is achieved after 3 rounds. Moreover, the SAC property is satisfied after 4 rounds when the MixColumns operation is skipped in the last (fourth) round.

4.3.4 Serpent

Serpent[18] is an SP-Network using 32 rounds of successive substitution and permutation layers. Substitution layer consists of 32 4×4 *s*-boxes with good cryptographic properties. Permutation layer consists of a linear transformation using shift, rotation and XOR operations.

Serpent achieves randomness after 4 out of 32 rounds, which is a large enough security margin for block ciphers. The results are given in Table 4.4.

4.3.5 Twofish

Twofish[19] is a 128-bit block cipher that can handle variable-length key up to 256 bits. The cipher is a 16-round Feistel network that uses key-dependent 8×8 *s*-boxes, a 4×4 maximum distance separable matrix, a pseudo-Hadamard transform and rotations.

Due to the fact that the round function of the Twofish algorithm has a Feistel network structure, only the even number of its rounds are tested. The results given in Table 4.4 suggest that

the algorithm produces random outputs after 4 out of 16 rounds of its round function.

4.3.6 Comparison with the previous work

During the AES selection process, J. Soto proposed a method to test block ciphers for randomness using the NIST Test Suite[8]. However, as the tests defined in that suite are more suitable to test long sequences, the block ciphers are considered as PRNGs and the outputs obtained from various input types are concatenated to form long sequences. After the sequences are generated, the statistical randomness tests defined in the NIST Test Suite are applied to these sequences. As a result for this testing process, a total of 189 p-values are produced from 16 tests for each input type.

Contrary to the above mentioned work on the subject, the cryptographic randomness tests proposed in this work are defined solely for the purpose of testing the cryptographic properties of the algorithms. Four tests are applied to the algorithms and a total of six p-values are produced. Although the number of p-values produced is relatively small, the results obtained from the cryptographic randomness tests are more precise than the results of the above mentioned work (see Table 4.4). We also present the results of statistical randomness tests in Table 4.4.

Table 4.4: Combined table stating the number of rounds which the algorithms achieve randomness

	SAC	Lin Span	Collision		Coverage		Stat Rand	Pre Work[8]
Algorithms	-	-	16	20	12	14	-	-
MARS	6	2	3	3	3	3	4	4
RC6	5	2	5	5	5	5	4	4
Rijndael	4	2	3	3	3	3	3	3
Serpent	4	2	4	4	4	4	4	4
Twofish	4	2	4	4	4	4	4	4

4.4 Application to SHA3 Second Round Candidate Algorithms

In this section, we apply statistical analysis to the SHA-3 second round candidate algorithms. For this purpose, first we define reduced versions of the candidates. Then, we apply statistical randomness tests and cryptographic randomness tests to observe the number of rounds that

the algorithms achieve randomness.

When defining reduced versions of the algorithms, we only reduce the number of compression function rounds and ignore the finalization functions, if exist. Reduced rounds of finalization functions can also be taken into account, but this process together with the reduced rounds of compression functions increases the complexity of the tests by introducing plenty of versions for only one algorithm.

On the other hand, BMW and SHAvite-3 have two tunable parameters for the compression function that can be changed when reducing the algorithms. Therefore, to have a level testing field, we fix the first parameter and make the second one variable when defining the reduced versions of these algorithms. Also for SIMD, as the feed forward operation has a significant importance in the security of the algorithm, we do not exclude this operation in the reduced versions. Hence, 1 round of SIMD refers to 1 round of its compression function besides the feed forward operation.

For the sake of simplicity and ease of computations, we apply statistical analysis only to the 256-bit versions of the algorithms. In the case of cryptographic randomness testing, we apply the package directly to the compression functions of the candidate algorithms. Therefore, initialization and padding are excluded. In the case of statistical randomness testing, all the algorithms are tested for 256-bit output values, since the internal state size of the algorithms vary from 256 bits to 1600 bits and one needs to find the exact bin values of all the statistical randomness tests for each state size, which is not feasible.

In order to spot the indistinguishability of the compression functions, the initial values and salts are taken as zero vectors. Also finalization or output functions are excluded. Moreover, the padding operation is discarded and the output is generated from a single message block which is iterated only once.

4.4.1 Statistical Randomness Test Results

When applying statistical randomness tests to the candidates, we reduce the algorithms by excluding finalization, setting IV and salt values to 0. Moreover, we discard the padding operation and the output is generated from a single message block which is iterated only once.

Table 4.5: Statistical randomness test results for the 256-bit versions of the algorithms

Algorithm	# Rounds	# Rounds Random	Data Sets
Blake [20]	10	3	Av1
BMW [21]	2/14	2/1	All
CubeHash [22]	16	6	LW
ECHO [23]	8	4	Av1
Fugue [24]	2	>2	All
Grøstl [25]	10	3	LW, HW, Av1, Rot
Hamsi [26]	3	1	All
JH [27]	35.5	8	LW, HW, Av1, Av8
Keccak [28]	24	3	LW, HW, Av1, Av8
Luffa [29]	8	3	LW, HW
Shabal [30]	3	1	All
Shavite3 [31]	3/12	3/1	All
SIMD [32]	4	1	All
Skein [33]	72	8	All

For statistical tests, we produce approximately 2^{24} 256-bit outputs for each algorithm from the data sets mentioned in Section 4.1. In Table 4.5, for each algorithm, the number of rounds that the randomness is achieved is given together with the data sets which produce the corresponding results. According to these results, Fugue does not achieve randomness after its full rounds of compression functions. However, we observe that two rounds of $G1$ finalization or a single round of $G2$ finalization is enough for it to have random looking outputs.

4.4.2 Cryptographic Randomness Test Results

Different from statistical testing of the outputs of the algorithms, cryptographic randomness tests are applied directly to the compression functions of the candidate algorithms. As mentioned before, the core operations of each test are advised to be repeated 2^{20} times. However, especially in Coverage Test and Collision Test, 2^{12} calls (or 2^{14} calls depending on the chosen parameter) to the compression functions of the algorithms is required. This poses some limitations towards the number of times the cores of the tests can be repeated, since the performances of the reference codes provided in the submissions of the candidate algorithms vary significantly. Hence, we repeat the core operations of the tests as given in Table 4.6 to have reasonable running times for all algorithms.

Table 4.6: Number of times the core of the tests repeated with corresponding testing parameters

Test (Parameter)	# Repetitions
SAC Test (-)	2^{20}
Linear Span Test (-)	2^{16}
Collision Test (16)	2^{16}
Collision Test (20)	2^{12}
Coverage Test (12)	2^{16}
Coverage Test (14)	2^{12}

Table 4.7: Cryptographic randomness test results for the compression functions of the 256-bit versions of the algorithms

Algorithm	# Rounds	# Rounds Random	Tests that give the best results
Blake	10	2	Col. Test, Cov. Test, SAC Test
BMW	2/14	2/5	SAC Test
CubeHash	16	7	SAC Test
ECHO	8	2	Col. Test, Cov. Test, SAC Test
Fugue	2	> 2	Col. Test, Cov. Test, SAC Test
Grostl	10	3	Col. Test (20)
Hamsi	3	> 3	Col. Test, Cov. Test
JH	35.5	11	Cov. Test (12), SAC Test
Keccak	24	4	SAC Test
Luffa	8	4	SAC Test
Shabal	3	> 3	Col. Test, Cov. Test, SAC Test
Shavite-3	3/12	3/3	SAC Test
SIMD	4	1	Col. Test, Cov. Test, SAC Test, LS Test
Skein	72	9	SAC Test

In Table 4.7, the number of rounds that the compression functions achieve randomness is given together with the tests from which the best results are obtained for each algorithm. The abbreviations LS Test, Col. Test and Cov. Test in the table stand for Linear Span Test, Collision Test and Coverage Test respectively. According to these results, compression functions of Fugue, Hamsi and Shabal do not achieve randomness after their full rounds, but each of these algorithms have finalization rounds after all message blocks are processed. Therefore, we believe it is highly unlikely that these observations on the compression functions will lead to a cryptanalysis effort on the hash function.

The results in Table Table 4.7 shows how conservative the compression functions are. According to these results Keccak, SIMD and Skein are the most conservative designs. On the other hand, the results obtained from Fugue, Shabal and Hamsi indicate relatively weak compression functions used for their designs. However they make up for this with use of cryptographically strong finalization functions.

CHAPTER 5

CONCLUSION

In this thesis, we have studied the methods to evaluate block ciphers and hash functions statistically. For this purpose, we determine the statistical randomness tests, which produce reliable results for short sequences and propose an alternative evaluation method. Then, we propose a method to classify the statistical randomness tests. Afterwards, we propose a new test suite and apply this suite to the finalist algorithms in AES competition and the second round candidate algorithms in SHA-3 competition, by constructing data sets having certain structures and evaluating the corresponding outputs of these sets.

We also propose another package of statistical tests which are designed based on certain cryptographic properties of block ciphers and hash functions method to evaluate their randomness, namely the cryptographic randomness testing. Throughout the work, we adapt SAC Test, Linear Span Test and Coverage Test, which were previously proposed for stream ciphers, to test the round functions of block ciphers and compression functions of hash functions. Also, we adapt Collision Test, which was originally proposed by Knuth for testing sequences, to test algorithms. Afterwards, we apply the package to AES finalists and we observed that the number of rounds where the randomness is achieved for MARS, RC6 and Rijndael is more precise than the previous results. We also apply the package to SHA-3 second round candidate algorithms.

The contributions of the thesis are as follows:

- We propose two new test suites to evaluate block ciphers and hash functions statistically.
- We propose an alternative evaluation method for generators which produce short sequences.

- We calculate the exact distributions for statistical randomness tests which produce reliable results for short sequences and calculate the subinterval probabilities for the new evaluation method.
- We propose 7 new statistical randomness tests (Periodic Frequency Test, Linear Complexity Profile Test, Rotation Test, Integer Frequency Test, Maximum Difference Test, Repeating Point Test, Saturation Point Test) for short sequences.
- We propose a method to classify the statistical randomness tests.

REFERENCES

[1] M. S. Turan. *On Statistical Analysis of Synchronous Stream Ciphers*. PhD thesis, Middle East Technical University, Ankara, Turkey, April 2008.

[2] I. Goldberg and D. Wagner. Randomness and the netscape browser. In *Dr. Dobb's Journal*, pages 66–70, 1996.

[3] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001.

[4] D. E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison-Wesley, 1981.

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

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

[7] 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. 2001.

[8] 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.

[9] 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.

[10] Meidl W. and Niederreiter H. Counting functions and expected values for the k-error linear complexity. *Finite Fields and Their Applications*, 8:142–154(13), April 2002.

[11] A. Doğanaksoy and F. Göloğlu. On lempel-ziv complexity of sequences. In Hong-Yeop Song Guang Gong, Tor Helleseth and Kyeongcheol Yang, editors, *SETA*, volume 4086 of *Lecture Notes in Computer Science*, pages 180–189. Springer, 2006.

[12] A. Doğanaksoy, Ç. Çalık, F. Sulak, and M. S. Turan. New randomness tests using random walk. In *National Cryptology Symposium II, TURKEY*, 2006.

[13] A. F. Webster and S. E. Tavares. On the design of s-boxes. In *Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85*, pages 523–534, New York, NY, USA, 1986. Springer-Verlag New York, Inc.

[14] M. Sönmez Turan, Ç. Çalık, N. Buz Saran, and A. Doğanaksoy. New distinguishers based on random mappings against stream ciphers. In Solomon W. Golomb, Matthew G. Parker, Alexander Pott, and Arne Winterhof, editors, *SETA*, volume 5203 of *Lecture Notes in Computer Science*, pages 30–41. Springer, 2008.

[15] Carolynn Burwick, Don Coppersmith, Edward D'Avignon, Rosario Gennaro, Shai Halevi, Charanjit Jutla, Stephen M. Matyas Jr, Luke O'Connor, Mohammad Peyravian, Jr. Luke, O'connor Mohammad Peyravian, David Stafford, and Nevenko Zunic. Mars - a candidate cipher for aes. *NIST AES Proposal*, 1999.

[16] Block Cipher, Ronald L. Rivest, M. J. B. Robshaw, Y.L. Yin, and R. Sidney. The rc6 block cipher, 1998.

[17] J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.

[18] Eli Biham, Ross Anderson, and Lars Knudsen. Serpent: A new block cipher proposal. In *In Fast Software Encryption 98*, pages 222–238. Springer-Verlag, 1998.

[19] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. Twofish: A 128-bit block cipher. In *in First Advanced Encryption Standard (AES) Conference*, 1998.

[20] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C.-W. Phan. Sha-3 proposal blake. Submission to NIST, 2008.

[21] Danilo Gligoroski, Vlastimil Klima, Svein Johan Knapskog, Mohamed El-Hadedy, Jorn Amundsen, and Stig Frode Mjolsnes. Cryptographic hash function blue midnight wish. Submission to NIST (Round 2), 2009.

[22] Daniel J. Bernstein. Cubehash specification (2.b.1). Submission to NIST (Round 2), 2009.

[23] Ryad Benadjila, Olivier Billet, Henri Gilbert, Gilles Macario-Rat, Thomas Peyrin, Matt Robshaw, and Yannick Seurin. Sha-3 proposal: Echo. Submission to NIST (updated), 2009.

[24] Shai Halevi, William E. Hall, and Charanjit S. Jutla. The hash function fugue. Submission to NIST (updated), 2009.

[25] Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. Grøstl – a sha-3 candidate. Submission to NIST, 2008.

[26] Özgül Küçük. The hash function hamsi. Submission to NIST (updated), 2009.

[27] Hongjun Wu. The hash function jh. Submission to NIST (updated), 2009.

[28] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Keccak specifications. Submission to NIST (Round 2), 2009.

[29] Christophe De Canniere, Hisayoshi Sato, and Dai Watanabe. Hash function luffa: Specification. Submission to NIST (Round 2), 2009.

[30] Emmanuel Bresson, Anne Canteaut, Benoît Chevallier-Mames, Christophe Clavier, Thomas Fuhr, Aline Gouget, Thomas Icart, Jean-François Misarsky, Marìa Naya-Plasencia, Pascal Paillier, Thomas Pornin, Jean-René Reinhard, Céline Thuillet, and Marion Videau. Shabal, a submission to nist cryptographic hash algorithm competition. Submission to NIST, 2008.

[31] Eli Biham and Orr Dunkelman. The shavite-3 hash function. Submission to NIST (Round 2), 2009.

[32] Gaëtan Leurent, Charles Bouillaguet, and Pierre-Alain Fouque. Simd is a message digest. Submission to NIST (Round 2), 2009.

[33] Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, and Jesse Walker. The skein hash function family. Submission to NIST (Round 2), 2009.

Appendix A

Subinterval Probabilities

Table A.1: Subinterval Probabilities for Frequency Test, Frequency Test within a Block, Periodic Frequency Test, and Runs Test

p-value	Freq	Bl Freq	Per Freq	Runs
0,0 - 0,1	0,091312	0,099498	0,099498	0,099315
0,1 - 0,2	0,097936	0,097137	0,097137	0,099622
0,2 - 0,3	0,098741	0,100424	0,100424	0,100594
0,3 - 0,4	0,128568	0,109070	0,109070	0,110248
0,4 - 0,5	0,075286	0,101255	0,101255	0,089509
0,5 - 0,6	0,082019	0,095544	0,095544	0,096585
0,6 - 0,7	0,087971	0,097023	0,097023	0,095026
0,7 - 0,8	0,092898	0,106247	0,106247	0,106260
0,8 - 0,9	0,096584	0,100846	0,100846	0,102243
0,9 - 1,0	0,148685	0,092955	0,092955	0,100598

Table A.2: Subinterval Probabilities for Test for the Longest Run of Ones in a Block, Serial 1 Test, and Approximate Entropy Test

p-value	Lon Run	Serial 1	App Ent
0,0 - 0,1	0,094347	0,097450	0,099245
0,1 - 0,2	0,104510	0,098970	0,097256
0,2 - 0,3	0,105749	0,099146	0,102839
0,3 - 0,4	0,101619	0,091051	0,103637
0,4 - 0,5	0,090870	0,114303	0,098356
0,5 - 0,6	0,112755	0,093207	0,098838
0,6 - 0,7	0,096412	0,094892	0,096122
0,7 - 0,8	0,097298	0,118739	0,103614
0,8 - 0,9	0,098344	0,093366	0,101217
0,9 - 1,0	0,098095	0,098876	0,098876

Table A.3: Subinterval Probabilities for Rotation Test, Cumulative Sums Test, and Integer Frequency Test

p-value	Rotation	Cu Sums	Int Freq
0,0 - 0,1	0,091312	0,091424	0,100618
0,1 - 0,2	0,097936	0,091200	0,093300
0,2 - 0,3	0,098741	0,085796	0,107104
0,3 - 0,4	0,128568	0,109924	0,104243
0,4 - 0,5	0,075286	0,090897	0,101561
0,5 - 0,6	0,082019	0,103963	0,090387
0,6 - 0,7	0,087971	0,114207	0,114157
0,7 - 0,8	0,092898	0,060357	0,089471
0,8 - 0,9	0,096584	0,111896	0,094973
0,9 - 1,0	0,148685	0,140337	0,104186

Table A.4: Subinterval Probabilities for Non-overlapping Template Matching Test, and Linear Complexity Test

Nonover Temp			Lin Comp		
Obs Min	Obs Max	Prob	Obs Min	Obs Max	Prob
0	11	0,064163	1	126	0,083333
12	12	0,056124	127	127	0,250000
13	13	0,082799	128	128	0,500000
14	14	0,108476	129	129	0,125000
15	15	0,126699	130	256	0,041667
16	16	0,132354			
17	17	0,123983			
18	18	0,104367			
19	19	0,079080			
20	256	0,121954			

Table A.5: Subinterval Probabilities for Linear Complexity Profile Test, and Maximum of t Test

Lin Comp Pro			Maximum		
Obs Min	Obs Max	Prob	Obs Min	Obs Max	Prob
0	57	0,095660	0	933	0,100270
58	60	0,124544	934	959	0,098926
61	62	0,118769	960	975	0,101928
63	64	0,137401	976	986	0,097377
65	65	0,070877	987	995	0,101515
66	66	0,069403	996	1002	0,095679
67	68	0,126403	1003	1009	0,113124
69	70	0,100505	1010	1014	0,093140
71	72	0,070485	1015	1019	0,104828
73	128	0,085952	1020	1023	0,093213

Table A.6: Subinterval Probabilities for Maximum Difference Test and Repeating Point Test

Max Differ			Rep Point		
Obs Min	Obs Max	Prob	Obs Min	Obs Max	Prob
0	226	0,10442	2	4	0,091087
227	233	0,11068	5	6	0,123378
234	237	0,10026	7	7	0,073644
238	240	0,09718	8	8	0,077863
241	243	0,11674	9	9	0,079253
244	245	0,08744	10	10	0,078015
246	247	0,0926	11	12	0,143633
248	249	0,09357	13	14	0,117440
250	251	0,08766	15	17	0,118928
252	255	0,10945	18	> 42	0,096759

Table A.7: Subinterval Probabilities for Coupon Collector Test and Saturation Point Test

Coup Coll			Sat Point		
Obs Min	Obs Max	Prob	Obs Min	Obs Max	Prob
1	28	0,125799	16	38	0,193609
29	29	0,120168	39	45	0,179686
30	30	0,164722	46	53	0,196007
31	31	0,183675	54	64	0,195881
32	32	0,165988	> 64		0,234818
33	33	0,120914			
34	42	0,118734			

Appendix B

P-Value Tables

Table B.1: *P*-Value Table for the Linear Complexity Profile Test

T-value	p-value	T-value	p-value
140	0,000000	68	0,634981
41	0,000026	69	0,513884
42	0,000058	70	0,405999
43	0,000125	71	0,312874
44	0,000260	72	0,234996
45	0,000521	73	0,171904
46	0,001011	74	0,122393
47	0,001899	75	0,084763
48	0,003454	76	0,057066
49	0,006085	77	0,037329
50	0,010390	78	0,023712
51	0,017203	79	0,014619
52	0,027636	80	0,008744
53	0,043096	81	0,005071
54	0,065269	82	0,002850
55	0,096054	83	0,001552
56	0,137440	84	0,000818
57	0,191320	85	0,000417
58	0,259256	86	0,000206
59	0,342232	87	0,000098
60	0,440408	88	0,000045
61	0,552950	89	0,000020
62	0,677947	90	0,000009
63	0,812468	91	0,000004
64	0,952748	92	0,000001
65	1,000000	93	0,000001
66	0,905497	94-128	0,000000
67	0,766690		

Table B.2: *P*-Value Table for the Maximum of *t* Test

T-value	p-value	T-value	p-value	T-value	p-value	T-value	p-value	T-value	p-value	T-value	p-value
1-556	0,000000	634	0,000013	712	0,000235	790	0,003147	868	0,033039	946	0,283322
557	0,000001	635	0,000013	713	0,000243	791	0,003248	869	0,034003	947	0,290897
558	0,000001	636	0,000014	714	0,000252	792	0,003352	870	0,034993	948	0,298666
559	0,000001	637	0,000015	715	0,000261	793	0,003460	871	0,036012	949	0,306635
560	0,000001	638	0,000015	716	0,000270	794	0,003570	872	0,037058	950	0,314807
561	0,000001	639	0,000016	717	0,000280	795	0,003684	873	0,038134	951	0,323188
562	0,000001	640	0,000016	718	0,000290	796	0,003802	874	0,039240	952	0,331783
563	0,000001	641	0,000017	719	0,000300	797	0,003923	875	0,040377	953	0,340597
564	0,000001	642	0,000018	720	0,000310	798	0,004048	876	0,041545	954	0,349635
565	0,000001	643	0,000018	721	0,000321	799	0,004176	877	0,042746	955	0,358904
566	0,000001	644	0,000019	722	0,000333	800	0,004309	878	0,043980	956	0,368408
567	0,000001	645	0,000020	723	0,000344	801	0,004445	879	0,045248	957	0,378154
568	0,000001	646	0,000021	724	0,000356	802	0,004586	880	0,046551	958	0,388147
569	0,000001	647	0,000022	725	0,000369	803	0,004731	881	0,047890	959	0,398393
570	0,000001	648	0,000022	726	0,000382	804	0,004880	882	0,049266	960	0,408899
571	0,000001	649	0,000023	727	0,000395	805	0,005034	883	0,050680	961	0,419670
572	0,000001	650	0,000024	728	0,000409	806	0,005192	884	0,052133	962	0,430713
573	0,000001	651	0,000025	729	0,000423	807	0,005356	885	0,053626	963	0,442035
574	0,000001	652	0,000026	730	0,000438	808	0,005524	886	0,055160	964	0,453643
575	0,000001	653	0,000027	731	0,000453	809	0,005697	887	0,056736	965	0,465542
576	0,000001	654	0,000028	732	0,000469	810	0,005876	888	0,058355	966	0,477741
577	0,000001	655	0,000029	733	0,000485	811	0,006059	889	0,060018	967	0,490247
578	0,000001	656	0,000030	734	0,000502	812	0,006249	890	0,061727	968	0,503067
579	0,000001	657	0,000032	735	0,000519	813	0,006444	891	0,063482	969	0,516208
580	0,000001	658	0,000033	736	0,000537	814	0,006645	892	0,065286	970	0,529678
581	0,000001	659	0,000034	737	0,000556	815	0,006851	893	0,067138	971	0,543485
582	0,000002	660	0,000035	738	0,000575	816	0,007065	894	0,069041	972	0,557638
583	0,000002	661	0,000037	739	0,000595	817	0,007284	895	0,070996	973	0,572144
584	0,000002	662	0,000038	740	0,000615	818	0,007510	896	0,073003	974	0,587011
585	0,000002	663	0,000040	741	0,000636	819	0,007742	897	0,075065	975	0,602250
586	0,000002	664	0,000041	742	0,000658	820	0,007982	898	0,077183	976	0,617867
587	0,000002	665	0,000043	743	0,000681	821	0,008229	899	0,079359	977	0,633873
588	0,000002	666	0,000044	744	0,000704	822	0,008483	900	0,081593	978	0,650277
589	0,000002	667	0,000046	745	0,000728	823	0,008744	901	0,083887	979	0,667088
590	0,000002	668	0,000048	746	0,000753	824	0,009013	902	0,086243	980	0,684315
591	0,000002	669	0,000050	747	0,000778	825	0,009290	903	0,088663	981	0,701970
592	0,000002	670	0,000051	748	0,000805	826	0,009576	904	0,091148	982	0,720061
593	0,000002	671	0,000053	749	0,000832	827	0,009869	905	0,093699	983	0,738599
594	0,000003	672	0,000055	750	0,000860	828	0,010172	906	0,096319	984	0,757595
595	0,000003	673	0,000058	751	0,000889	829	0,010483	907	0,099010	985	0,777059
596	0,000003	674	0,000060	752	0,000919	830	0,010803	908	0,101772	986	0,797003
597	0,000003	675	0,000062	753	0,000950	831	0,011133	909	0,104608	987	0,817438
598	0,000003	676	0,000064	754	0,000982	832	0,011472	910	0,107520	988	0,838375
599	0,000003	677	0,000067	755	0,001015	833	0,011822	911	0,110510	989	0,859827
600	0,000003	678	0,000069	756	0,001049	834	0,012181	912	0,113580	990	0,881805
601	0,000003	679	0,000072	757	0,001085	835	0,012551	913	0,116731	991	0,904322
602	0,000004	680	0,000075	758	0,001121	836	0,012932	914	0,119966	992	0,927390
603	0,000004	681	0,000077	759	0,001158	837	0,013324	915	0,123287	993	0,951022
604	0,000004	682	0,000080	760	0,001197	838	0,013727	916	0,126696	994	0,975232
605	0,000004	683	0,000083	761	0,001237	839	0,014142	917	0,130196	995	1,000000
606	0,000004	684	0,000086	762	0,001278	840	0,014569	918	0,133789	996	0,999967
607	0,000004	685	0,000089	763	0,001321	841	0,015008	919	0,137476	997	0,974561
608	0,000005	686	0,000093	764	0,001365	842	0,015460	920	0,141261	998	0,948536
609	0,000005	687	0,000096	765	0,001410	843	0,015925	921	0,145146	999	0,921877
610	0,000005	688	0,000100	766	0,001457	844	0,016404	922	0,149133	1000	0,894570
611	0,000005	689	0,000103	767	0,001505	845	0,016896	923	0,153225	1001	0,866601
612	0,000005	690	0,000107	768	0,001555	846	0,017403	924	0,157425	1002	0,837952
613	0,000006	691	0,000111	769	0,001606	847	0,017924	925	0,161736	1003	0,808609
614	0,000006	692	0,000115	770	0,001659	848	0,018460	926	0,166159	1004	0,778555
615	0,000006	693	0,000120	771	0,001714	849	0,019011	927	0,170699	1005	0,747774
616	0,000006	694	0,000124	772	0,001770	850	0,019578	928	0,175357	1006	0,716250
617	0,000007	695	0,000128	773	0,001828	851	0,020161	929	0,180138	1007	0,683964
618	0,000007	696	0,000133	774	0,001888	852	0,020761	930	0,185043	1008	0,650899
619	0,000007	697	0,000138	775	0,001950	853	0,021379	931	0,190077	1009	0,617038
620	0,000007	698	0,000143	776	0,002014	854	0,022013	932	0,195242	1010	0,582362
621	0,000008	699	0,000148	777	0,002080	855	0,022666	933	0,200541	1011	0,546852
622	0,000008	700	0,000154	778	0,002148	856	0,023337	934	0,205978	1012	0,510488
623	0,000008	701	0,000159	779	0,002218	857	0,024028	935	0,211557	1013	0,473253
624	0,000009	702	0,000165	780	0,002290	858	0,024738	936	0,217281	1014	0,435124
625	0,000009	703	0,000171	781	0,002364	859	0,025468	937	0,223153	1015	0,396082
626	0,000009	704	0,000177	782	0,002441	860	0,026219	938	0,229177	1016	0,356106
627	0,000010	705	0,000184	783	0,002520	861	0,026991	939	0,235357	1017	0,315175
628	0,000010	706	0,000190	784	0,002602	862	0,027784	940	0,241697	1018	0,273266
629	0,000011	707	0,000197	785	0,002686	863	0,028601	941	0,248201	1019	0,230357
630	0,000011	708	0,000204	786	0,002773	864	0,029440	942	0,254873	1020	0,186426
631	0,000012	709	0,000211	787	0,002862	865	0,030302	943	0,261716	1021	0,141448
632	0,000012	710	0,000219	788	0,002954	866	0,031190	944	0,268736	1022	0,095401
633	0,000012	711	0,000227	789	0,003049	867	0,032101	945	0,275937	1023	0,048260

Table B.3: *P*-Value Table for the Maximum Difference Test

T-value	p-value	T-value	p-value	T-value	p-value
1- 137	0,000000	177	0,000424	217	0,076346
138	0,000002	178	0,000488	218	0,085693
139	0,000002	179	0,000562	219	0,096100
140	0,000002	180	0,000647	220	0,107676
141	0,000002	181	0,000744	221	0,120536
142	0,000002	182	0,000856	222	0,134808
143	0,000002	183	0,000983	223	0,150624
144	0,000002	184	0,001129	224	0,168132
145	0,000004	185	0,001296	225	0,187484
146	0,000004	186	0,001486	226	0,208845
147	0,000004	187	0,001704	227	0,232388
148	0,000006	188	0,001952	228	0,258297
149	0,000006	189	0,002235	229	0,286762
150	0,000008	190	0,002558	230	0,317981
151	0,000009	191	0,002926	231	0,352158
152	0,000010	192	0,003344	232	0,389499
153	0,000012	193	0,003820	233	0,430211
154	0,000014	194	0,004361	234	0,474500
155	0,000016	195	0,004976	235	0,522563
156	0,000019	196	0,005673	236	0,574585
157	0,000022	197	0,006465	237	0,630731
158	0,000026	198	0,007362	238	0,691139
159	0,000030	199	0,008378	239	0,755911
160	0,000035	200	0,009529	240	0,825095
161	0,000041	201	0,010831	241	0,898681
162	0,000047	202	0,012302	242	0,976571
163	0,000055	203	0,013964	243	1,000000
164	0,000064	204	0,015840	244	0,941430
165	0,000074	205	0,017955	245	0,855645
166	0,000086	206	0,020339	246	0,766553
167	0,000100	207	0,023023	247	0,674821
168	0,000116	208	0,026044	248	0,581349
169	0,000134	209	0,029439	249	0,487308
170	0,000155	210	0,033252	250	0,394206
171	0,000179	211	0,037532	251	0,303944
172	0,000207	212	0,042330	252	0,218892
173	0,000239	213	0,047706	253	0,141982
174	0,000276	214	0,053722	254	0,076802
175	0,000319	215	0,060448	255	0,027716
176	0,000367	216	0,067962		

Table B.4: P-Value Table for the Coupon Collector Test

T-value	p-value	T-value	p-value
1	0,000000	22	0,000098
2	0,000000	23	0,000592
3	0,000000	24	0,002919
4	0,000000	25	0,011788
5	0,000000	26	0,039268
6	0,000000	27	0,108653
7	0,000000	28	0,251598
8	0,000000	29	0,491934
9	0,000000	30	0,821378
10	0,000000	31	1,000000
11	0,000000	32	0,811272
12	0,000000	33	0,479297
13	0,000000	34	0,237468
14	0,000000	35	0,096530
15	0,000000	36	0,031481
16	0,000000	37	0,008032
17	0,000000	38	0,001554
18	0,000000	39	0,000218
19	0,000000	40	0,000021
20	0,000001	41	0,000001
21	0,000013	42	0,000000

Table B.5: P-Value Table for the Repeating Point Test

T-value	p-value	T-value	p-value
2	0,031250	23	0,033175
3	0,092773	24	0,021771
4	0,182175	25	0,013947
5	0,295789	26	0,008717
6	0,428930	27	0,005312
7	0,576218	28	0,003154
8	0,731944	29	0,001823
9	0,890451	30	0,001026
10	1,000000	31	0,000561
11	0,953519	32	0,000298
12	0,804531	33	0,000154
13	0,666252	34	0,000077
14	0,541330	35	0,000037
15	0,431372	36	0,000017
16	0,337010	37	0,000008
17	0,258023	38	0,000003
18	0,193517	39	0,000001
19	0,142114	40	0,000001
20	0,102145	41	0,000000
21	0,071820	42	0,000000
22	0,049377	> 42	0,000000

Table B.6: *P*-Value Table for the Saturation Point Test

T-value	p-value	T-value	p-value
16	0,000002	41	0,536917
17	0,000019	42	0,588901
18	0,000089	43	0,641373
19	0,000292	44	0,694029
20	0,000774	45	0,746589
21	0,001752	46	0,798799
22	0,003522	47	0,850431
23	0,006442	48	0,901284
24	0,010921	49	0,951183
25	0,017385	50	0,999979
26	0,026255	51	1,000000
27	0,037918	52	0,952454
28	0,052704	53	0,906218
29	0,070868	54	0,861397
30	0,092580	55	0,818054
31	0,117922	56	0,776233
32	0,146887	57	0,735966
33	0,179384	58	0,697270
34	0,215250	59	0,660150
35	0,254260	60	0,624599
36	0,296138	61	0,590604
37	0,340570	62	0,558141
38	0,387218	63	0,527182
39	0,435729	64	0,497693
40	0,485746	> 64	0,469636

Appendix C

Statistical Randomness Test Results

Table C.1: Statistical Randomness Test Results for AES Finalist Algorithms

MARS								
Rnds	Freq.	B.Freq.	Run	L.Run	Ap.En.	C.Sum1	C.Sum2	Serial
1	0.8698	0.3289	0.5914	0.8848	0.1702	0.3659	0.2129	0.5484
2	0.4330	0.3106	0.4138	0.4819	0.3953	0.3474	0.0906	0.3304
3	0.3328	0.9079	0.5933	0.3932	0.1405	0.2391	0.2145	0.1942
RC6								
Rnds	Freq.	B.Freq.	Run	L.Run	Ap.En.	C.Sum1	C.Sum2	Serial
3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.2504	0.2245	0.0371	0.1145	0.1413	0.0755	0.9803	0.1453
5	0.6958	0.7972	0.9787	0.1823	0.6092	0.9830	0.5004	0.3810
Rijndael								
Rnds	Freq.	B.Freq.	Run	L.Run	Ap.En.	C.Sum1	C.Sum2	Serial
2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.7717	0.4658	0.4507	0.3248	0.7001	0.1536	0.6747	0.8385
4	0.2878	0.5585	0.2322	0.3352	0.3156	0.8272	0.3832	0.6447
Serpent								
Rnds	Freq.	B.Freq.	Run	L.Run	Ap.En.	C.Sum1	C.Sum2	Serial
1	0.0063	0.0000	0.1820	0.5037	0.5924	0.0000	0.0000	0.3469
2	0.3487	0.8226	0.0463	0.0599	0.5569	0.5131	0.6117	0.4858
3	0.3421	0.4239	0.5364	0.6143	0.4860	0.5027	0.4299	0.2736
Twofish								
Rnds	Freq.	B.Freq.	Run	L.Run	Ap.En.	C.Sum1	C.Sum2	Serial
2	0.1557	0.4578	0.0000	0.0157	0.0000	0.0575	0.0009	0.0000
4	0.4383	0.7038	0.9565	0.7466	0.7321	0.4159	0.9779	0.6924
6	0.3933	0.0131	0.8354	0.8263	0.3163	0.0109	0.6482	0.2544

Appendix D

Cryptographic Randomness Test Results

Table D.1: Cryptographic Randomness Test Results for AES Finalist Algorithms

MARS						
Rnds	SAC	Lin Span	Coll 16	Coll 20	Cov 12	Cov 14
4	0,000000	-	0,855437	0,269610	0,230847	0,345577
5	0,000000	-	-	-	-	-
6	0,684260	-	-	-	-	-
RC6						
Rnds	SAC	Lin Span	Coll 16	Coll 20	Cov 12	Cov 14
4	0,000000	-	0,000000	0,000000	0,000000	0,000000
5	0,422925	-	0,954115	0,550493	0,233968	0,500533
6	0,117274	-	0,099383	0,048752	0,213578	0,821564
Rijndael						
Rnds	SAC	Lin Span	Coll 16	Coll 20	Cov 12	Cov 14
2	0,000000	0,558167	0,000000	0,000000	0,000000	0,000000
3	0,000000	0,473152	0,107429	0,933749	0,179152	0,519714
4	0,680614	-	0,198314	0,806697	0,163467	0,945009
Serpent						
Rnds	SAC	Lin Span	Coll 16	Coll 20	Cov 12	Cov 14
3	0,000000	0,051753	0,000000	0,000000	0,000000	0,000000
4	0,980746	-	0,740121	0,764551	0,562332	0,480743
5	0,715258	-	0,740121	0,298605	0,723684	0,880793
Twofish						
Rnds	SAC	Lin Span	Coll 16	Coll 20	Cov 12	Cov 14
2	0,000000	0,848862	0,000000	0,000000	0,000000	0,000000
4	0,850661	-	0,167473	0,678987	0,272416	0,951575
6	0,907163	-	0,955974	0,158482	0,891265	0,448653

VITA

Fatih Sulak was born in Kulu, on April 16, 1979. He received

- his BSc degree in Electrical and Electronics Engineering from Middle East Technical University (METU) on July 2003,
- a double major degree in Mathematics Department from METU on July 2003,
- his MSc degree in Cryptography Department of METU in January 2006 with the thesis title 'Constructions of Bent Functions',
- his PhD degree in Cryptography Department of METU in February 2011.

He had worked as a teaching assistant in Mathematics Department, METU between 2005 and 2010. His research interests are design and analysis of symmetric cryptosystems, and statistical randomness testing. He received a silver medal from International Mathematical Olympiad in 1997.

List of Publications

- 'Evaluation of Randomness Test Results for Short Sequences', F. Sulak , A. Doğanaksoy, B. Ege, O. Koçak, Sequences and Their Applications, 2010.
- 'Statistical Testing of Some SHA-3 Candidates', A. Doğanaksoy, B. Ege, O. Koçak, F. Sulak, 4th International Information Security and Cryptology Conference, 2010.
- 'A Survey of the Attacks on AES', A. Doğanaksoy, A. Darbuka, D. Özberk, N. Öztop, F. Sulak, 3rd International Information Security and Cryptology Conference, 2008.
- 'A Survey of the Related-Key Attacks on AES', A. Doğanaksoy, A. Darbuka, D. Özberk, N. Öztop, F. Sulak, 3rd International Information Security and Cryptology Conference, 2008.

- ‘Cryptanalysis of the Dedicated Hash Functions’, A. Doğanaksoy, O. Özen, F. Sulak, K. Varıcı, E. Yüce, 3rd International Information Security and Cryptology Conference, 2007.

- ‘Observations on Hellman’s Cryptanalytic Time-Memory Trade-off’, A. Doğanaksoy, Ç. Çalık, F. Sulak, 2nd Turkish National Cryptology Conference, 2006.

- ‘New Randomness Tests Using Random Walk’, A. Doğanaksoy, Ç. Çalık, F. Sulak, M. S. Turan, 2nd Turkish National Cryptology Conference, 2006.

- ‘A Survey on Bent Functions and Normality’, A. Doğanaksoy, B. G. Dündar, F. Göloğlu, Z. Saygı, F. Sulak, M. Uğuz, 2nd Turkish National Cryptology Conference, 2006.

- ‘Constructions of Highly Nonlinear Balanced Boolean Functions’, A. Doğanaksoy, B. G. Dündar, F. Göloğlu, Z. Saygı, F. Sulak, M. Uğuz, 1st Turkish National Cryptology Conference, 2005.