12-2-2016

Implementation of the Array-based Approach for the Evaluation of Randomness of the Advanced Encryption Standard (AES)

Hanadi Yahia Alomari
hya8143@rit.edu

ROCHESTER INSTITUTE OF TECHNOLOGY

Implementation of the Array-based Approach for the Evaluation of Randomness of the Advanced Encryption Standard (AES)

By

Hanadi Yahia Alomari

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Applied Statistics

in the

School of Mathematical Sciences

College of Science

December 2, 2016

Declaration of Authorship

I, Hanadi Alomari, declare that this thesis titled, “Implementation of the Array-based Approach for the Evaluation of Randomness of the Advanced Encryption Standard (AES)” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

_____

Date:

_____

Committee Approval

_____

Dr. Peter Bajorski Date
Thesis Advisor

_____

Dr. Alan Kaminsky Date
Committee Member

_____

Dr. Marcin Łukowiak Date
Committee Member

ROCHESTER INSTITUTE OF TECHNOLOGY

Abstract

Implementation of the Array-based Approach for the Evaluation of Randomness of the Advanced Encryption Standard (AES)

by Hanadi Yahia Alomari

Randomness is critical when it comes to cryptography. When a cryptographic system has randomness in it, it is impossible to decrypt the codes and break the system. To evaluate a cryptographic function's randomness, many statistical tests were developed such as: Diehard and ENT, but these tests are not well suited for block ciphers. CryptoStat is the best statistical test suite to test the randomness of the Advanced Encryption Standard or AES, which is a block cipher established by NIST, the National Institute of Standard and Technology. An array based approach is also used to apply the CryptoStat test suite on AES.

Acknowledgements

I would first like to thank my thesis advisor Dr. Peter Bajorski for his tremendous support and help. I cannot thank him enough because I really learned a lot from him, and this thesis would not be materialized without his encouragement.

I wish to thank Dr. Alan Kaminsky and Dr. Marcin Lukowiak for their acceptance of being in my thesis committee. I also would like to thank Dr. Kaminsky for providing me with the data used in this thesis. Moreover, I am thankful for his previous work on the topic, which helped me to learn and understand more.

Finally, I would like to thank my parents and siblings for their endless love. I would not be here without their encouragement. Also, many thanks to my lovely husband for his patience and support.

Table of Contents

List of Figures

List of Tables

Chapter 1

1.1. Introduction

Using computers can be extremely unsafe when using personal and financial data such as names, passwords, addresses, and bank account information. To keep one's safety, some aspects and applications can be applied. One of these aspects is Cryptography, which is the most popular tool for keeping private information safe from the public or third parties. Cryptography ensures that information is accessible by authorized people only, which keep their privacy. Cryptography also helps with Authentication because it facilitates proofing a remote user's identity, which prevents third parties from pretending to be someone else. Moreover, it could be used to provide a means to secure data that is not viewed or altered during storage or transaction, which is referred to as Integrity (Dent & Mitchell, 2005). Cryptography also could be used for Non-repudiation, which prevents the denial of previous actions. This "Non-repudiation" term is popular in financial and e-commerce applications. For example, a customer who requested a money transformation from his bank account could claim later that he never made that request and demand that money be refunded to his account. In such a situation, cryptographic tools are used to prove that the customer has requested and authorized the money transaction(Dent & Mitchell, 2005).

To have an effective cryptographic system, it is crucial to have some randomness in it because randomness makes it impossible for someone to break the system and reach the private

information. The objective of this thesis is to apply a CryptoStat test suite to evaluate a cryptographic system's randomness using an array based approach and statistical hypothesis testing.

1.2. Thesis Organization

This thesis consists of 8 chapters and is organized as follows. Chapter 2 will be mostly an overview of randomness in cryptography. The third chapter represents the history and the structure of the data that is used here. Since the goal of this thesis is to investigate the randomness in this data, chapter 4 introduces some previous tests that were used for this purpose. Chapter 5 explains a new array based approach to be used in evaluating the randomness. Also, a CryptoStat test suite represented in chapter 6 will be applied on the data, and the results will be discussed in chapter 7.

Chapter 2

2.1. Cryptography Overview

"Cryptography is the science of secret writing with the goal of hiding the meaning of the message" (Paar & Pelzl, 2010). Figure 1 shows the basic idea of Cryptography. It starts with the unencrypted data which we call "Plaintext." Plaintext is encrypted into "Ciphertext," which will be decrypted later into the original plaintext (*Kessler*, 2015). Encryption is the process of disguising a message in such a way as to hide its substance while Decryption is the process of returning an encrypted message back into plaintext (Matthews, 2003).

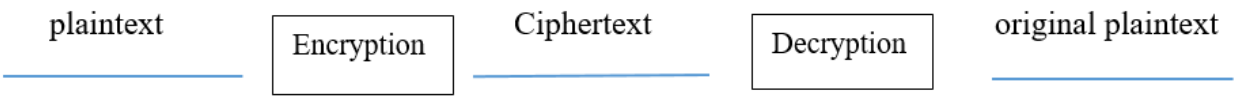

Figure 1: Cryptography basic idea

To clarify, let us say there are two people, Alex and Richard, and they want to communicate through a secure channel (Internet connection or telephone line). On the other hand, there is Jack, the bad guy here who could somehow interrupt the communication between Alex and Richard. When Alex sends the information (the plaintext) to Richard, he transforms the plaintext using the encryption function into codes (ciphertext) that go through the channel.

Although Jack can get the ciphertext by hacking the channel, he will not be able to understand what he received because he does not have the key to decrypt the ciphertext. While Richard, who knows the encryption key, will be able to decrypt the ciphertext, and get the plaintext that Alex has sent. This idea is shown in figure 2.

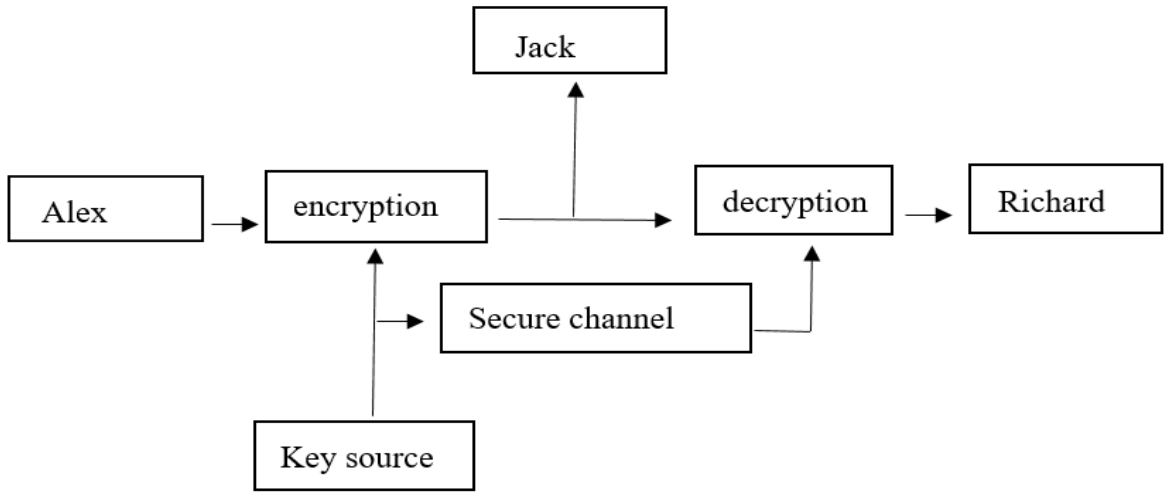

Figure 2: Cryptography process

2.2. Randomness Definition

A numeric sequence is considered to be statistically random when each number in this sequence is obtained completely randomly and has absolutely no correlation between any numbers in the sequence (Hörmanseder & Erik, 2007). These numbers in the sequences also have to be unpredictable and have no pattern or regularities to be considered as completely random (Boutin, 2005). A random bit sequence was also interpreted in (Rukhin & Bassham, 2008) as the result of flipping an unbiased coin with the sides “0” and “1.” The flips should be independent, so each flip is not affected by the previous one. Also, each flip has a probability of

½ to produce ether “0” or “1.” This unbiased coin is considered a perfect random generator since the “0” and “1” values are randomly distributed and [0,1] uniformly distributed.

2.3. Randomness Importance

Randomness is important in cryptography, and we expect to see in any ciphertext. Randomness can be found in stream ciphers, encryption keys, and in producing number sequences, etc. For instance, if the encryption key generates completely random mapping, the cryptographic system would be strongly secured. Having some pattern in a cryptographic system means that decrypting the ciphertext and attacking the system requires much less effort than attacking a system that has an unpredictable, completely random encryption key (Goldberg & Wagner, 1996). Therefore, more randomness in the encryption key means more strength in the cryptographic algorithm and more difficulty to break the system. In addition, randomness is an important resource for solving computational problems. Randomized algorithms and protocols play key roles in cryptography. These algorithms are typically faster and simpler than the deterministic counterparts (Rao, 2007).

2.4. Verifying Randomness in a Cryptographic System

In cryptography, secret keys must be chosen at random and even the cryptographic algorithms must be random as well because the most obvious way to reach randomness is to have access to two or more independent sources of randomness. On the other hand, a randomness principle exists in the fact that evidence of randomness in the outputs is not proof of

randomness of the cryptographic function itself. For ease of description, let us consider the basic cryptographic function that can be written as:

$$Ciphertext\ Output = f(key, plaintext) = plaintext \quad (1)$$

If random plaintexts and keys are inputted into the function, the outputs should be random as well (Hoyt, 2016). So, in order to have a strong cryptographic system, beside testing the randomness of their outputs, block ciphers and cryptographic functions should pass randomness tests too (Doganaksoy, Ege, Kocak, & Sulak, 2010). This is because testing randomness of outputs from a random selection of inputs is inefficient. When applying randomness tests, two possible outcomes would appear:

- Random outputs which are still do not prove the randomness of the cryptographic function which will be discussed later.
- Non-Random outputs which prove deficiency of the cryptographic function.

To conclude, non-random configurations of both keys and plaintexts should be tested (Hoyt, 2016).

Chapter 3

3.1. Data

The AES (Advanced Encryption Standard) (NIST, 2001) is a symmetric block cipher designed by Joan Daemen and Vincent Rijmen, which was approved above 15 other candidates by the National Institute of Standard and Technology (www.nist.gov). The AES block cipher has a block size of 128 bits and a key size of 128, 192, or 256 bits. However, only the 128-bit key version has been analyzed in this work. An AES dataset used in this thesis, which was computed by Dr. Alan Kaminsky using the AES-128 block cipher. It includes A (plaintexts) and B (encryption keys) inputs, and C matrix as the ciphertext output that consists of four main operations: 129 rows, 129 columns, 10 rounds or repeats, and 128 bytes (block cipher). The AES is considered a block cipher, which means that the scheme uses the cipher key to encrypt one block of data at a time (*Kessler*, 2015). The overall transformation process consists of repeated steps called “rounds” (Wright, 2001). For that reason, the AES dataset was produced using an iterated block cipher, meaning that the initial input block and cipher key undergoes multiple rounds of transformation before producing the output (*Kessler*, 2015).

Chapter 4

4.1. NIST Statistical Test Suite

To determine if a cryptographic function is considered random, many test suites can be applied. There are many statistical tests such as NIST (Rukhin & Bassham, 2008), Diehard (Marsaglia, 1995), Dieharder (Brown, Eddelbuettel, & Bauer, 2016), ENT (Walker, 2008), and TestU01 (L'Ecuyer & Simard, 2007). The NIST statistical test suite is a statistical package which contains 16 tests that were developed to determine whether a binary number sequence is random or not by detecting non- randomness types that may exist in the sequence. These tests analyze samples from the cryptographic outputs, and decide on the output randomness depending on the behavior of the samples selected.

4.2. NIST Statistical Test Suites and Block Ciphers

In spite of the fact that its title asserts that the NIST test suite is for cryptographic applications, the NIST is actually a statistical test suite for assessing binary sequences' randomness rather than being a statistical test on sequences of uniformly distributed numbers (Kaminsky, 2013). NIST test suite and the other test suites mentioned earlier are not well suited to evaluate block ciphers randomness. That is because to apply the NIST test suite (or another test suite), the block cipher must behave as a pseudo-random number generator (PRNG) and generate arbitrarily long binary sequences. Although, block ciphers are not PRNGs, a document from NIST (Soto & Bassham, 2000) describes how NIST made AES generates long binary sequences

using two techniques that include the encryption modes cipher block chaining (CBC) and electronic codebook (ECB). However, even after a block cipher is turned into a PRNG, the NIST test suite would evaluate the randomness of the PRNG not the randomness of the block cipher mapping directly (Kaminsky & Sorrell, 2013). Thus, these test suites are inefficient for the AES or any other block cipher because they only focus on the randomness of the outputs without considering the cryptographic functions' randomness, which is essential in the effectiveness of the cryptographic system. For example, a cryptographic system could pass these randomness test suites because it has random outputs that occur as a combination of random plaintexts and non-random encryption key, for instance.

Chapter 5

5.1. Applying an Array Based Approach on AES

To verify a cryptographic function's randomness, we should test the randomness of its parts. Using an array based tests is the best way to reach the randomness of both plaintexts and encryption keys. The array based approach(Hoyt, 2016) is using the structure of an array with three dimensions, which are: plaintexts, encryption keys, and encryption bits to apply any statistical test on the cryptographic algorithm. Then, any combination of these three dimensions is tested to determine which source is causing non-randomness in the outputs. This way the cryptographic function's randomness is tested instead of the outputs' randomness without specifying the exact reason for the non-randomness issues.

When performing a test on AES using the array based approach, a statistical hypothesis testing will be used. The null hypothesis H_0 assumes that $P = P_0$, where P represents the proportion of 1's in that sample and $P_0 = 2^{-1}$. On the other hand, H_1 is the alternative hypothesis, which assumes $P \neq P_0$. In addition, the Bonferroni correction is considered here since there are many statistical tests being performed. For each sample analyzed, a *P-value* is calculated. If the *P-value* > = α, then the hypothesis H_0 is accepted, and the cryptographic algorithm passes the test. On the contrary, if the *P-value* < α, then the hypothesis H_0 is rejected, and the cryptographic algorithm fails the test.

5.2. Normal Approximation to the Binomial Distribution

Although the binomial distribution is discrete unlike the normal distribution, when n, the sample size is large, the binomial distribution is approximately equal to the normal distribution, which is illustrated in the figure below. This concept is crucial since it becomes difficult to calculate binomial probabilities when n gets large. This occurs because the binomial distribution formula contains factorials which can be complicated in case of large sample sizes. Figure 3 below shows the binomial distribution and the normal distribution for a variety values of n.

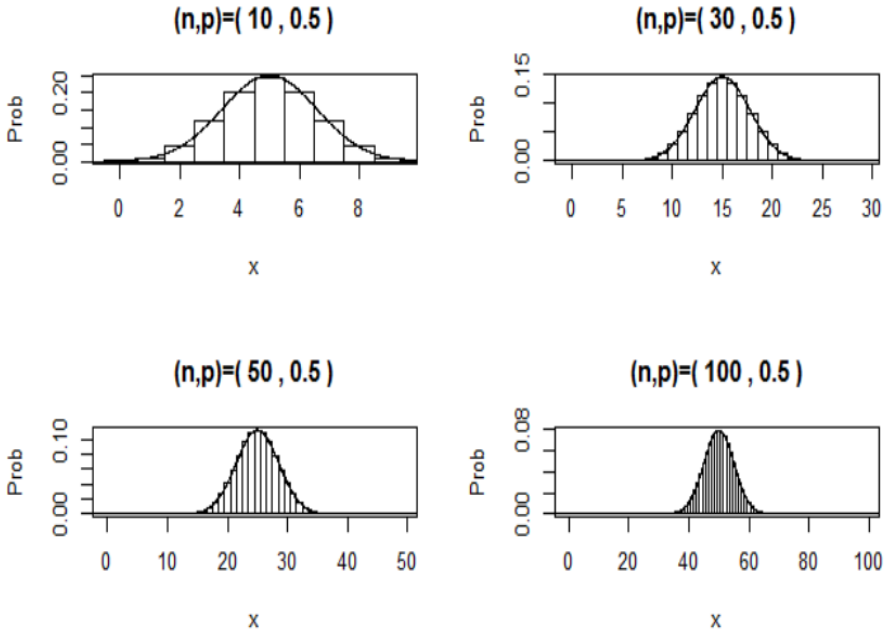

Figure 3: Normal approximation to the binomial

Normal approximation to the binomial distribution will be used in the statistical hypothesis testing applied on the block cipher AES. Considering k block of bits with n bits in each block and if in one block, the probability of 1 is p, and in the remaining blocks is p_0. Then, the fraction of 1's in all blocks approximately follows the normal distribution with the mean:

$$\mu = \frac{n \cdot p + (k-1) \cdot p_0}{n \cdot k} \qquad (2)$$

and variance:

$$\sigma^2 = \frac{n \cdot p \cdot (1-p) + (k-1) \cdot n \cdot p_0 \cdot (1-p_0)}{(n \cdot k)^2} \qquad (3)$$

5.3. Three Possible Ways to Use the Array Based Approach on the AES

First, we need to calculate the fraction of 1's in AES using the array based approach. If AES is random, the fraction of 1's should be 50%, which means the probability of 1 and 0 is the same, which is 0.5. Here is three ways to apply the array based approach:

- Test1: First test is done by counting all the 1's in the array, and then applying 1 binomial test.

Since AES dataset has 10 rounds, each round will be considered as a separate array, and we will have 10 binomial tests in this case. Table 1 below shows the number of times that the null hypothesis was rejected (p-value smaller than 0.05).

Round	Round1	Round2	Round3	Round4	Round5	Round6	Round7	Round8	Round9	Round10
<0.05	1	0	0	0	0	0	0	0	0	0

Table 1: Applying 1 binomial test on AES

- Test2: Second test is done by counting all the 1's in each column of the array, and then applying 129 binomial tests for each round.

The results below show that Round 1 has the highest failure rates with 92 failed binomial tests. On the other hand, there were no failure issues in the other rounds except for one case in Round 9.

Round	Round1	Round2	Round3	Round4	Round5	Round6	Round7	Round8	Round9	Round10
<0.05	92	0	0	0	0	0	0	0	1	0

Table 2: Applying 129 binomial tests

- Test 3: The third test is done by counting all the 1's in each cell, and then applying 129×129 binomial tests.

Using this approach will help to detect more failure binomial tests than the previous tests since it reaches each cell instead of each column or row. Table 3 shows that more than 11 thousand binomial tests have failed in the first round. On the other hand, round 9 has 129 failed binomial tests.

Round	Round1	Round2	Round3	Round4	Round5	Round6	Round7	Round8	Round9	Round10
<0.05	11868	0	0	0	0	0	0	0	129	0

Table 3: Applying 129×129 binomial tests

5.3.1. Investigating Rounds 1 and 9 Based on the Previous Results

The previous results illustrated that round 1 and round 9 has some issues. So, it is better to take a closer look at these rounds. Table 4 shows the number of failed binomial tests using rows, columns, and bits for all the rounds.

Round	Round1	Round2	Round3	Round4	Round5	Round6	Round7	Round8	Round9	Round10
Using rows	76	0	0	0	0	0	0	0	0	0
Using columns	92	0	0	0	0	0	0	0	1	0
Using bits	128	0	0	0	0	0	0	0	0	0

Table 4: Applying binomial tests on AES using rows, columns, and bits

To understand the nature of the randomness, figure 4 and 5 below show the frequencies of 1's in each ciphertext in round 1 and round 9. The points under the dashed lines fail the test. The dashed lines represent the $P_0 = 0.5 \pm z.value$.

Figure 4 shows the frequencies of 1's in round 1. The first plot in the figure shows plenty of points outside the dashed lines, which are the same 76 failure cases that were shown in table 4 earlier. The same situation in the second and third plots. On the other hand, figure 5 explains the situation in round 9. It shows the same results of table 4, which is one point that outside the dashed line and considered to be not random. This occurs when using columns to count the 1's and apply the binomial tests. Other points have proportion of 1's as 50% and considered random.

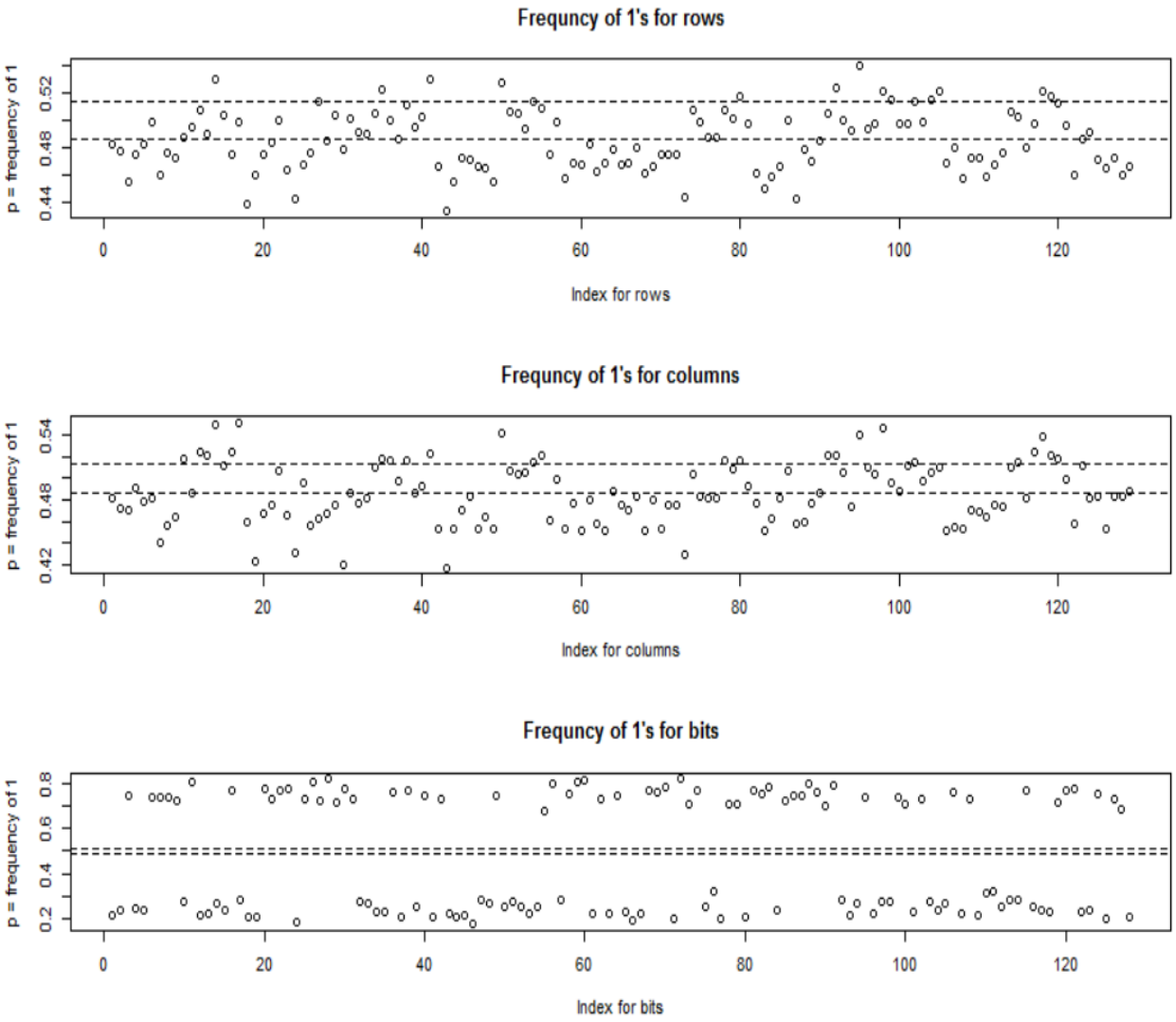

Figure 4: The frequencies of 1's in round 1

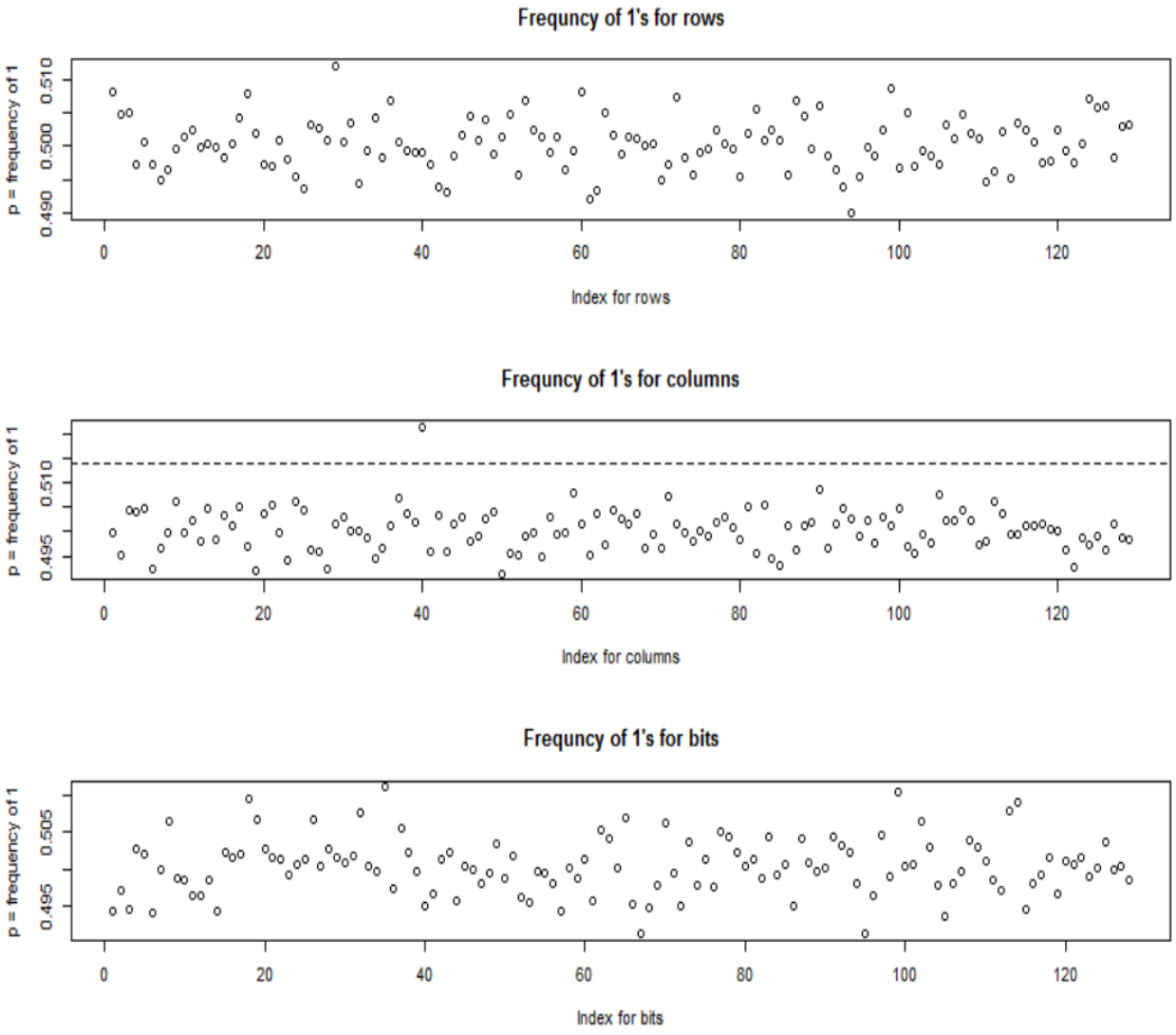

Figure 5: The frequencies of 1's in round 9

Figure 6 and 7 below display the P-values of the proportion of 1's in each ciphertext. Figure 6 shows lots of non-randomness issues for the first round. Especially in the third plot when using bits, all the 128 bits are under the significance level, which means the probabilities of 1's are less than 0.5. On the other hand, figure 7 shows only 1 point under the dashed line. The dashed line presents the level of significance or α.

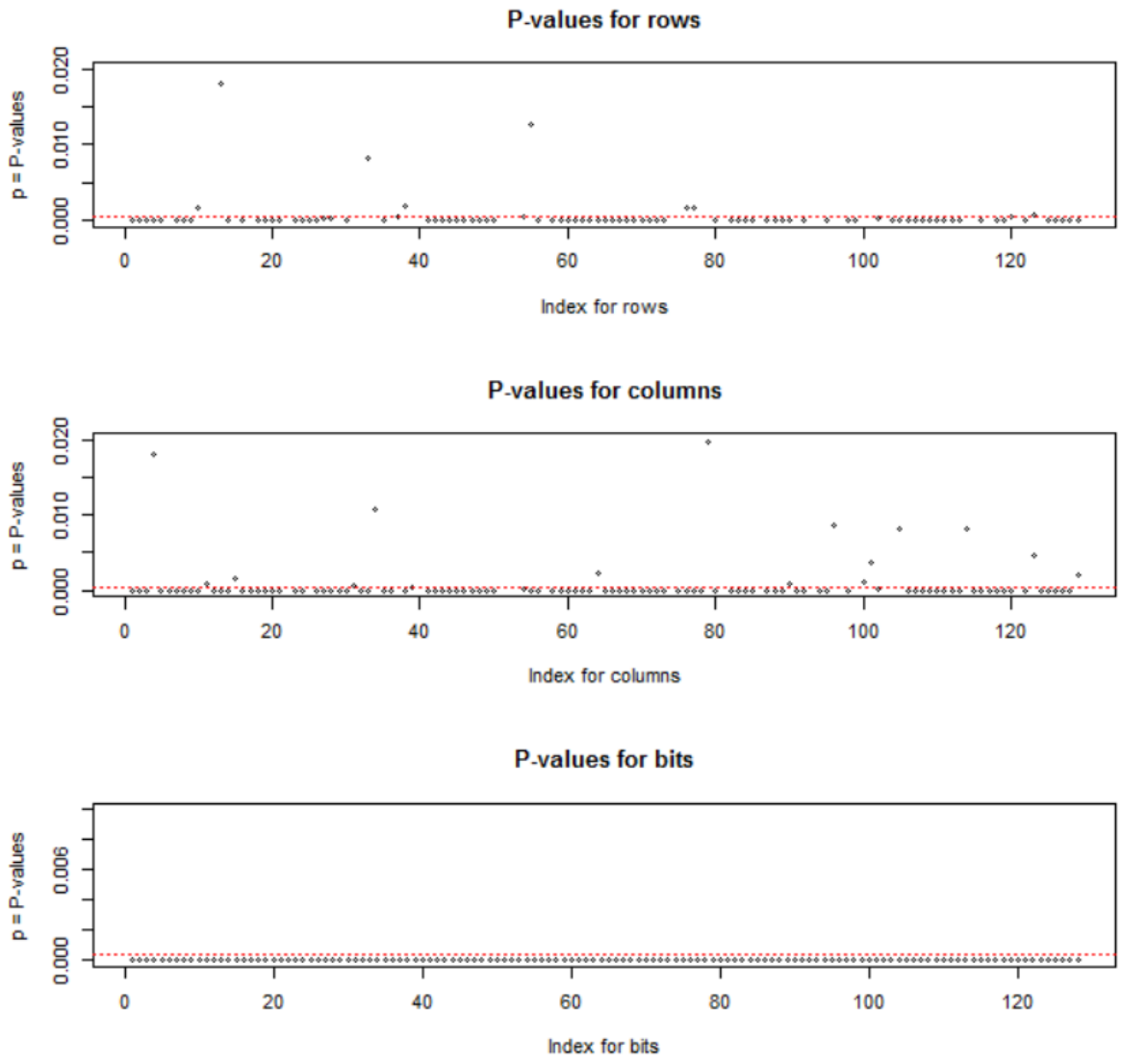

Figure 6: The frequncies in Round 1

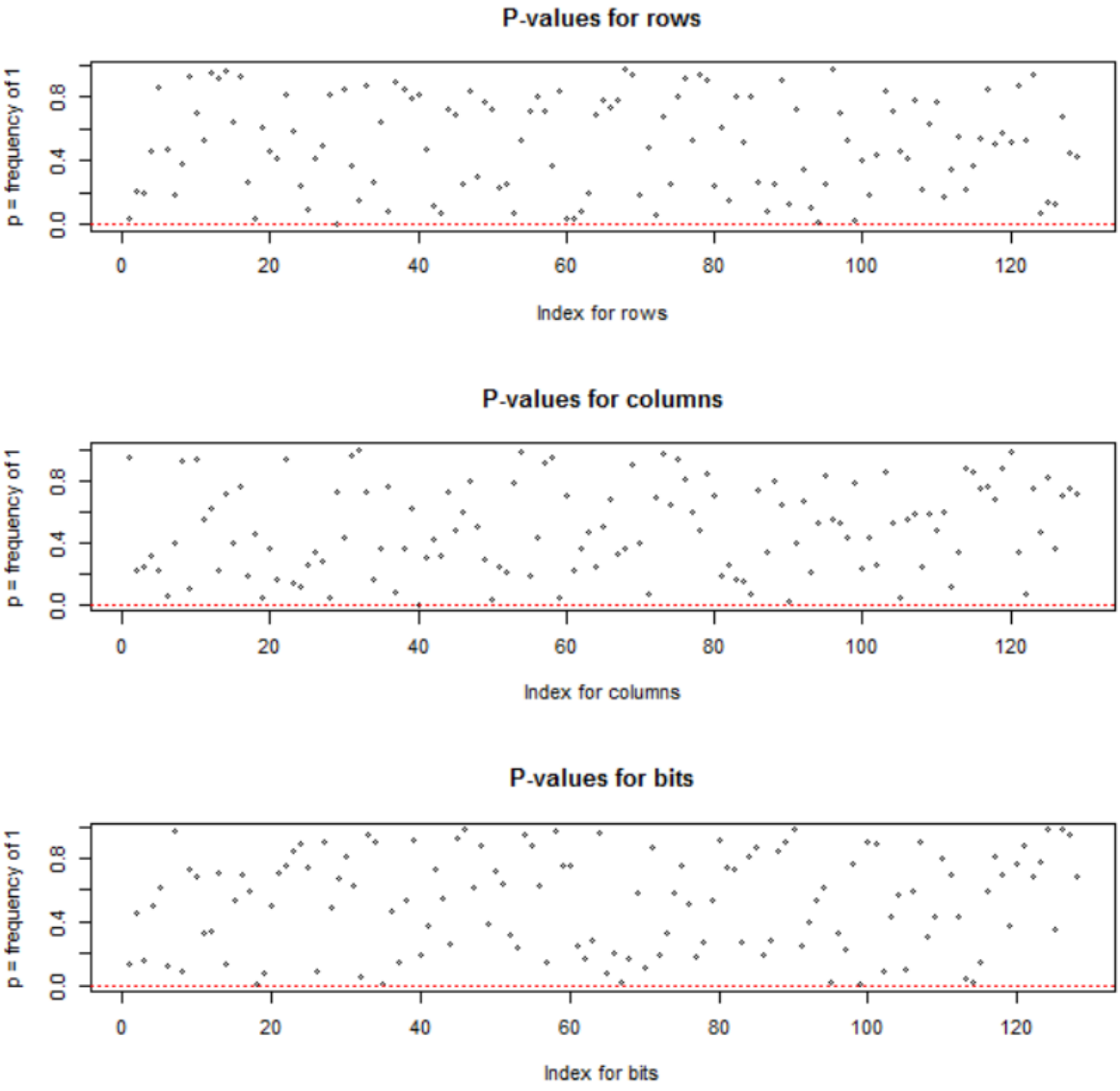

Figure 7: The frequncies in Round 9

Chapter 6

6.1. CryptoStat Test Suite

Kaminsky and Sorrell have introduced CryptoStat (Kaminsky & Sorrell, 2013), which is a statistical test suite that evaluates cryptographic functions' randomness. It implements the Bayesian model selection to analyze Block ciphers' outputs since some earlier statistical tests have failed in detecting randomness in cryptographic functions as previously discussed in 4.2. However, the array based approach that was explained in chapter 5 will be used when applying the tests on AES instead of Bayesian models.

CryptoStat test suite consists of 7 tests: the linear approximation test, the coincidence test, the input- output independence test, the complement test, the ciphertext independence test, the strong avalanche test, and the uniformity test. These tests were applied on 7 block ciphers: Kasumi (3gpp.org), Blowfish (Schneier, 1993), SQUASH (Shamir, 2008), AES (NIST, 2001), IDEA (Lai & Massey, 1999), PRESENT (Bogdanov, Knudsen, Leander, Paar, Poschmann, Robshaw et al., 2007), and ThreeWay (Daemen, Govaerts, & Vandewalle, 1993). The results were presented and discussed in the paper (Kaminsky & Sorrell, 2013).

6.2. Applying CryptoStat Test Suite on AES

Some of the tests mentioned above will be applied on AES block cipher to evaluate its randomness using the array based approach. The statistical hypothesis testing is used, where H_0 is the null hypothesis that assumes $P = 0.5$ and passing the test. On the other hand, we have H_1 as the alternative hypothesis, which assumes $P \neq 0.5$ and failing the test.

6.2.1. The Linear Approximation Test

- Test Purpose

The purpose of the linear approximation test is to assess the cryptographic function's behavior by evaluating precision of an equation that approximates the cryptographic function's activity.

- Function Call

$$(\sum_{b \epsilon H \cup G} b) mod 2 \qquad (4)$$

where:

b is a bit group of size g, where g takes on the values 1, 2, 4, 8, ..., *N*, and *N* is the maximum size of the ciphertext *C*.

H is a selected bit group from the ciphertext *C* with the same size as b.

G is a selected bit group from the string *K*//*P* with the same size as b, where *K* is the encryption key and *P* is the original plaintext.

- Test Description

The previous sum either evaluates to 0 or 1. If it evaluates to 1, the outcome will be considered as a Bernoulli success. Otherwise, it is considered a Bernoulli failure. For each bit group, the successes are counted. After that, equation 2 is applied for each column and row of AES. For each round, if the bit group examined cannot be presented as a linear estimation of the cryptographic function's activity, the Bernoulli success probability θ will be equal to p, and some other value otherwise. Figure 8 below from Prof. Kaminsky's paper shows how the linear approximation test works.

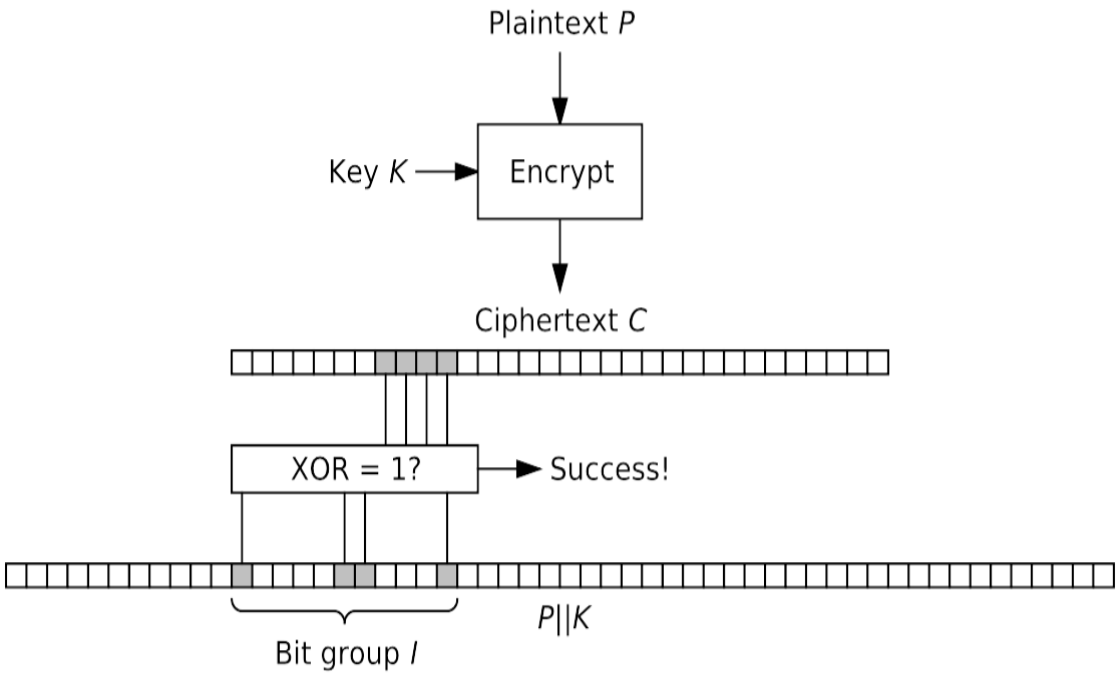

Figure 8: Linear Approximation Test

- Applying the Linear Approximation Test on AES

When applying a test, the outputs should be 10 numbers for each dimension used since there are 10 rounds in AES. These 10 numbers represent the number of cases that having a P-value different than P_0 , or $P \neq 0.5$, which means failing the test. Table 5 shows the results of applying the linear approximation test on the first two rounds of the AES block cipher.

Dimension	Round1	Round2
Rows	50	0
Columns	64	0
Bits	108	65

Table 5: Linear approximation test results

Using rows and columns dimensions produced 50, 64 failure cases, respectively in the first round. On the other hand, using bits' dimension led to failure in the second round too. While the rounds 3 to 10 pass the test, and have 0 failure cases, which means the other rounds are random. Figure 9 and 10 shows the frequencies of 1's after applying the linear approximation test for the first two rounds. Points inside the dashed line are considered random, and we fail to reject the null hypothesis. Points outside the dashed lines have non-randomness issues. Figure 11 and 12 in the following pages show the P-values of each ciphertext when applying the linear approximation test on the first two rounds of AES. The first round is highly non-random because there are plenty of points under the significance level, which means rejecting the null hypothesis. Figure 12 also shows the P-values of the applying the linear approximation test on round 2. Only using bits reveals 6 points under the level of significance, which was shown in table 5 earlier.

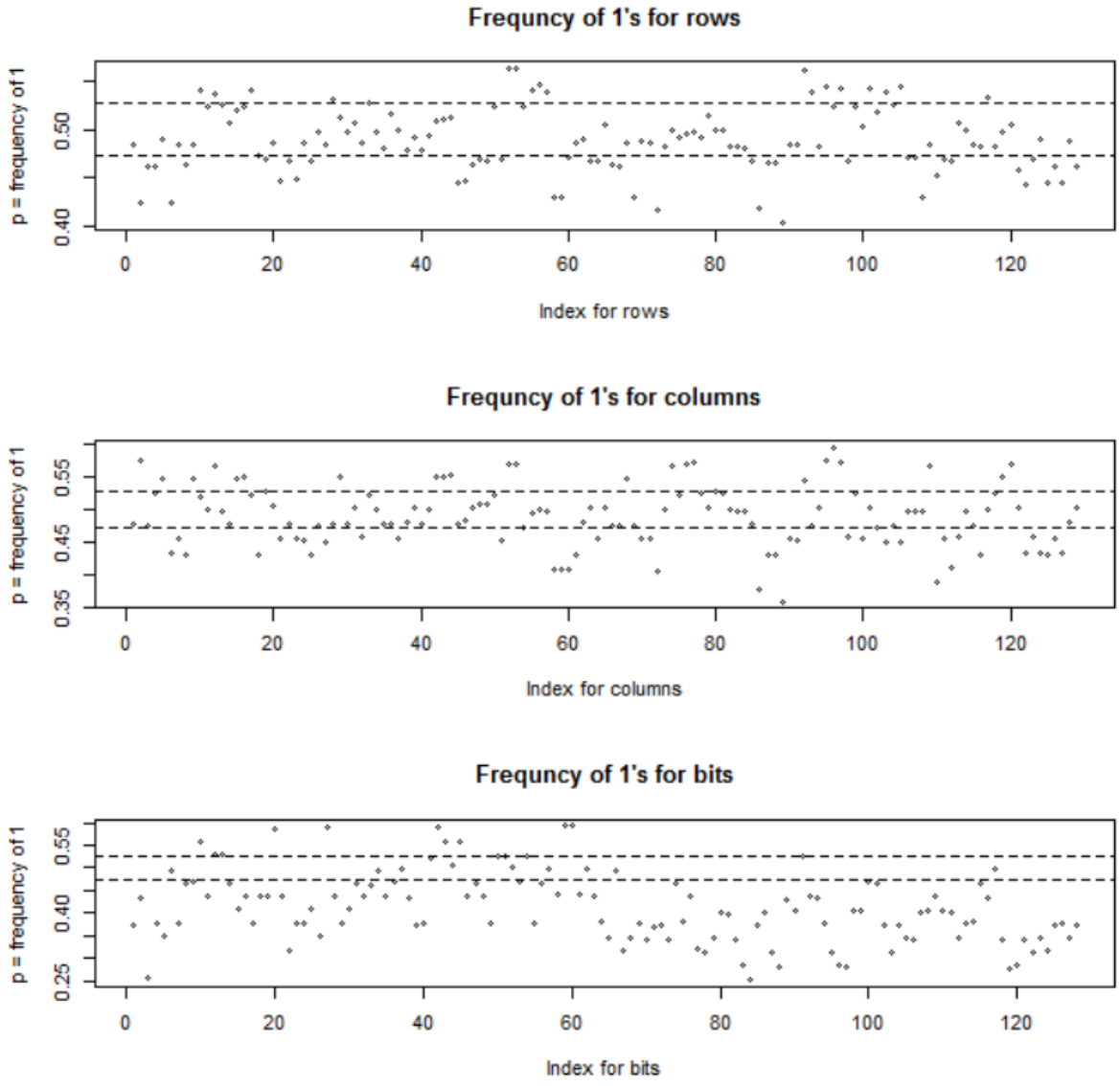

Figure 9: Frequencies of 1's for the linear approximation test in round 1

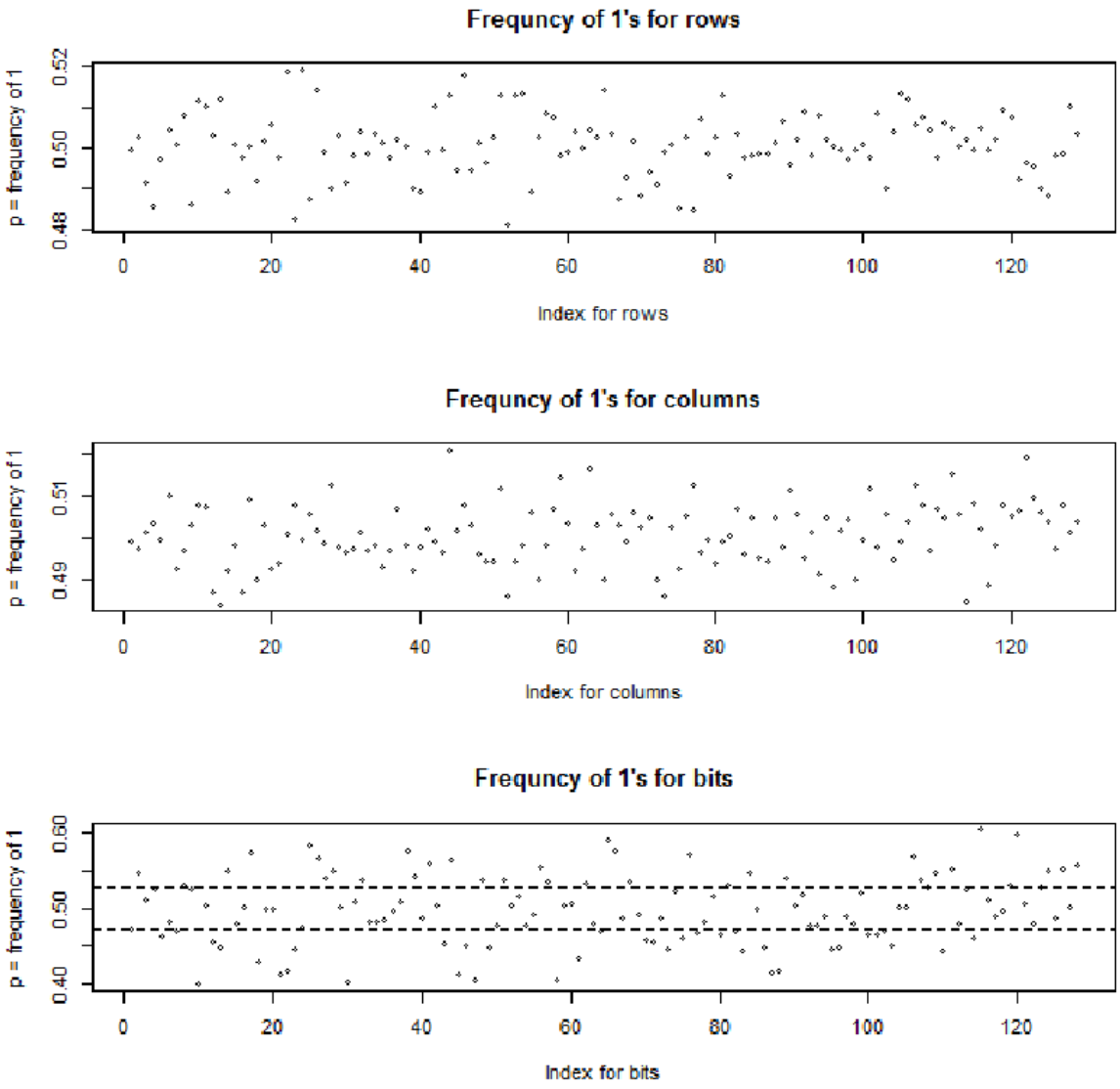

Figure 10: Frequencies of 1's for the linear approximation test in round 2

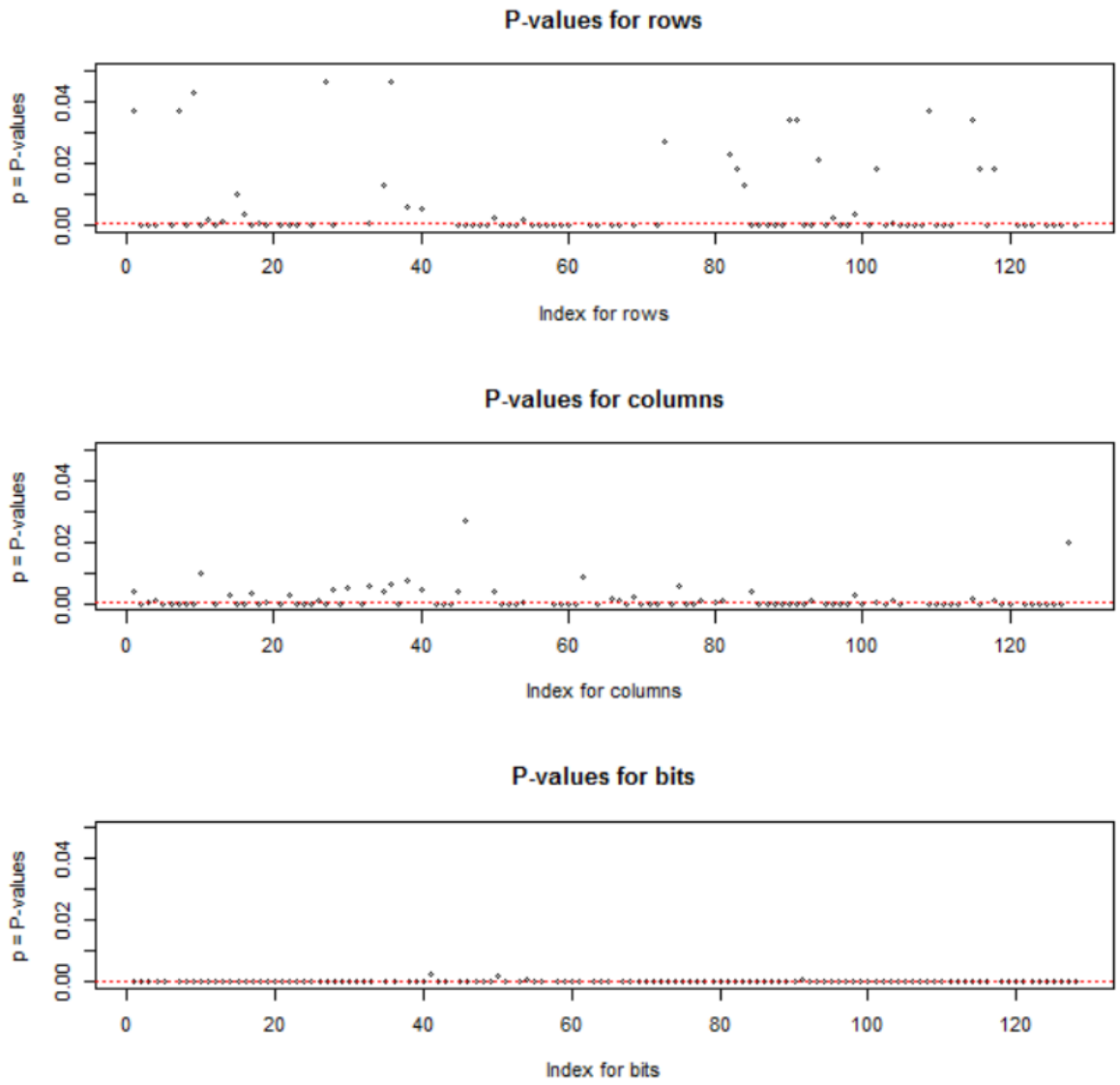

Figure 11: Linear approximation test P-values for round1

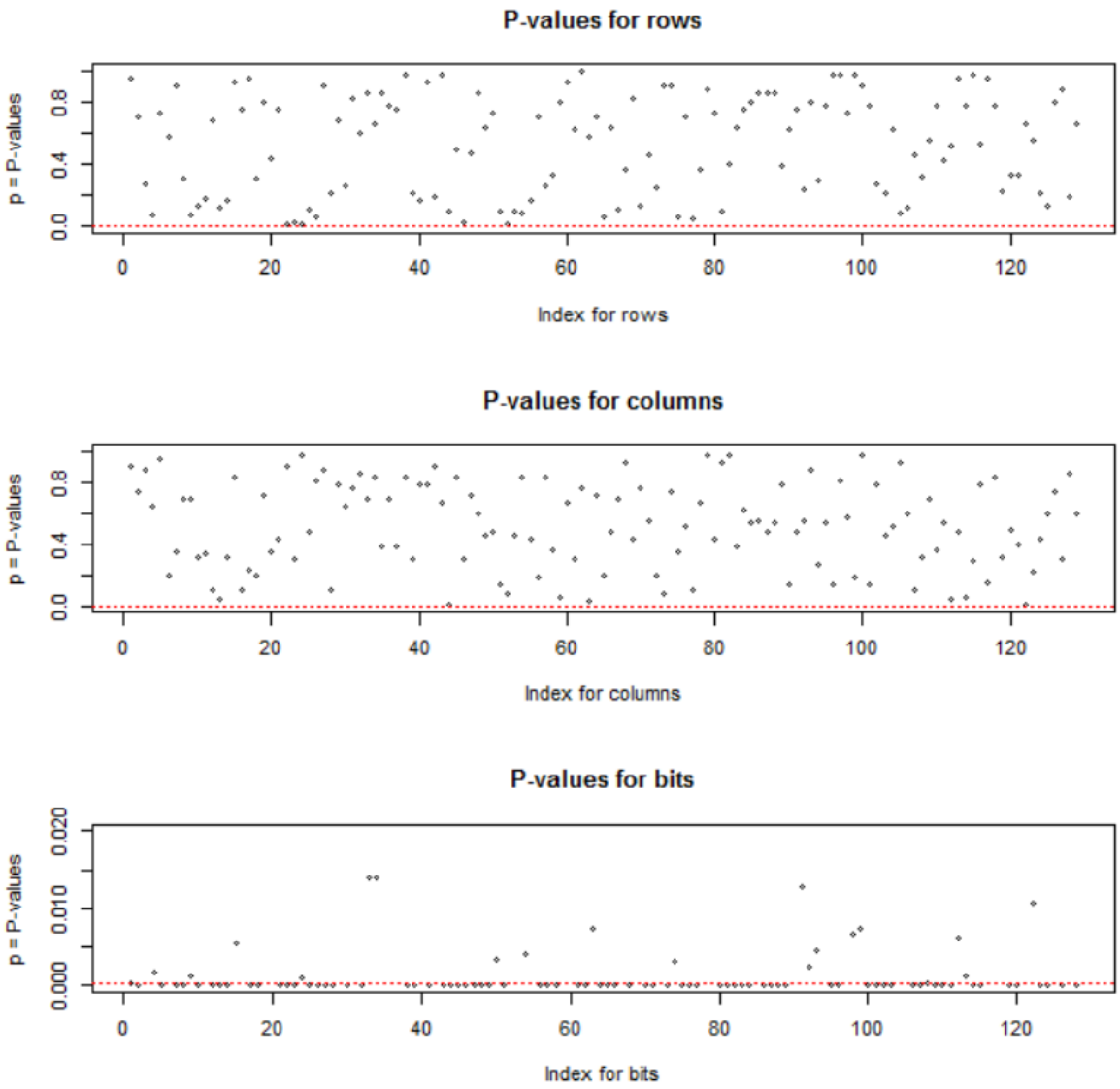

Figure 12: Linear approximation test P-values for round2

6.2.2. The Coincidence Test

- Test Purpose

The coincidence test as described by (Kaminsky, 2013) uses a Bayesian technique to evaluate the mapping of a (plaintext, key) to ciphertext in block ciphers.

- Test Description

When *P* the plaintext is encrypted with the encryption key *K*, The Ciphertext output *C* is then obtained. *V* is an arbitrarily chosen output which is compared to *C*. The coincidence test examines all the bit sequences of size g, where g takes a value of 1, 2, 4, 8, ..., *N*, and *N* is the maximum size of the ciphertext *C*. *R* and *O* are bit sequences selected from *V* and *C* where $|R|=|O|=g$. A coincidence accrues when each bit of *R* is equal to the corresponding bit of *O*, which is a success Bernoulli trial.

The coincidence test performs *n* Bernoulli trials and counts the number of coincidences *k*. If the block cipher's (plaintext, key) to ciphertext mapping is random, a coincidence should occur with a Bernoulli success probability $p = 2^{-g}$. The coincidence test decides between two binomial models: H_0 where the success probability is 2^{-g}, and H_1 where the success probability is otherwise. Then, the coincidence test is repeated for each round in AES. Figure 13 below describes how the coincidence test works.

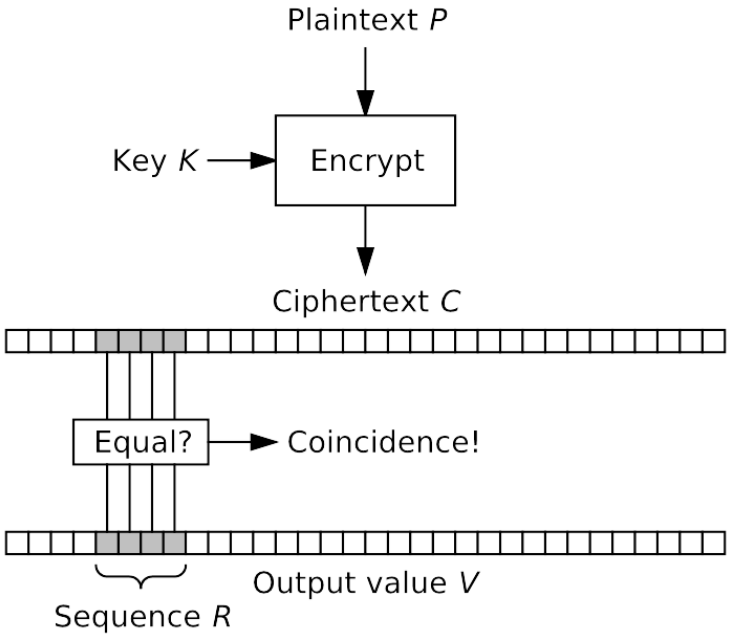

Figure 13: Coincidence Test

- Applying the Coincidence Test on AES

Table 6 below represents the results of applying the coincidence test on AES data. When using rows or columns to apply the coincidence test, only the first round appears to fail the test. While using bits to calculate *P-values* shows non-randomness in the first two rounds. There were no non-randomness issues detected in rounds 3 to 10.

Dimension	Round1	Round2
Rows	5	0
Columns	16	0
Bits	83	8

Table 6: Coincidence test results

Figure 14 and 15 below show the frequencies of 1's when applying the coincidence test for round 1 and round 2.

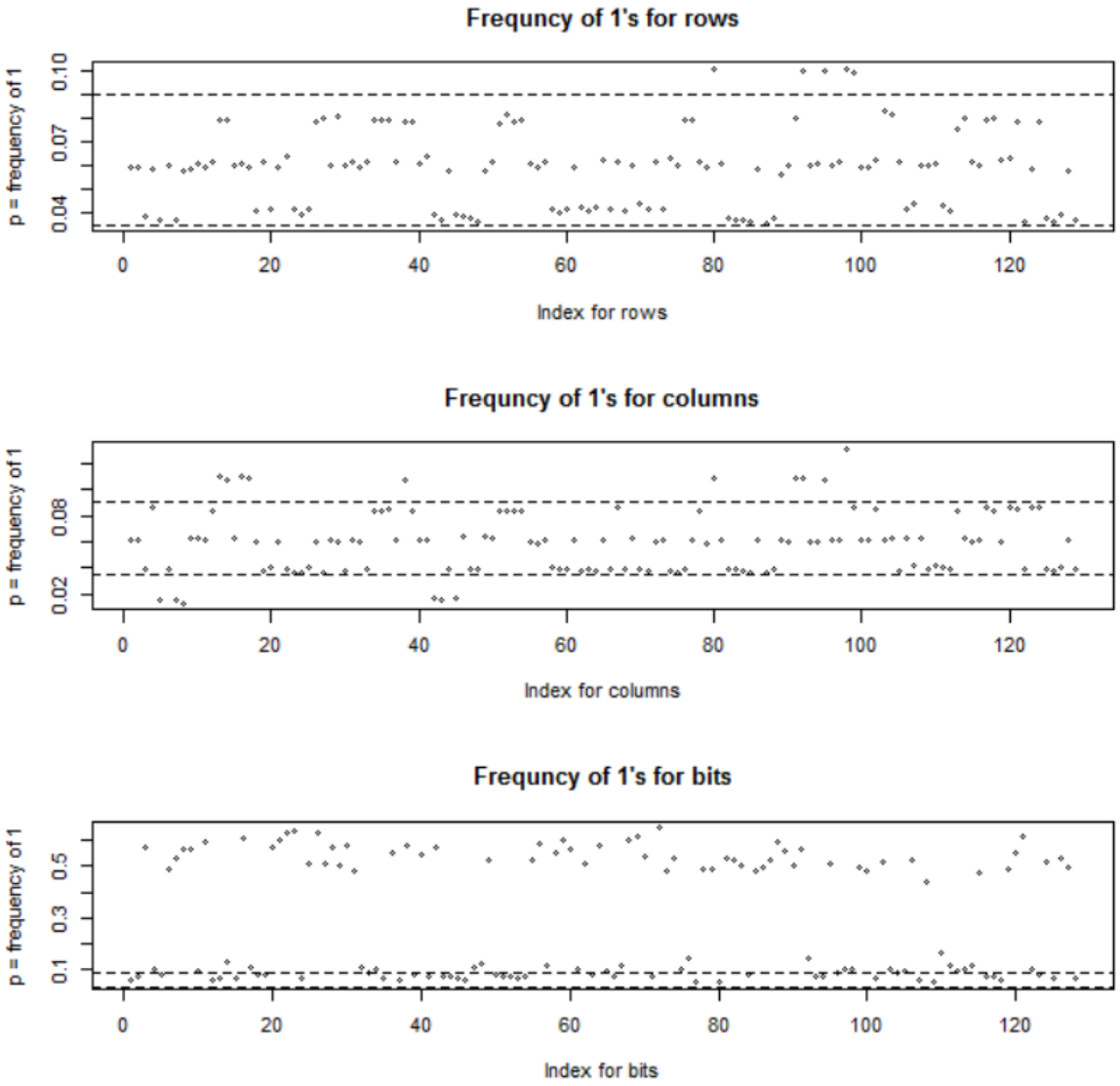

Figure 14: Frequencies of 1's for the coincidence test in round 1

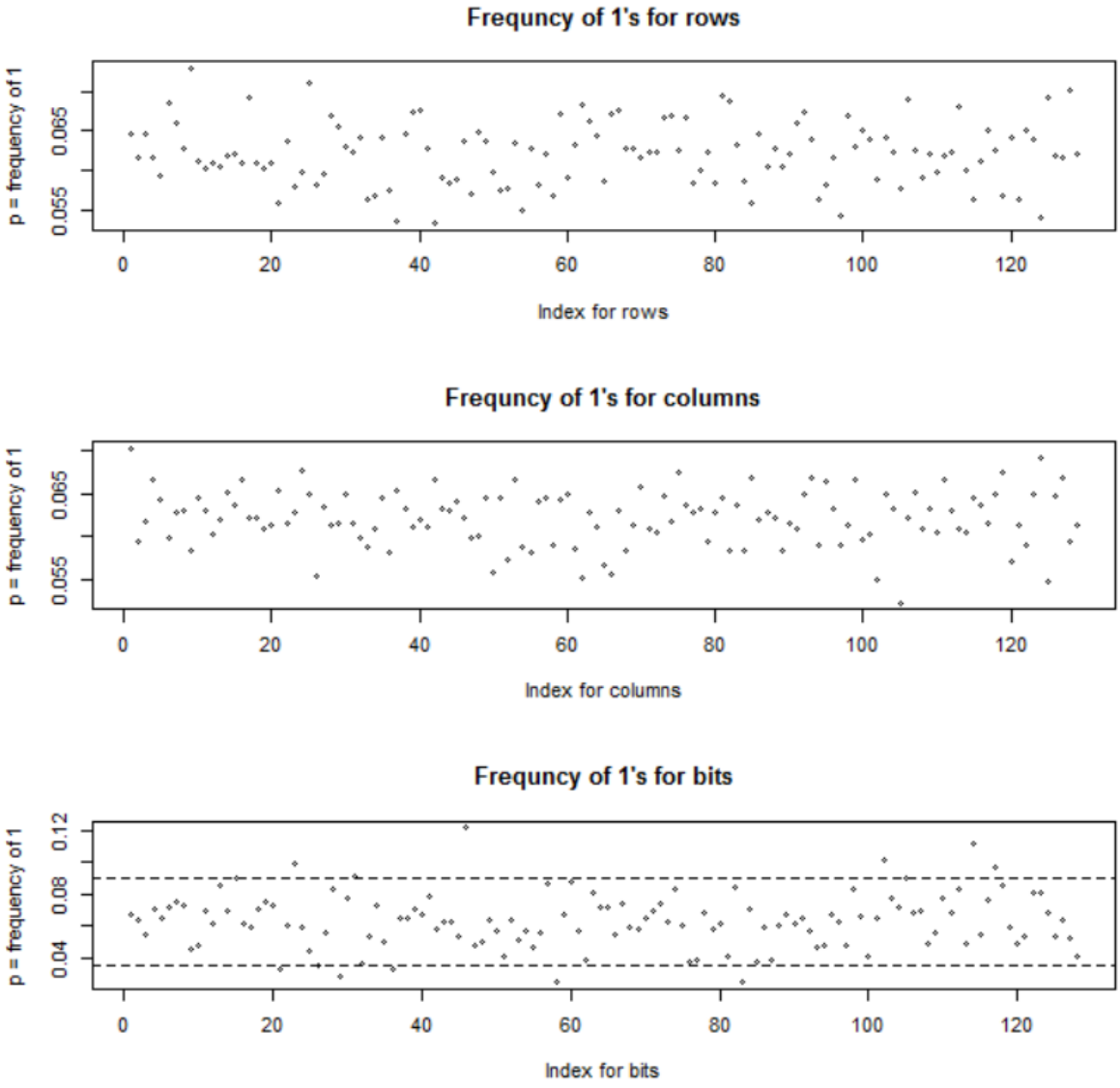

Figure 15: Frequencies of 1's for the coincidence test in round 2

Figure 16 displays the P-values for the first round. The three plots in the figure show the 5, 16, and 83 points mentioned above in table 6. These points are under the line, which means they are not random and fail the coincidence test.

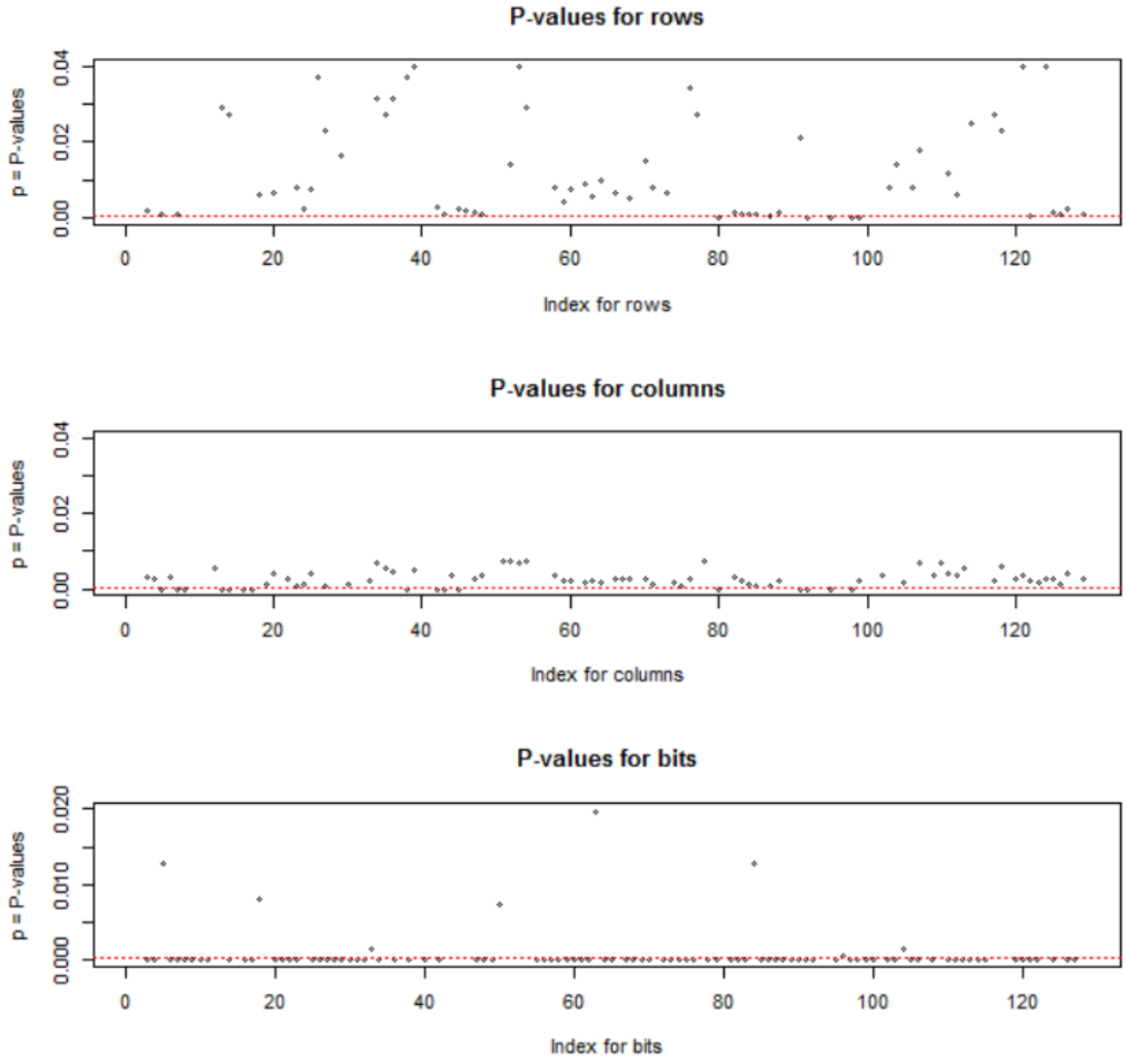

Figure 16: Coincidence test P-values for round1

On the other hand, we have figure 17 that shows the P-values for round 2. Only using bits when calculating P-values reveals 8 non-randomness issues, which was also shown in table 6. Lack of dashed lines for the first two plots means that the significance threshold is outside of the range of the values shown.

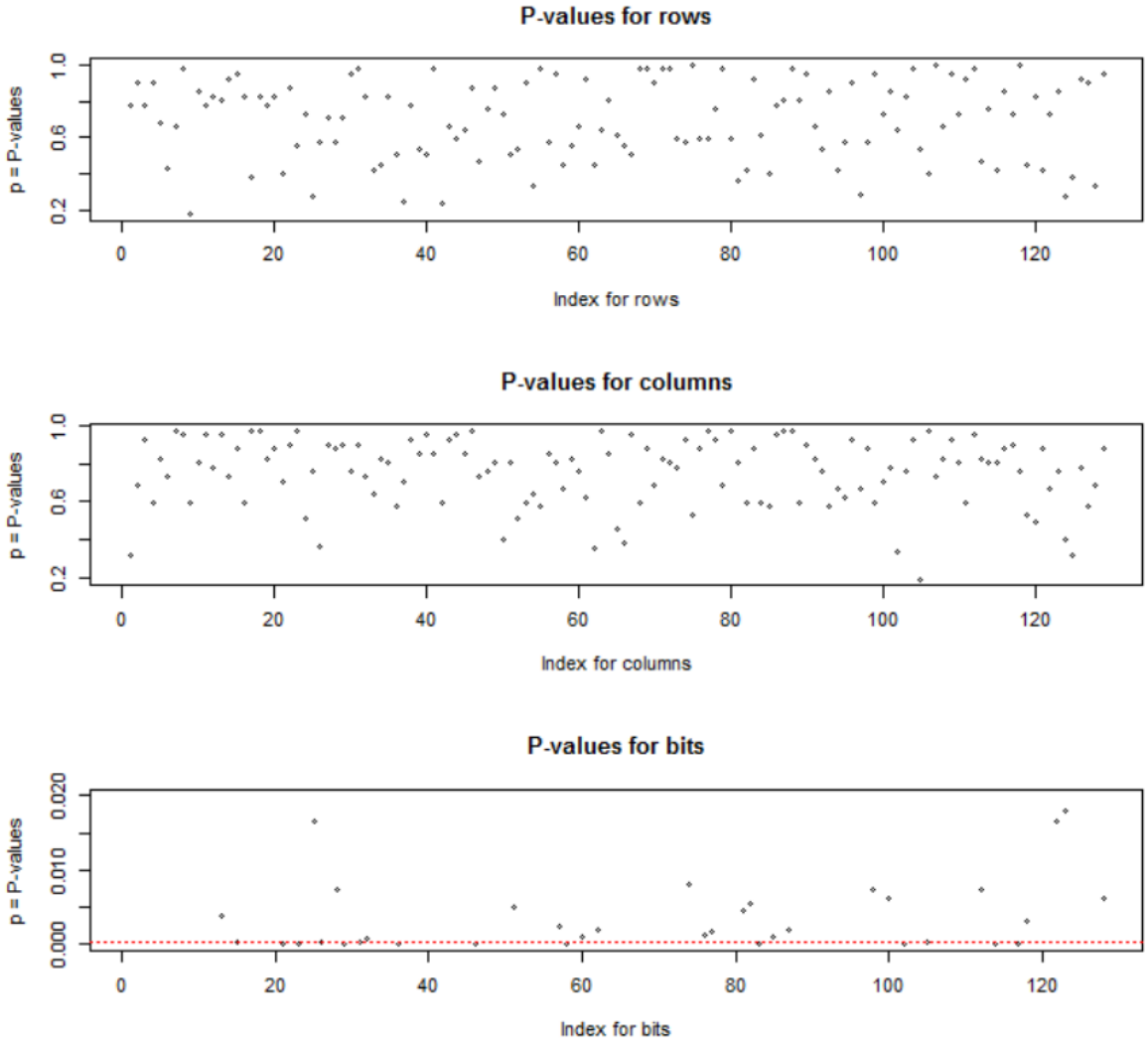

Figure 17: Coincidence test P-values for round2

6.2.3. The Input- Output Independence Test

- Test Purpose

The input-output independence test consists of number of subtests. Each subtest examines whether a particular bit group of a cryptographic function's output is independent of particular input bit group with the same size.

- Test Description

Let *C* be the ciphertext output from the plaintext *P* and the encryption key *K* from randomly chosen plaintexts, keys and the corresponding ciphertext produced by the block cipher AES. The input-output independence test's subsets examine each pair of bit groups chosen randomly from the input and the output. The size of these bit groups is g, where g takes on the values 1, 2, 4, 8, ..., *N*, and *N* is the maximum size of the ciphertext. The input bit group *I* is constructed from bits selected from randomly generated positions in *K* \\ *P* where $|I| = g$. While *O*, the output bit group is constructed from bits selected from randomly generated positions in *C*, where $|O| = g$. The test then *XOR*s these like-sized bit groups, $V = I \oplus O$. *V* is then checked for uniform distribution using two methods: counting the 1's in each bit position and Chi-square test. If the input bit groups are independent of the output bit groups, then *V* is uniformly distributed and classified as random, otherwise it is nonrandom. The input-output independence test choses between two binomial models: H_0 where *V* is uniformly distributed and H_1, where *V* has some other distribution. Input-output independence test is shown in figure 18 below.

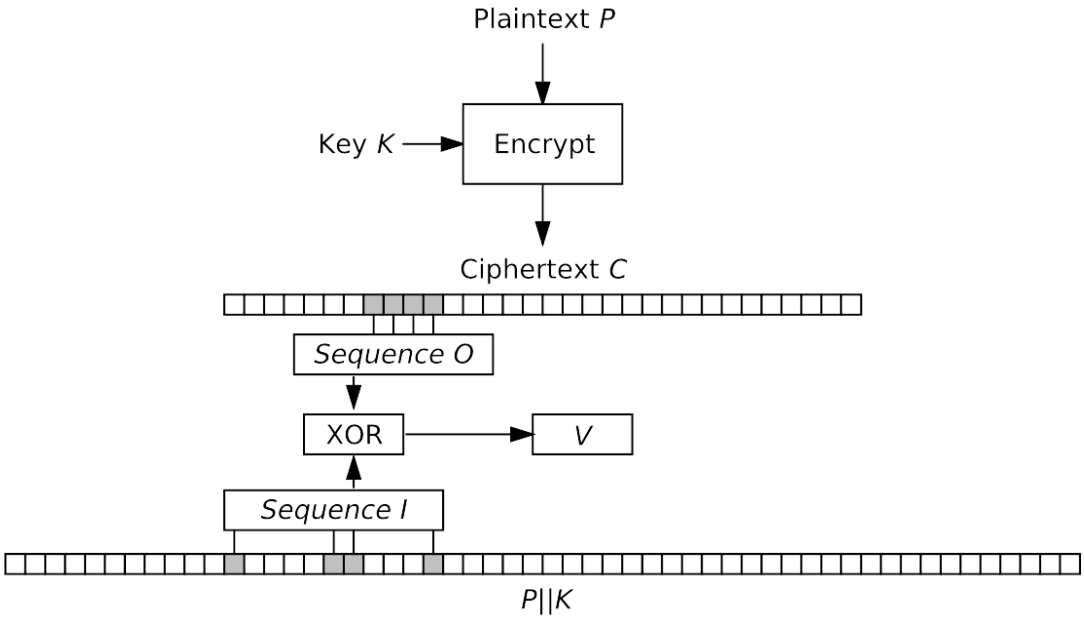

Figure 18: Input-output independence test

- Applying the Input-Output Independence Test on AES

Table 7 shows the results of applying the input-output independence test on AES. Applying the test revealed failing the test issues in the first two rounds while other rounds passed the test and showed acceptable random behavior.

Dimension	Round1	Round2
Rows	36	0
Columns	41	0
Bits	128	0

Table 7: Input-output independence results

Figure 19 shows the frequencies of 1's after applying the input-output independence test on the first round.

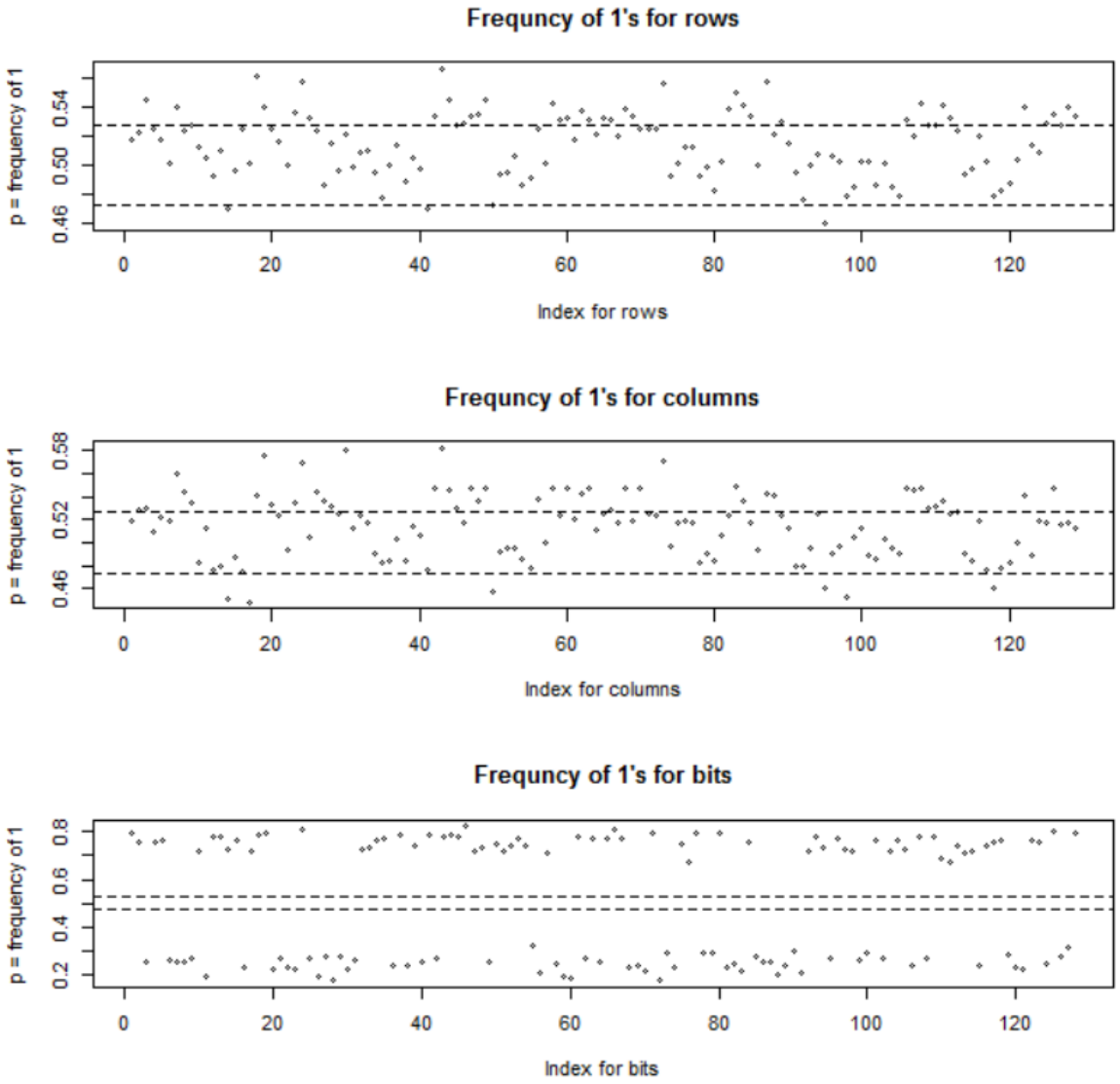

Figure 19: Frequencies of 1's for the input-output independence test in round 1

Figure 20 below shows the P-values when applying the input-output independence test on the first round.

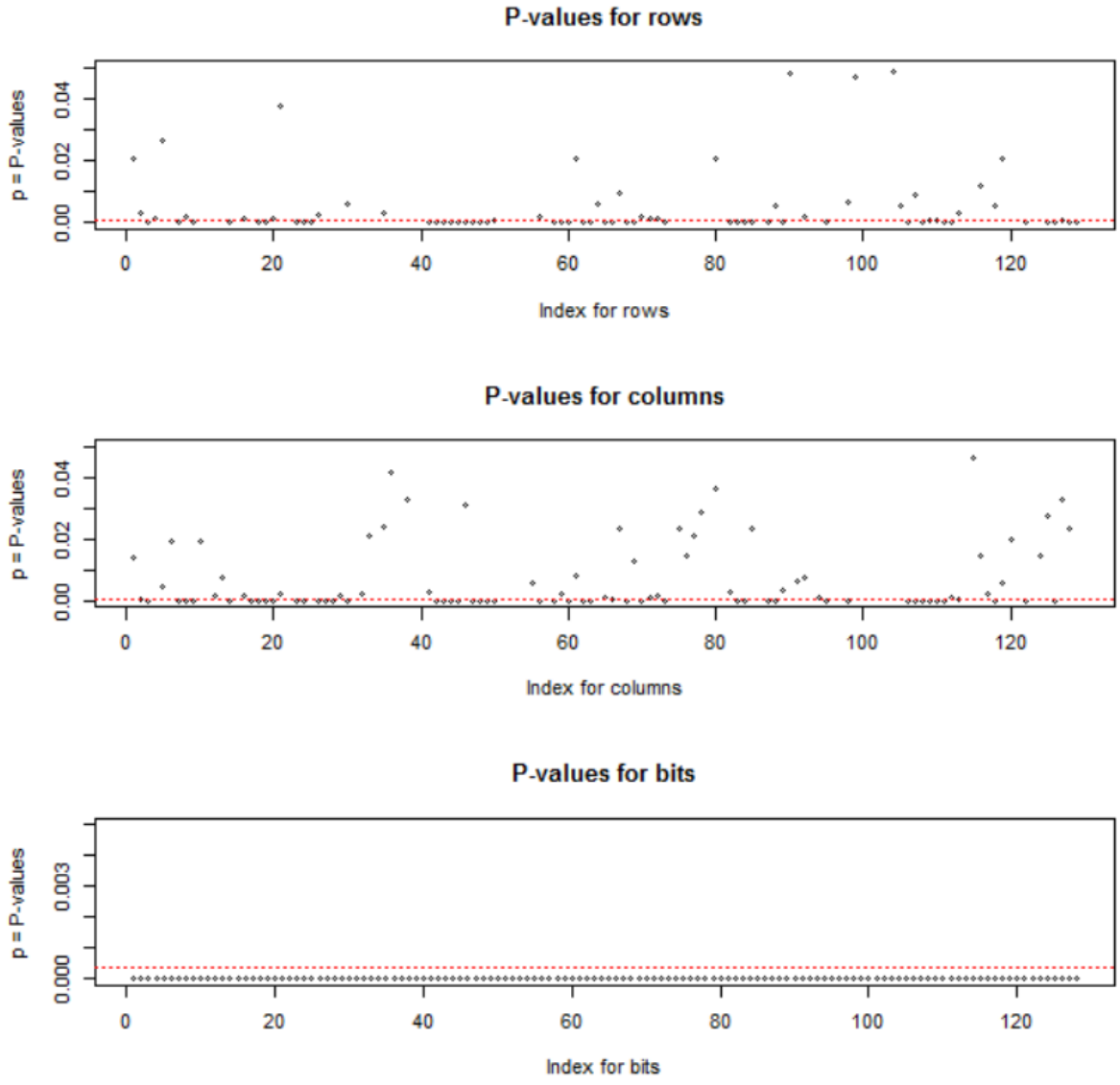

Figure 20: Input-output independence test P-values for round1

When using Chi-square test, the first two rounds failed the input-output independence test. Table 8 displays the number of rows, columns, and bits failing the input-output independence test.

Dimension	Round1	Round2
Rows	16	0
Columns	51	0
Bits	89	9

Table 8: Input-output independence results using Chi-squared test

Figure 21 and 22 below show the P-values when applying the input-output independence test on the first and the second rounds using Chi-square test.

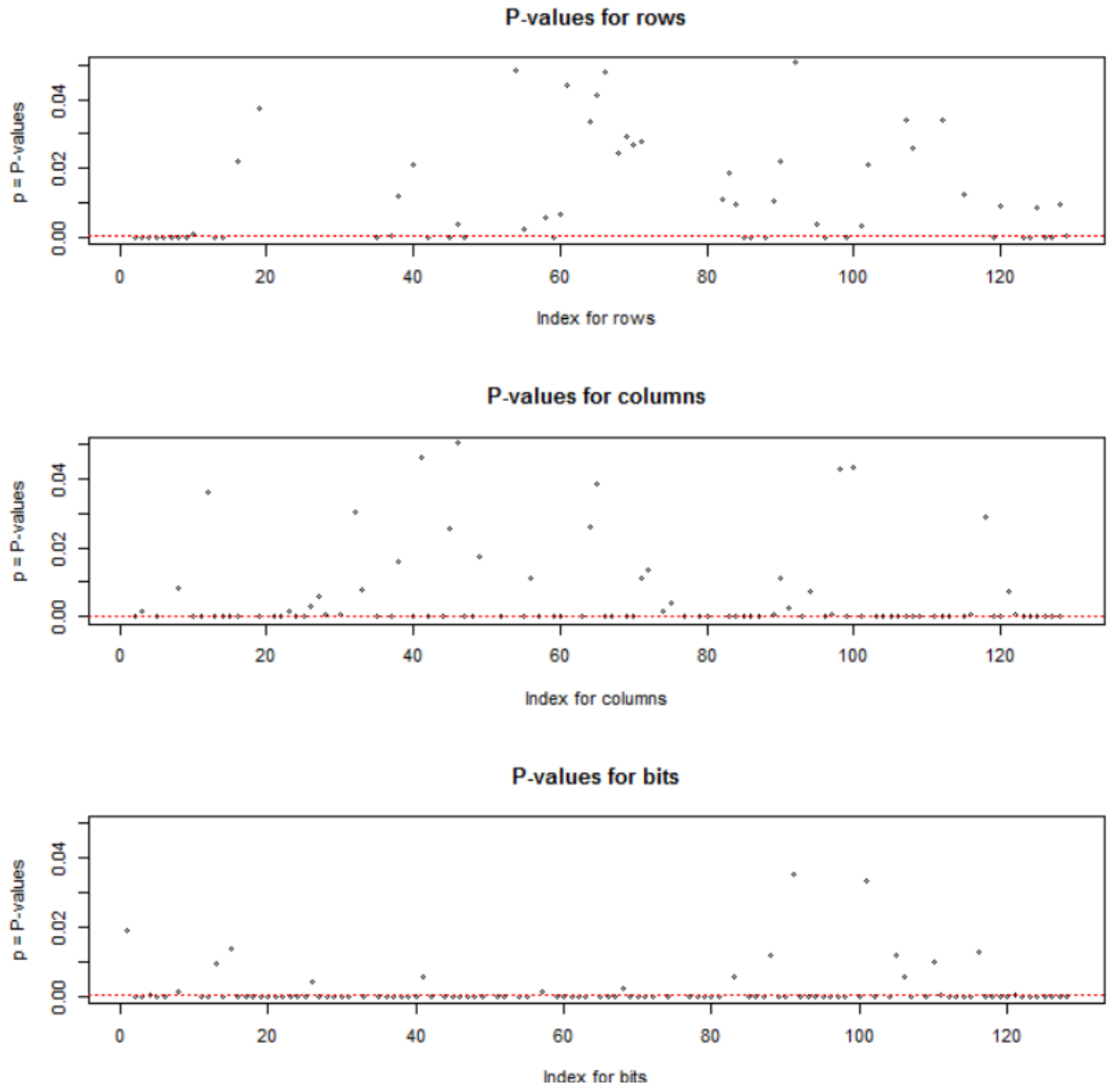

Figure 21: Input-output independence test P-values for round1 using Chi-squared test

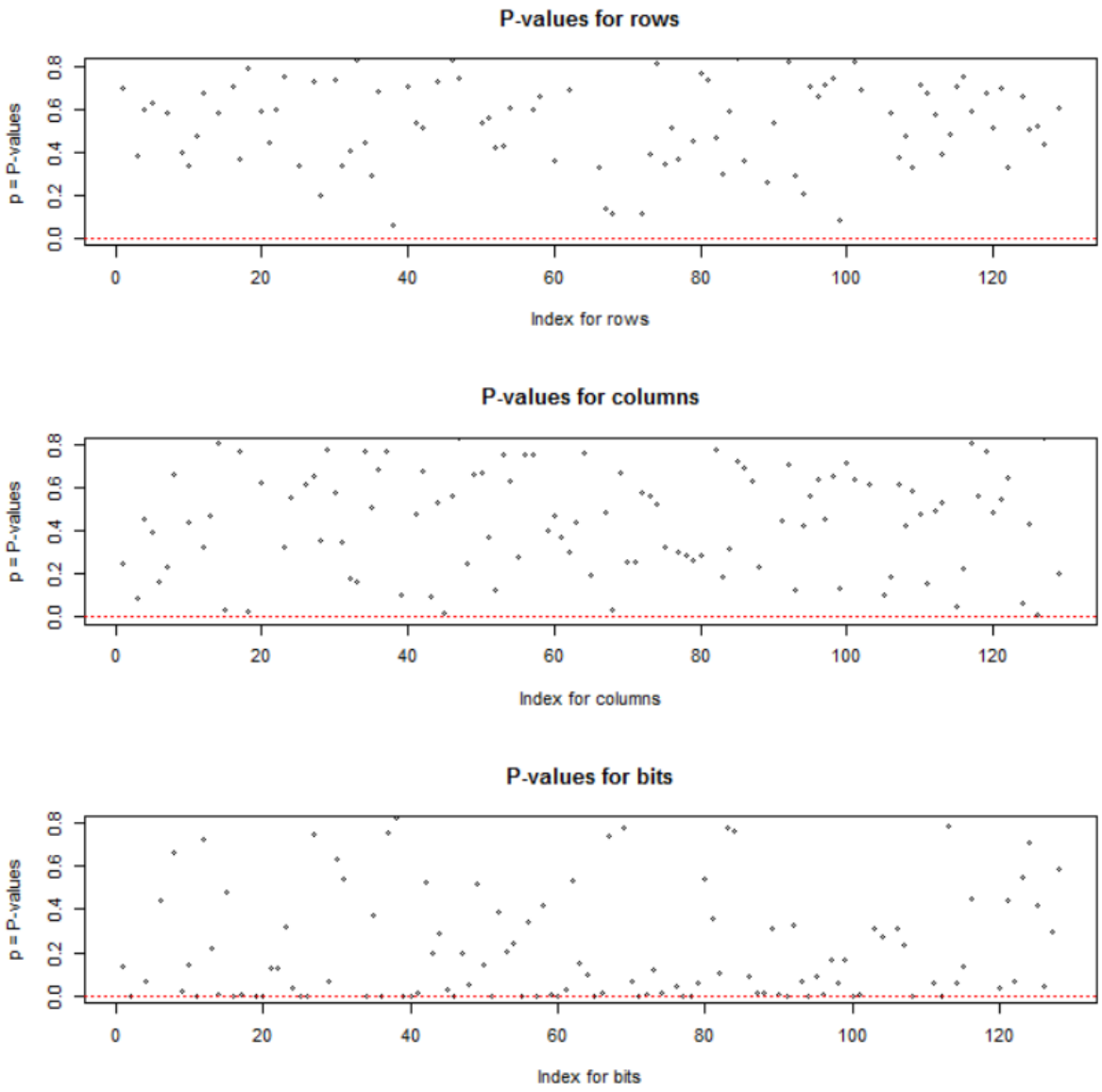

Figure 22: Input-output independence test P-values for round2 using Chi-squared test

6.2.4. The Complement Test

- Test Purpose

If C' is a cryptographic function's output for the inputs: P' and K', which are the bitwise complements of the original inputs: a plaintext (P) and an encryption key (K). Each subset of the complement test's subsets examines whether particular bit sequences of a cryptographic function's output C are independent of the same bit sequences of C'.

- Test Description

Let C be the ciphertext output when P, a valid plaintext is encrypted with the encryption key K.

On the other hand, let C' be the ciphertext output when ¬P, the bitwise complement of the original plaintext is encrypted with key ¬K, the bitwise complement of the original encryption key. The complement test investigates all bit groups of size g=1, and B bit groups of size g where g takes on the values 2, 4, 8, ..., N and N is the maximum size of the ciphertext.

Let O and O' are like-sized bit groups produced from randomly chosen positions from C and C' respectively. If $V = O \oplus O'$ and V is uniformly distributed, then the cipher-texts C and C' are independent of each other and V can be said to be random. The complement test decides between two binomial models: H_0 where V is uniformly distributed and H_1, where V has some other distribution.

- Applying the Complement Test on AES

Due to the complement test's description, it cannot be applied on AES because we don't have the output C' since we have no access to encrypt the bitwise complement plaintexts and keys.

6.2.5. The Ciphertext Independence Test

- Test Purpose

The ciphertext independence test's subsets examine if each bit group of a cryptographic function's output is independent of other bit groups in that same output.

- Test Description

Let *C* be the ciphertext output when a plaintext *P* is encrypted with an encryption key *K*. The ciphertext independence test evaluates each pair of bit groups from the ciphertext *C*. Let O_1 and O_2 be two bit groups from two randomly chosen bit positions in the ciphertext *C*. These two bit groups are like-sized with $|O_1|=|O_2|=g$ where *g* takes on the values 2, 4, 8, ..., *N/2* and *N* is the maximum size of the ciphertext. In addition, $O_1 \cap O_2 = \emptyset$, the intersection of these two sets must be empty. If the cryptographic function being tested satisfies the ciphertext independence criterion, the two bit positions in the ciphertext *C* (where O_1 and O_2 were constructed from) are independent and $V=O_1 \oplus O_2$ is a uniformly distributed variable. Then, one of the two models will be chosen: H_0 where *V* is uniformly distributed and H_1, where *V* has some other distribution. Figure 23 below from Kaminsky and Sorrell paper shows how the cipher independence test works.

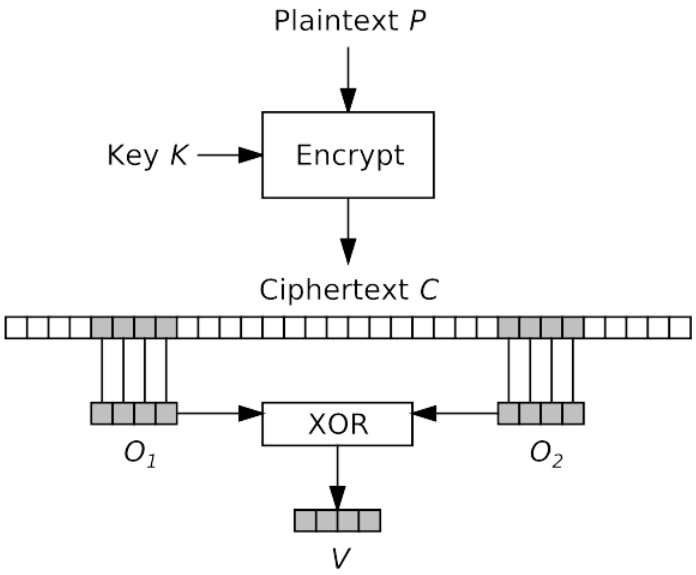

Figure 23: Ciphertext independence test

- Applying the Ciphertext Independence Test on AES

Table 9 represents the number of rejecting the null hypothesis cases when applying the ciphertext independence test on AES using counting the 1's in each bit position method.

Dimension	Round1	Round2
Rows	61	0
Columns	56	0
Bits	128	25

Table 9: Ciphertext independence test results

Figure 24 and 25 show the frequencies of 1's after applying the ciphertext independence test on round 1 and 2. In figure 24 for round 1, all the points are under the dashed lines when using bits. The second round in figure 25 only have some failing the test cases.

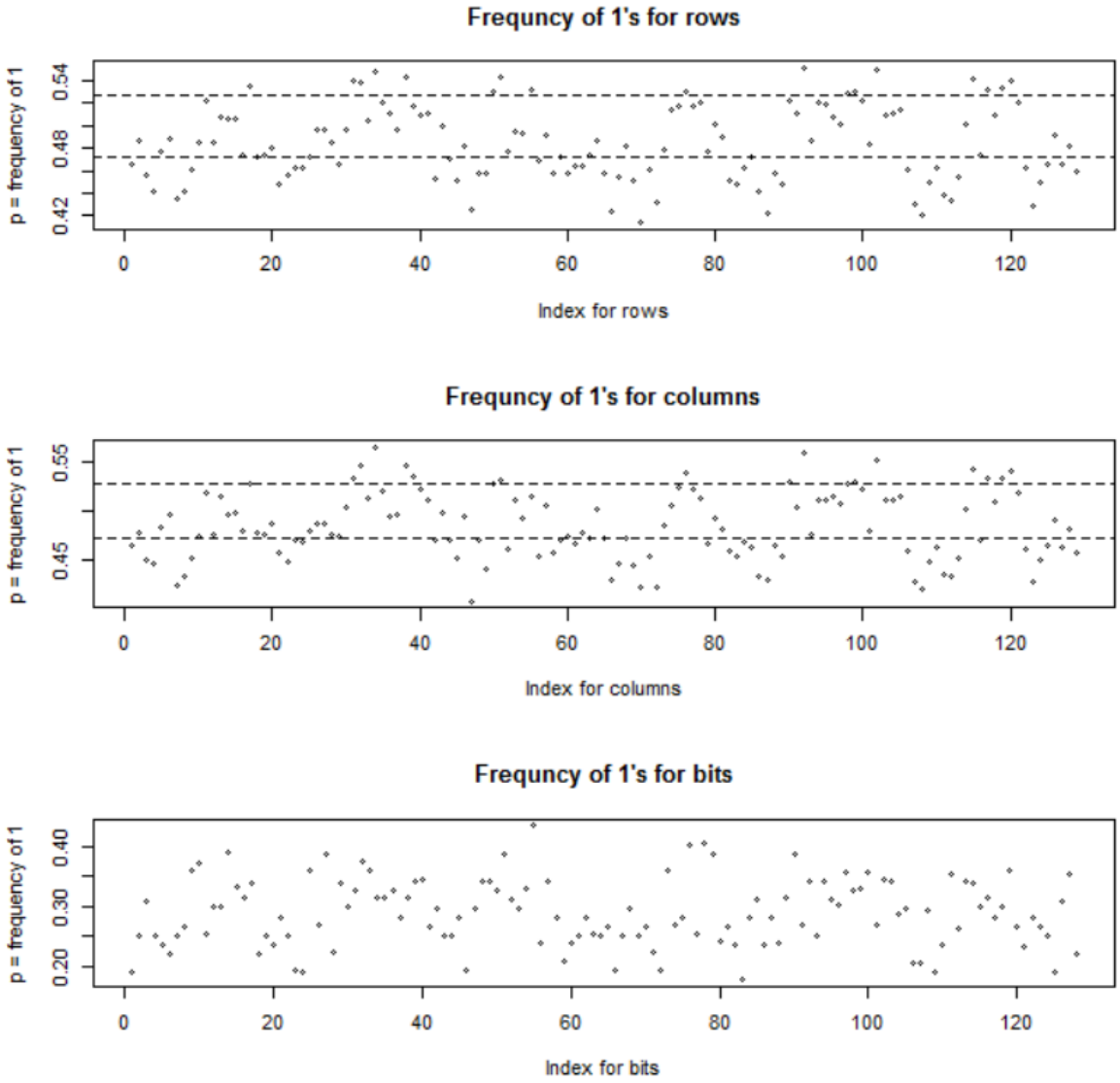

Figure 24: Frequencies of 1's for the ciphertext independence test in round 1

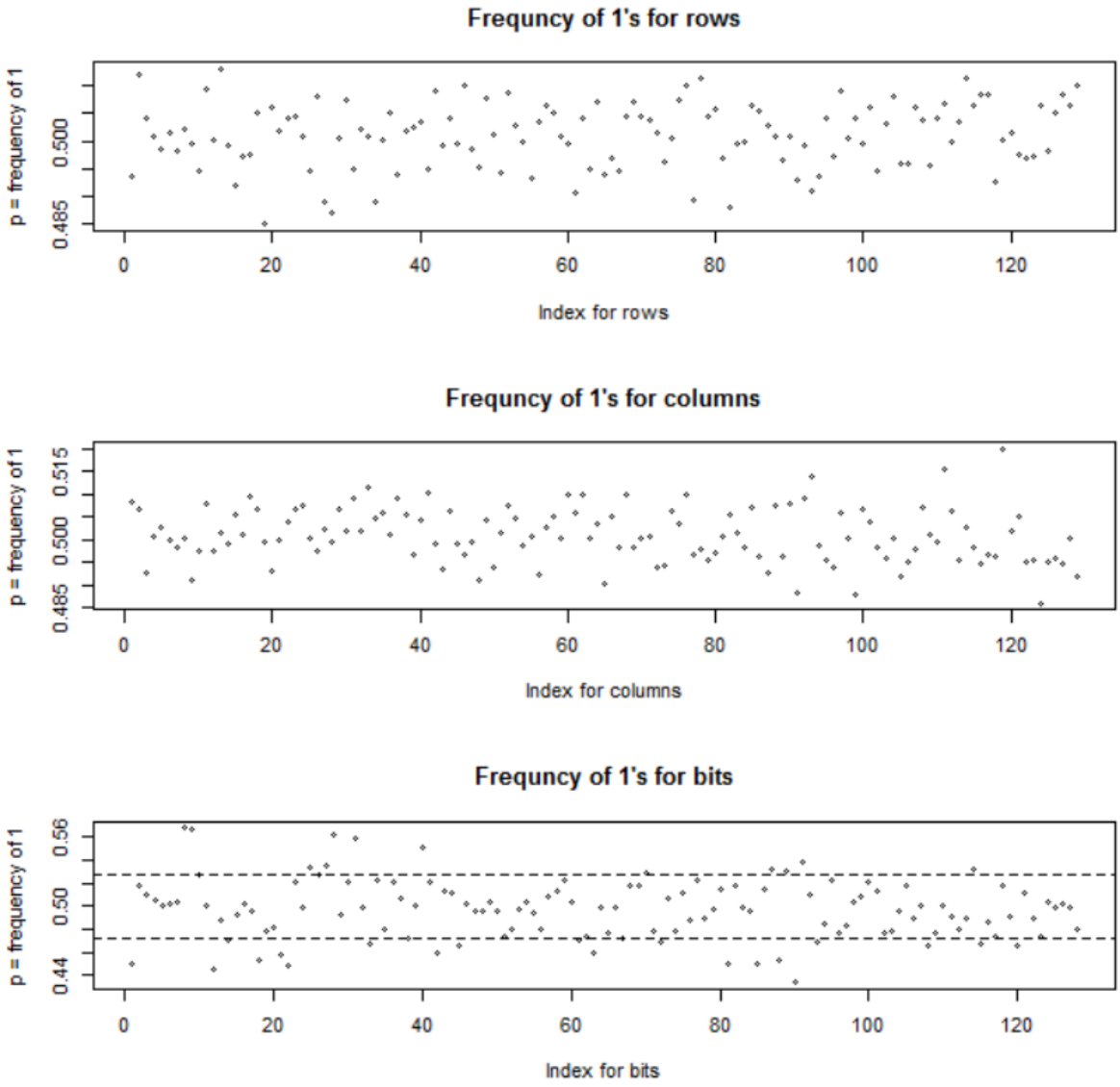

Figure 25: Frequencies of 1's for the ciphertext independence test in round 2

Figure 26 shows the P-values of the cyphertexts in round 1. All the points in the third plot are located under the level of significance. On the other hand, figure 27 only shows some issues in the second round.

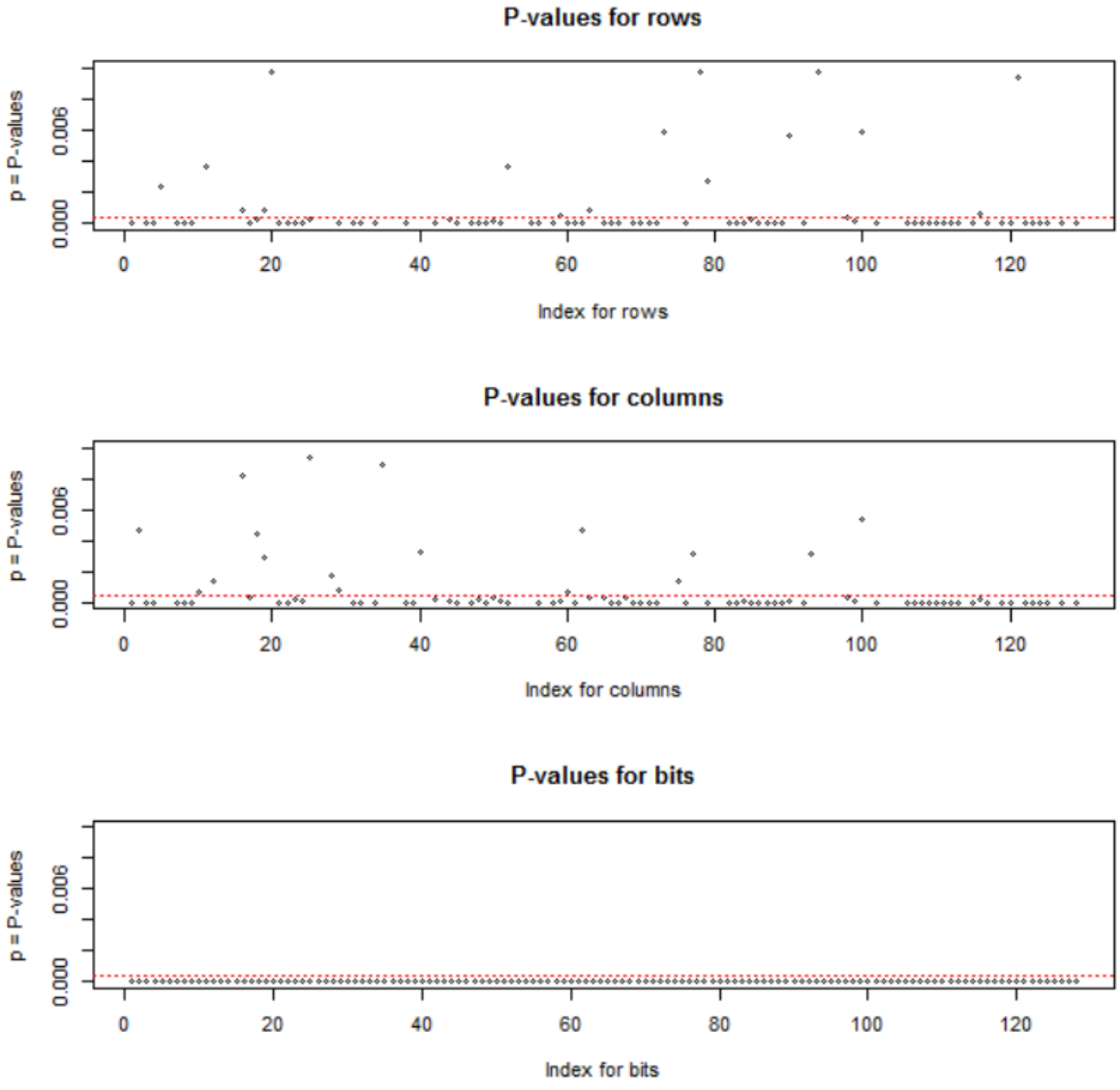

Figure 26:: Ciphertext independence test P-values for round1

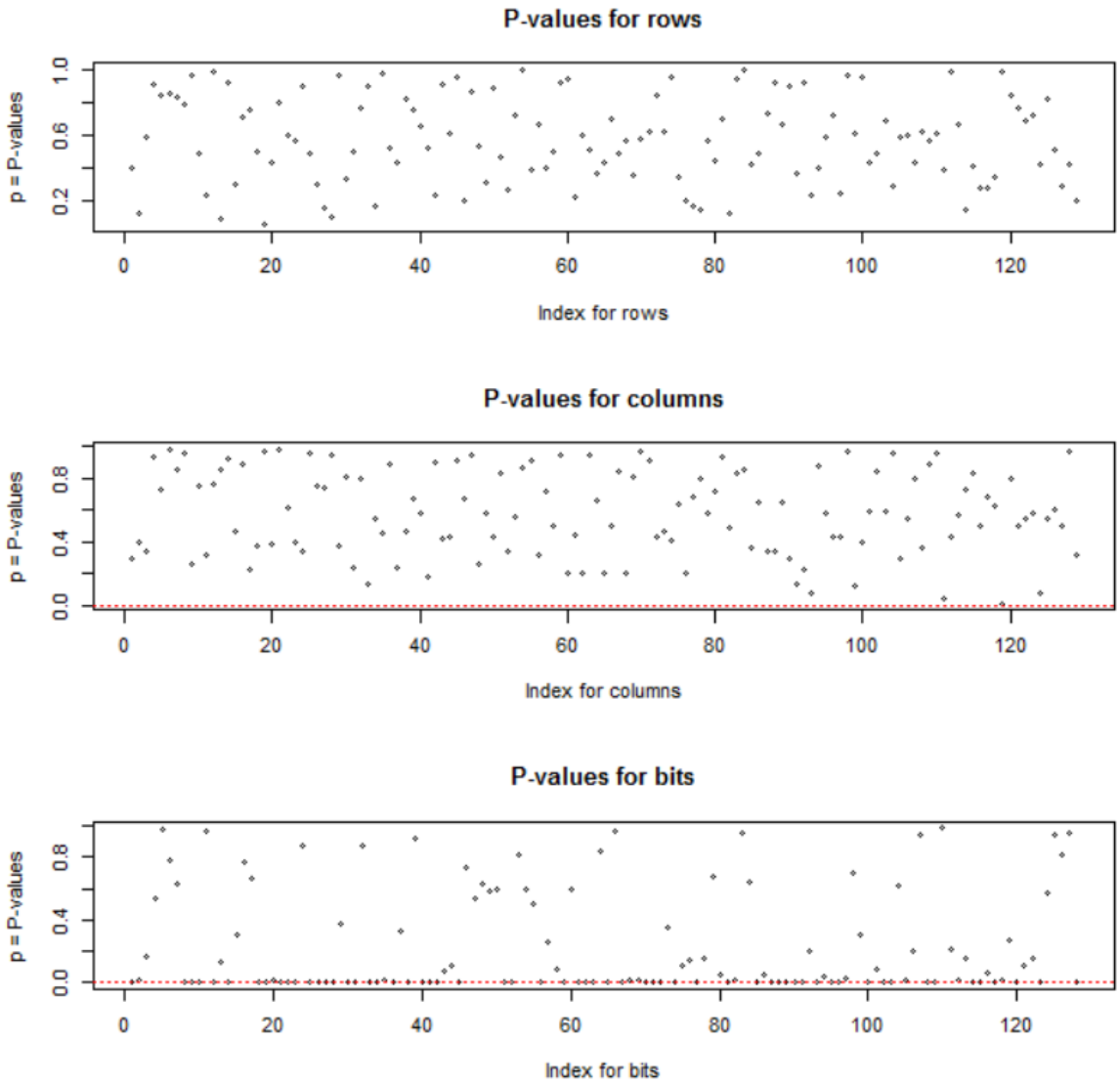

Figure 27: Ciphertext independence test P-values for round2

Figure 28 and 29 shows the P-values of applying the ciphertext independence test using the Chi-squared test method.

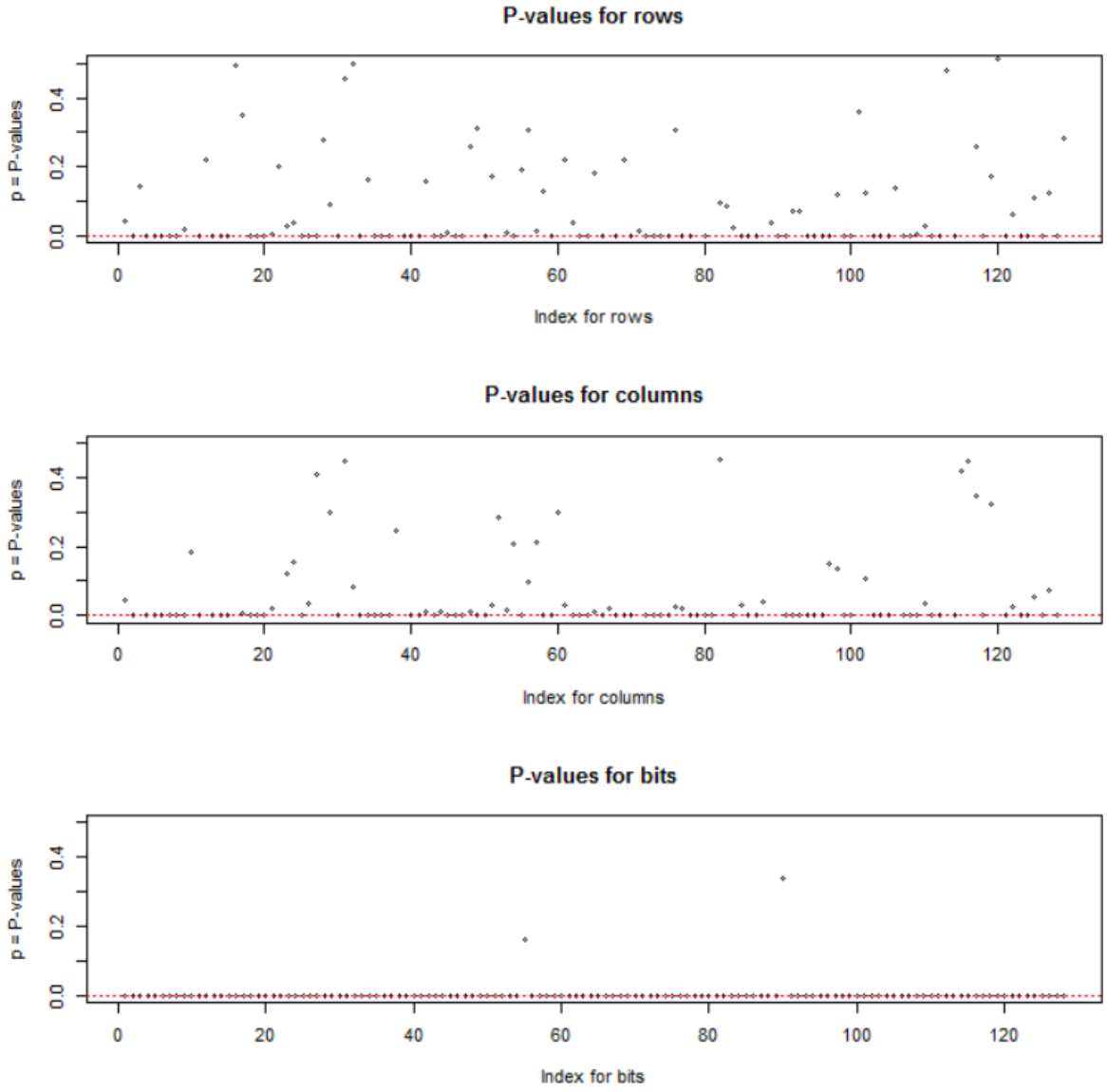

Figure 28: Ciphertext independence test P-values for round1 using Chi-squared test

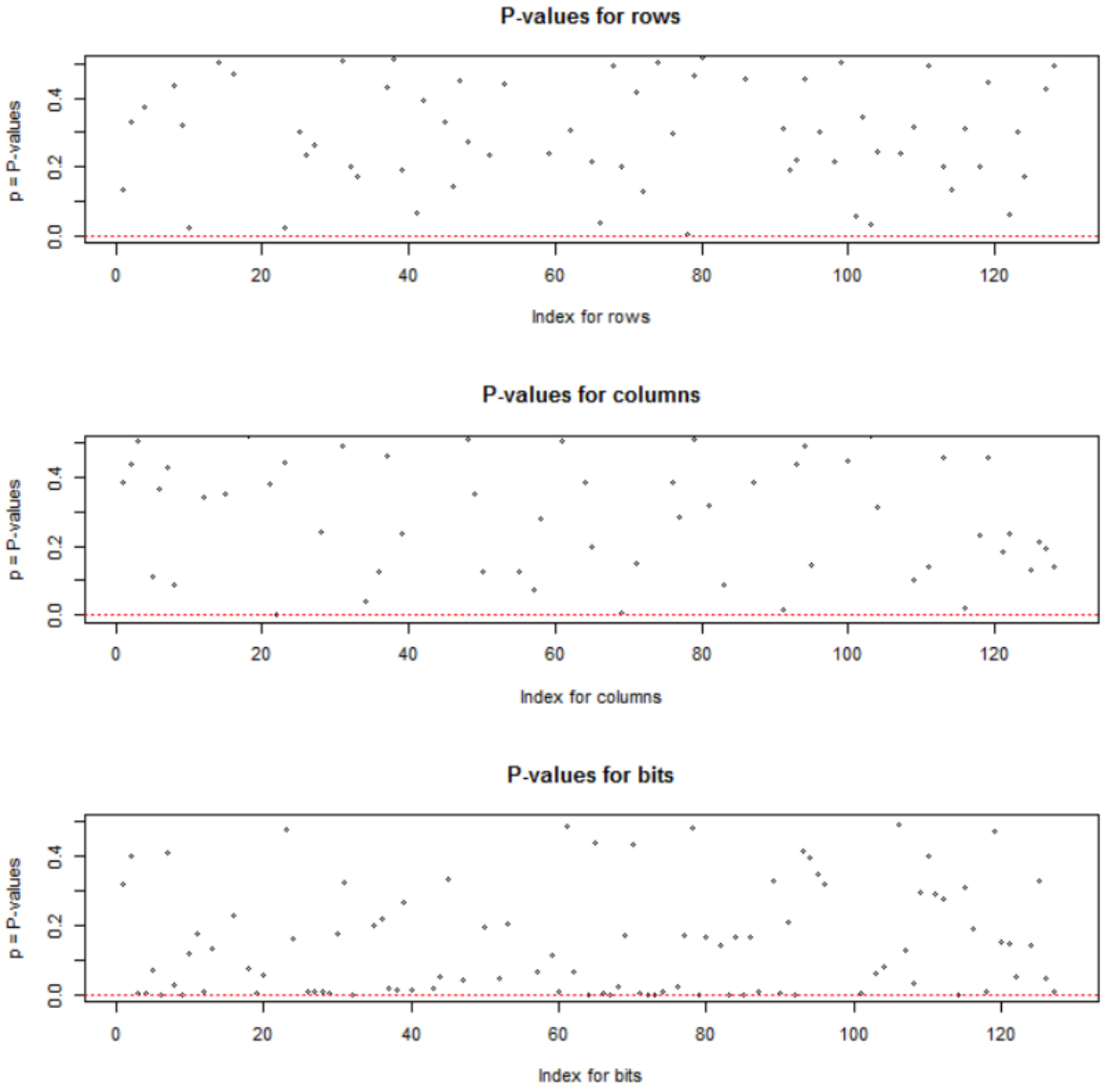

Figure 29: Ciphertext independence test P-values for round2 using Chi-squared test

6.2.6. The Strong Avalanche Test

- Test Purpose

The strong avalanche test's subsets examine all possible ciphertexts produced by flipping each bit of the plaintexts and keys.

- Test Description

When *P* the plaintext is encrypted with the encryption key *K*, The Ciphertext output *C* is then obtained. Let P' is the plaintext produced by flipping one bit in *P*, the original plaintext. Also, let K' is the encryption key produced by flipping one bit in the original encryption key *K*, and C' is the ciphertext output when either a plaintext P' is encrypted with the original key *K*, or the original plaintext P is encrypted with key K'. Thus, the ciphertext C' will be containing the bits that were produced by flipping all possible bits in both *K* and *P*. The strong avalanche test analyzes bit groups from both ciphertext outputs *C* and C' by comparing between each bit group *O* from *C* and the corresponding bit group O' from C' when $|O|=|O'|=g$, where g takes a value of 1, 2, 4, 8, ..., *N*, and *N* is the maximum size of the ciphertext *C*. If $V= O \oplus O'$ and *V* is uniformly distributed, then each bit of the bit group O' differs from the corresponding bit of *O* with probability 1/2 and *V* can be said to be random. The strong avalanche test then decides between two binomial models: H_0 where *V* is uniformly distributed and H_1, where *V* has some other distribution. Figure 30 shows a simple structure for one round from AES. Each Ciphertext *C* consists of 128 bits. If strong Avalanche test was applied on the first ciphertext $C_{1,1}$, both $C_{1,2}$ and $C_{2,1}$ are produced by flipping one bit in the plaintext or the encryption key, and could be considered as C'.

$C_{1,1}$	$C_{1,2}$	....	$C_{1,129}$
$C_{2,1}$	$C_{2,2}$	.....	$C_{2,129}$
⋮	⋮	⋱	⋮
$C_{129,1}$	$C_{129,2}$	.....	$C_{129,129}$

Figure 30: Structure of an AES round

- Applying the Strong Avalanche Test on AES

The results are shown in table 10 and figures 31, 32, 33, and 34 below. Only the first two rounds seem to fail the strong avalanche test. Second round only has problems when using bits in calculating the P-values.

Dimension	Round1	Round2
Rows	27	0
Columns	44	2
Bits	119	27

Table 10: Strong Avalanche test results

Figure 31 and 32 show the frequencies of 1's when applying the strong avalanche test on the first two rounds.

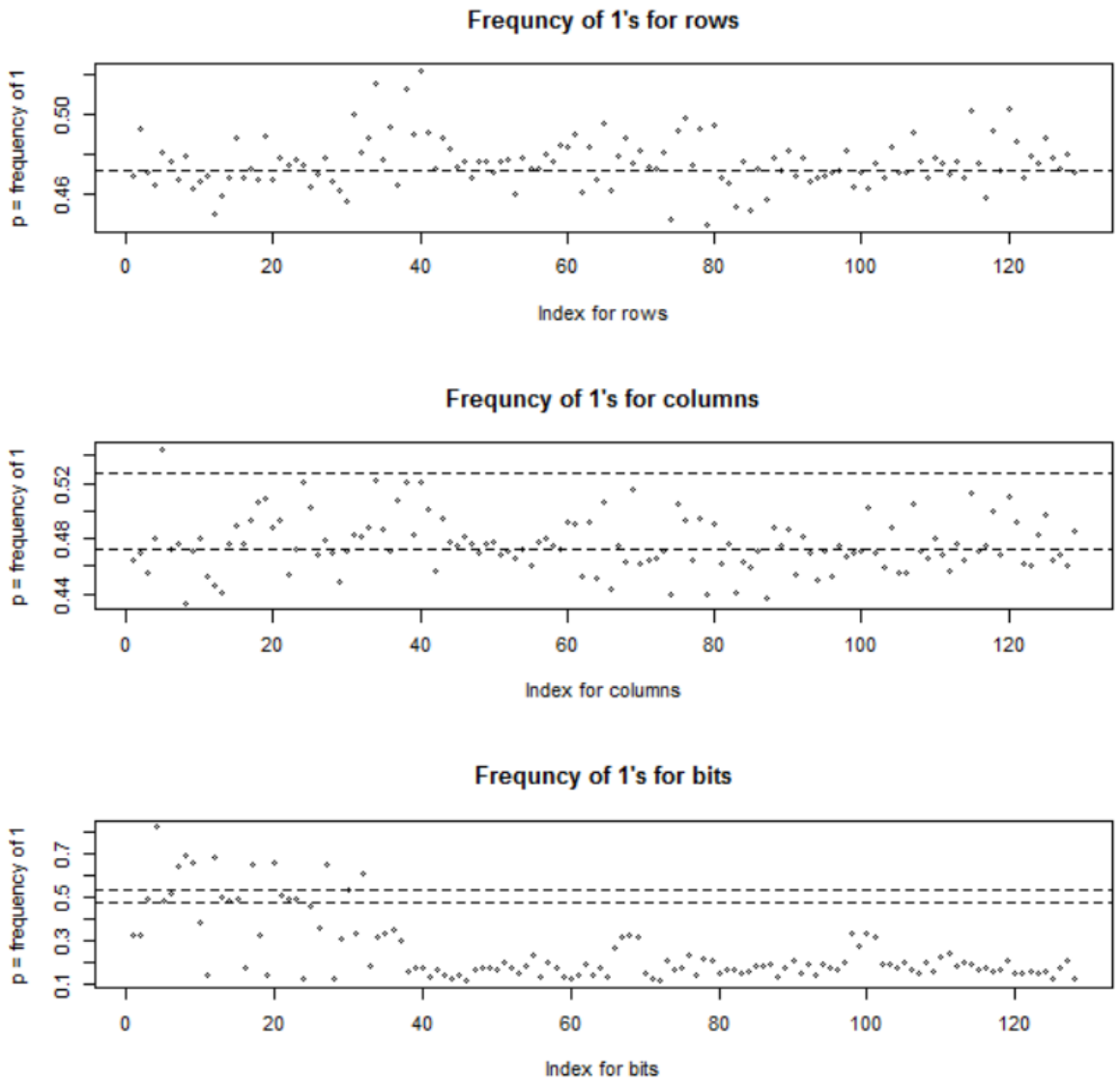

Figure 31: Frequencies of 1's for the strong avalanche test in round 1

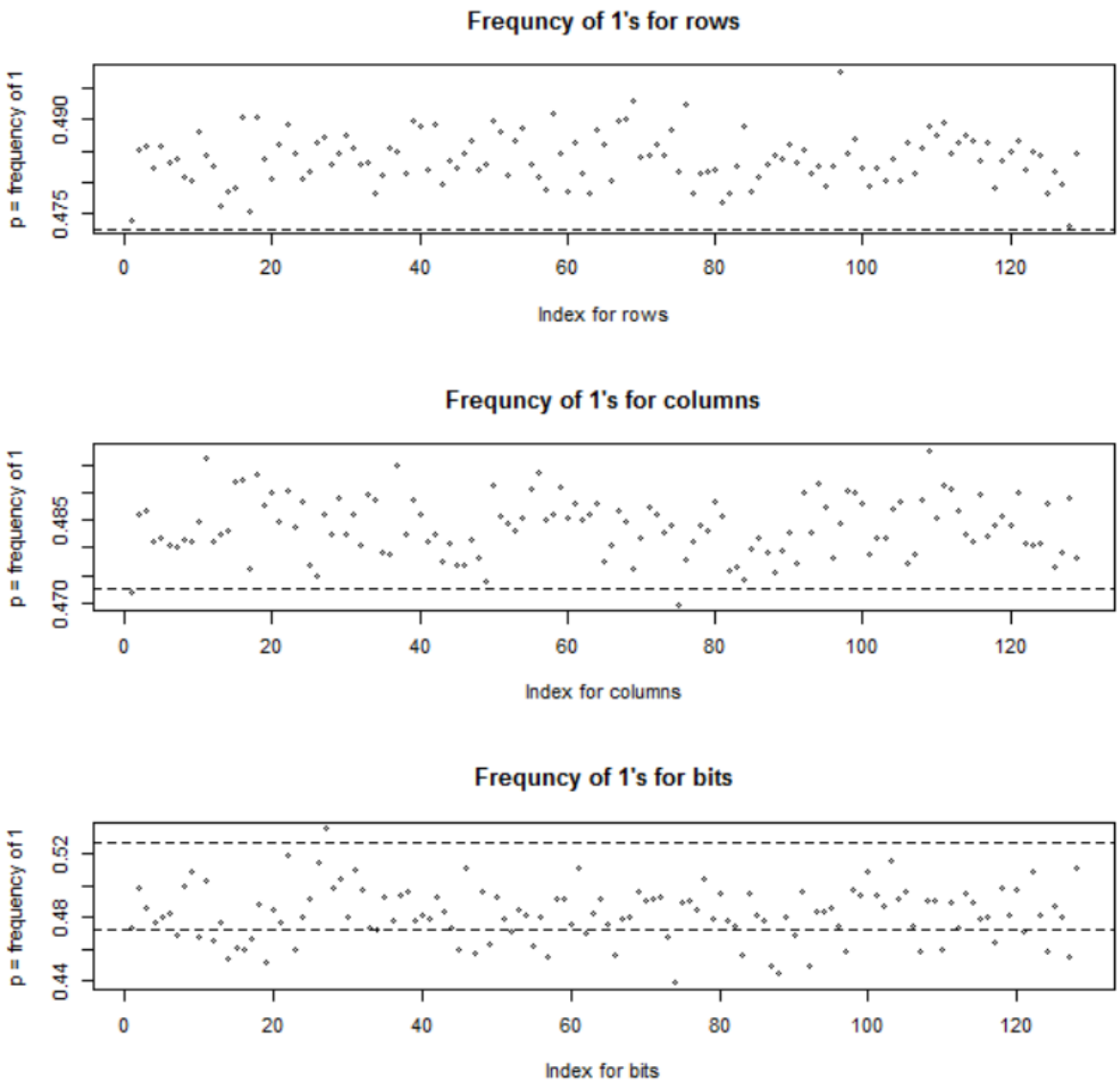

Figure 32: Frequencies of 1's for the strong avalanche test in round 2

Below, there are figure 33 and 34 showing the P-values of applying the strong avalanche test on round 1 and 2 of AES. Figure 33 for first round has plenty of points under the significance level. Figure 34 for round 2 on the other hand has few non-randomness issues.

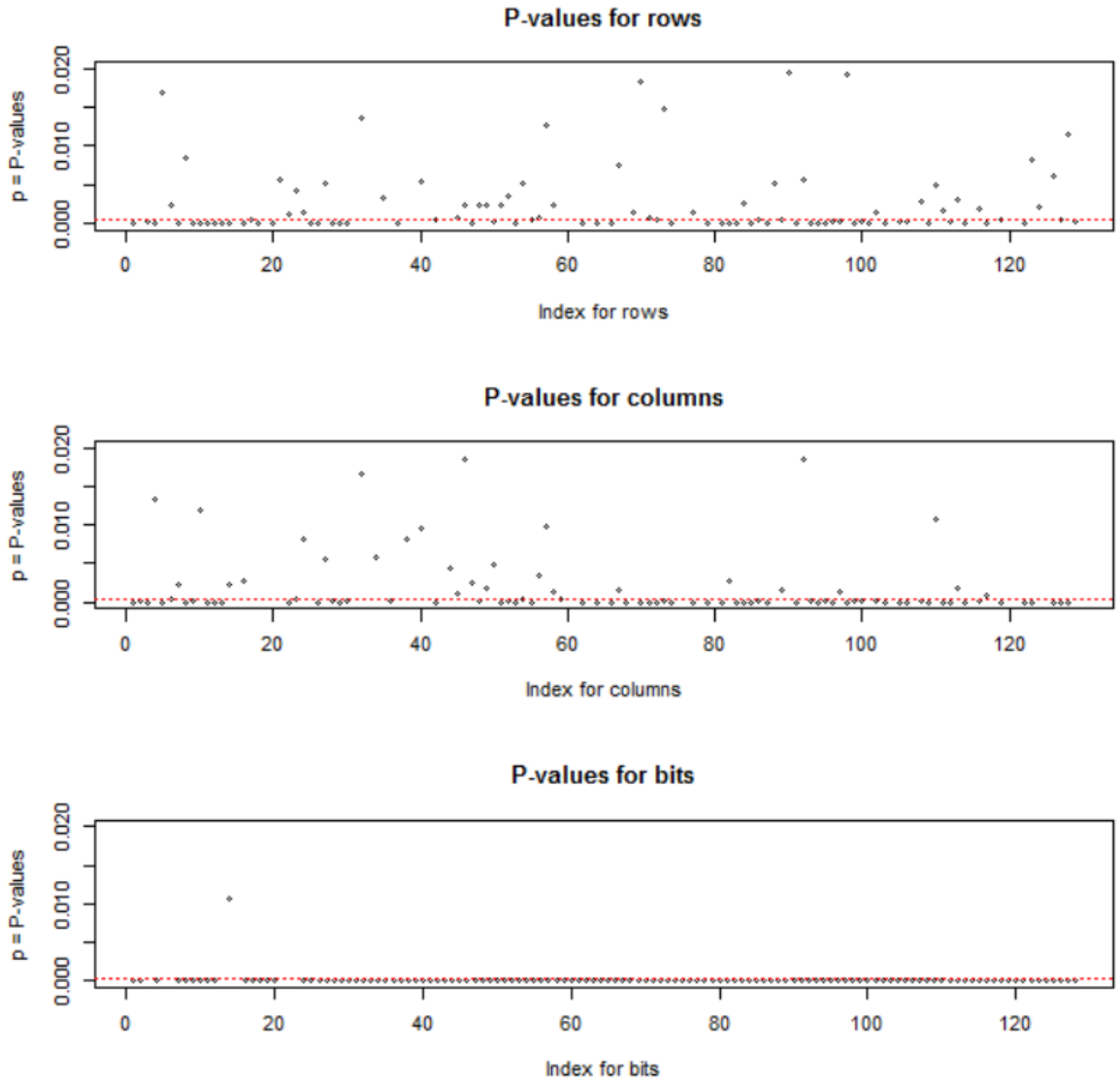

Figure 33: Strong Avalanche Test P-values for round 1

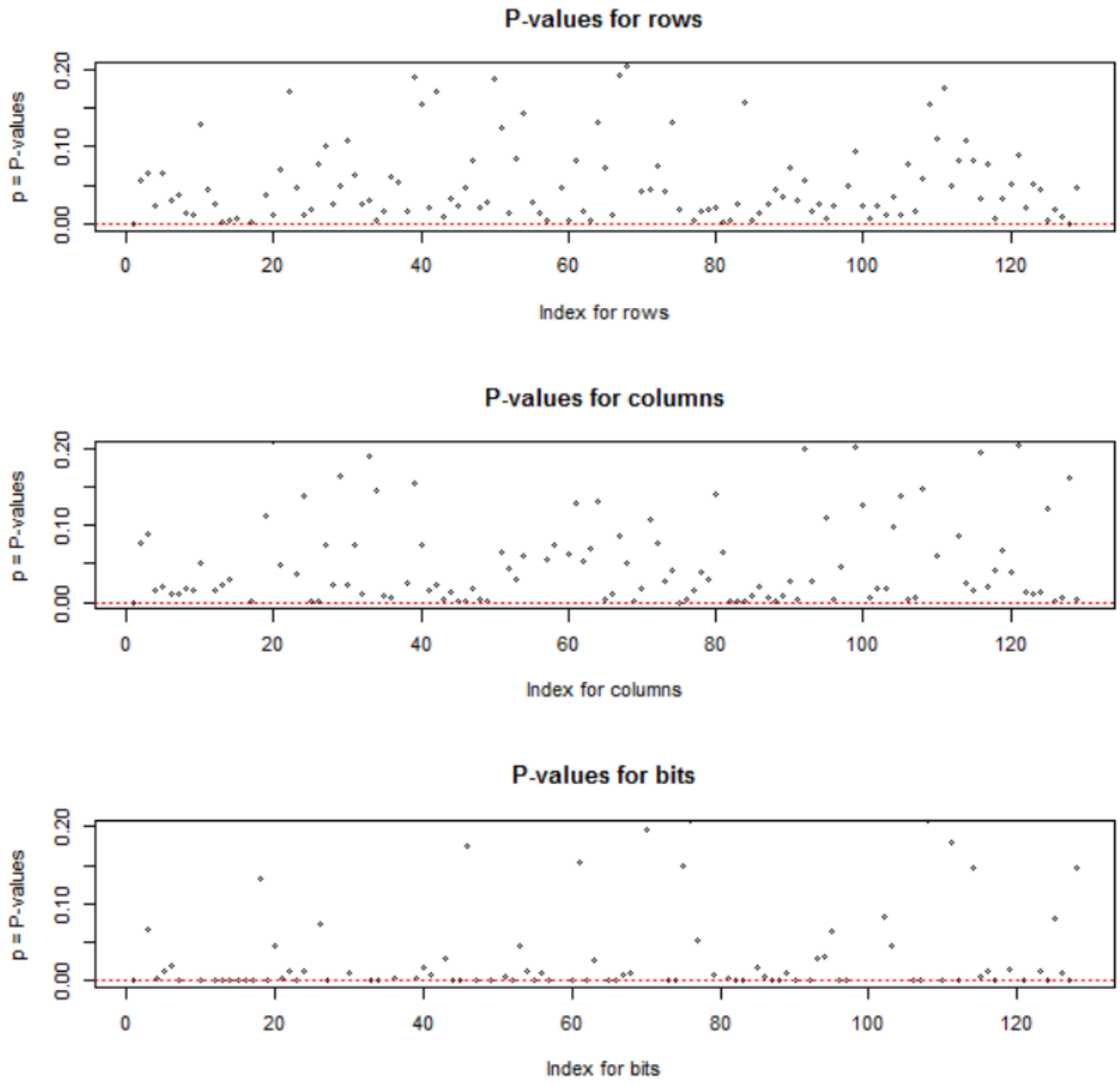

Figure 34: Strong Avalanche Test P-values for round 2

6.2.7. The Uniformity Test

- Test Purpose

The uniformity test examines whether specific bit groups from the ciphertext output *C* are uniformly distributed or not.

- Test Description

Let *C* be the ciphertext output when *P*, a valid plaintext is encrypted with the encryption key *K*. The uniformity test investigates all bit groups of size *g*=1, and *B* bit groups of size *g* from the ciphertext output where *g* takes on the values 2, 4, 8, ..., *N* and *N* is the maximum size of the ciphertext. Let *O* be a bit group produced from randomly chosen positions from *C* when $|O|=g$. If *O* is uniformly distributed, then the ciphertext output *C* can be said to be random. The uniformity test choses between two binomial models: H_0 where *V* is uniformly distributed and H_1, where *V* has some other distribution.

- Applying the Uniformity Test on AES

The results of applying the uniformity test on AES's rounds are shown in table 11, figure 35 and 36 below. The first round is the only round that fails the test and considered not random based on the uniformity test.

Dimension	Round1	Round2
Rows	32	0
Columns	42	0
Bits	128	0

Table 11: Uniformity test results

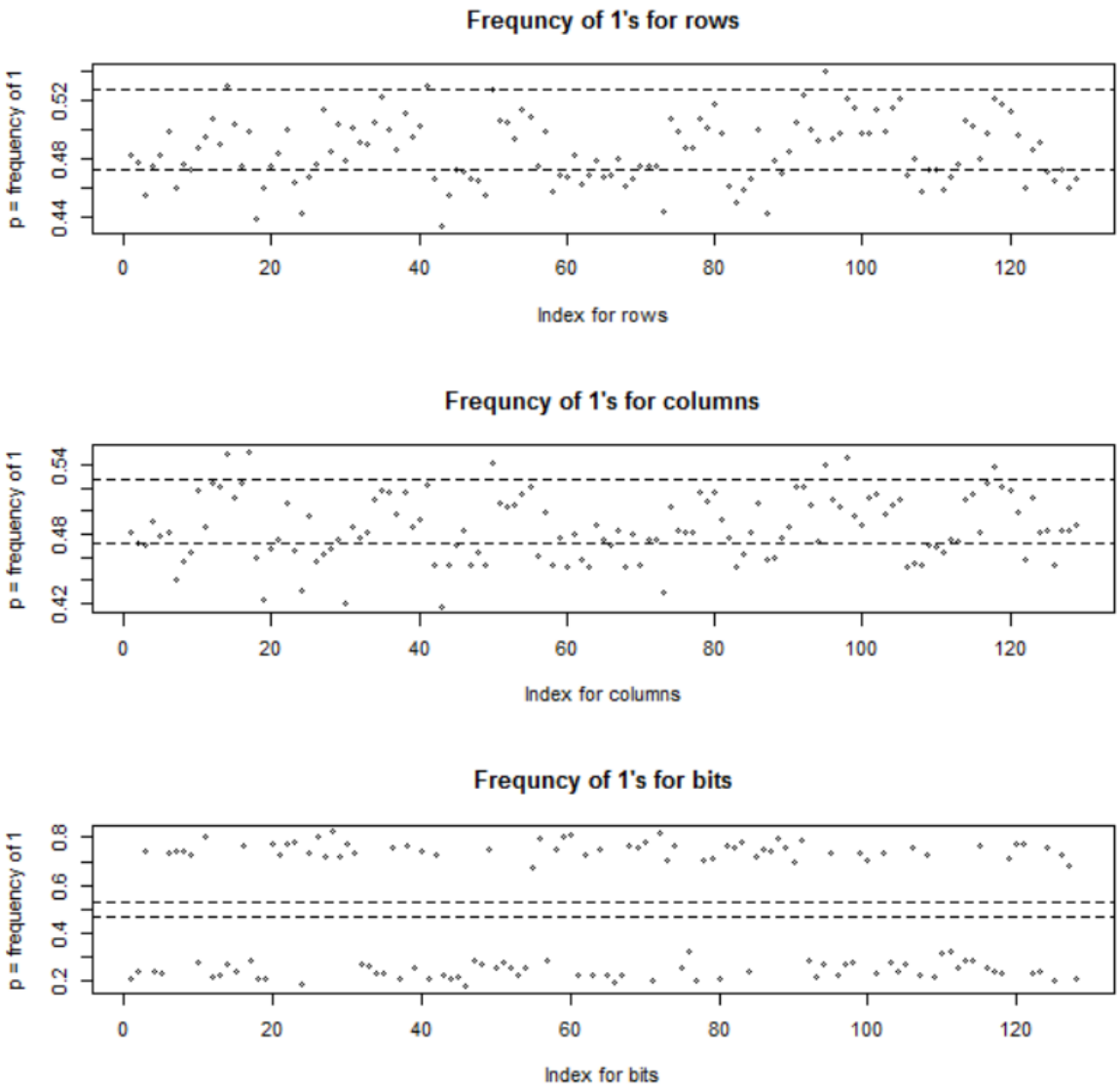

Figure 35: Frequencies of 1's for the uniformity test in round 1

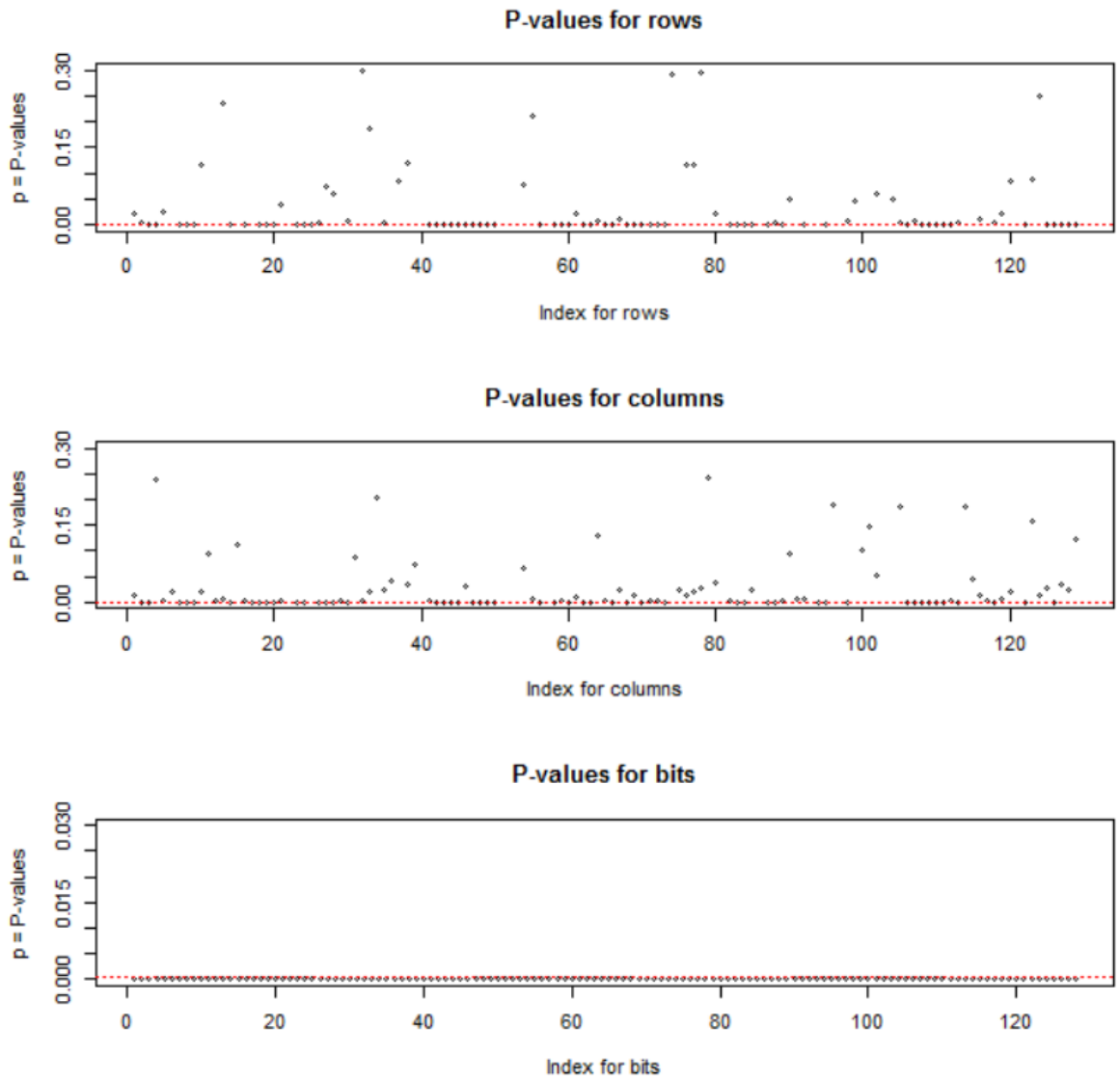

Figure 36: Uniformity test P-values for round1

Table 12, figure 37 and 38 represent applying the uniformity test using the Chi-squared test method. In this case, two rounds failed the uniformity test.

Dimension	Round1	Round2
Rows	43	0
Columns	61	0
Bits	128	11

Table 12: Uniformity test results using Chi-squared test

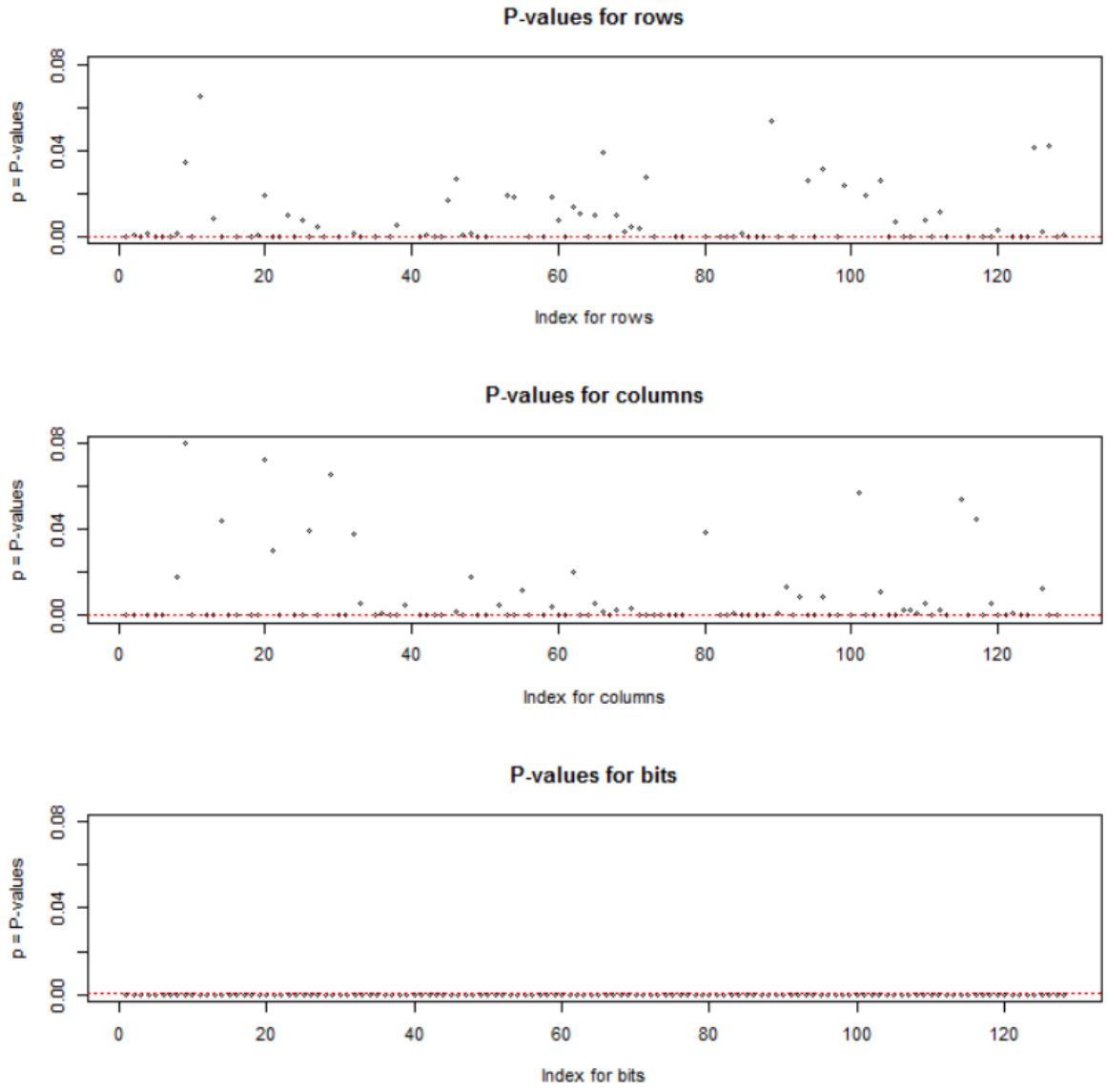

Figure 37: Uniformity test P-values for round1 using Chi-squared test

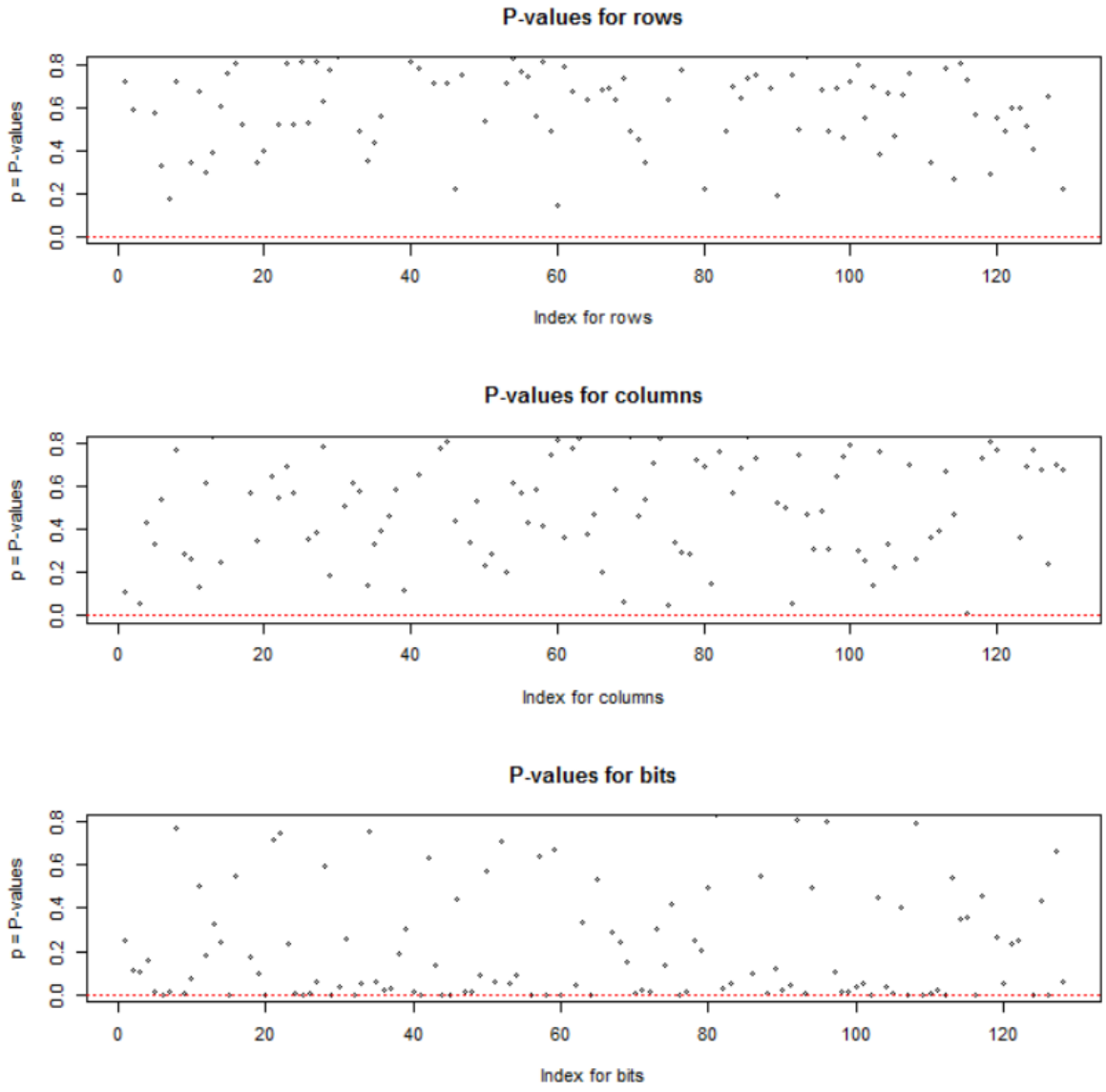

Figure 38: Uniformity test P-values for round2 using Chi-squared test

Chapter 7

Results and Discussion

In the previous chapter, CryptoStat test suite was applied on AES. When using counting the numbers of 1's in each bit position, the first round failed all the tests, but the second round failed the linear approximation test, coincidence test, ciphertext independence test, and strong avalanche test. On the other hand, when using the Chi-squared test method, the first two rounds failed all the tests. Other rounds pass the tests and show acceptable randomness. Figure 39 shows the results of applying the CryptoStat test suite on AES.

Also, using the array based approach allowed us to understand the nature of non-randomness in AES. For example, using bits always shows more non-randomness cases because it has more access to the ciphertexts more than using rows or columns.

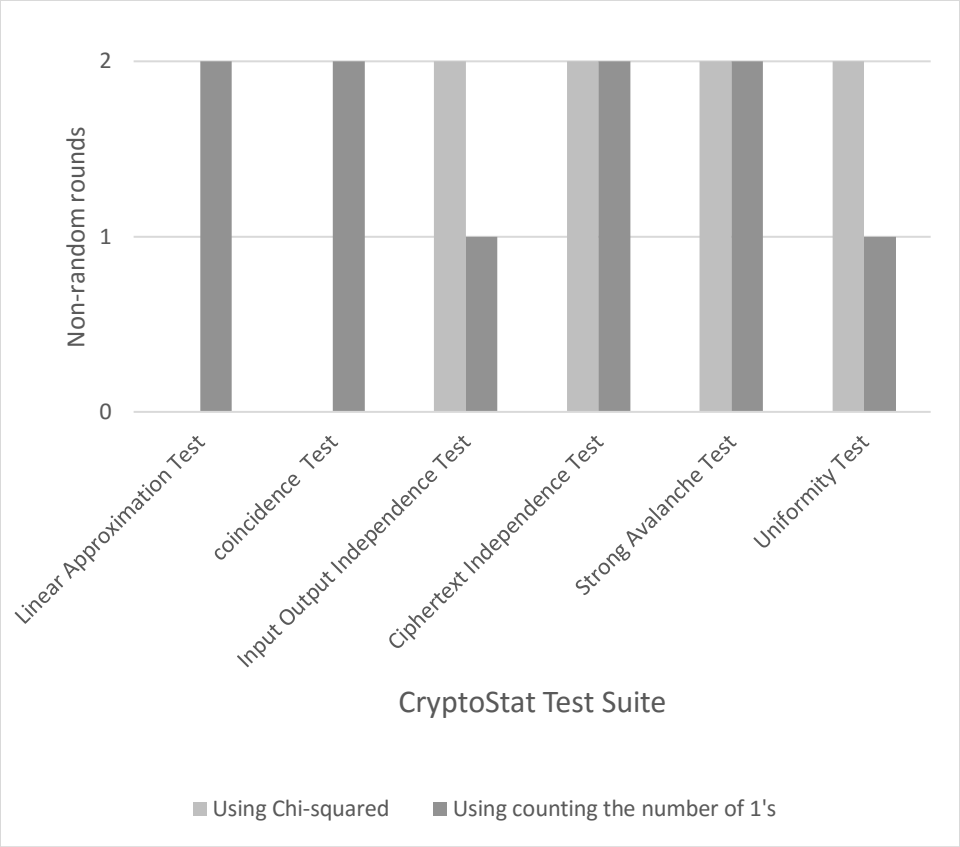

Figure 39: CryptoStat test suite results

To conclude, round 1 and 2 of the AES block cipher are non-random based on CryptoStat test suite. The rounds 3, 4, 5, 6, 7, 8, 9, and 10 are random rounds. Although AES was designed to have 10 rounds to eliminate the non-randomness, the first two rounds were enough.

Chapter 8

Conclusion and Future Work

We hereby conclude the results of this thesis by summarizing the analysis and providing some future work. The focus of this thesis is to evaluate the randomness of the Advanced Encryption Standard (AES) block cipher, which comprises of 10 rounds. CryptoStat is a statistical test suite that evaluates cryptographic functions' randomness, which includes 7 tests: linear approximation test, coincidence test, input output independence test, complement test, ciphertext independence test, strong avalanche test, and uniformity test. Except for the complement test, these tests were applied on AES using an array based approach and hypothesis testing.

The results show that the first two rounds have some randomness issues. On the other hand, the other rounds pass the tests and can be considered random.

Future work can include adding new statistical tests to CryptoStat test suite, and then applying them to AES to investigate its randomness. The array based approach could be used to apply other tests on AES or any other block cipher.

Appendices

Appendix A

Selected R Code

########AES Structure########

```
> str(C.mat.list)
List of 3
 $ A: int [1:129, 1:128] 0 1 0 1 1 1 0 1 0 1 ...
 $ B: int [1:129, 1:128] 0 1 1 1 0 0 0 0 1 0 ...
 $ C: int [1:129, 1:129, 1:10, 1:128] 0 1 0 0 1 1 0 1 0 0 ...
```

######## An array based approach#######

##Test 1: counting all the 1s and applying 1 binomial test##

```
load("C.mat.list")

C.mat <- C.mat.list$C

Ind <- c(1,2,4) # 1 for rows, 2 for columns, and 4 for bits

v <- apply(C.mat,3,sum)

n <- prod(dim(C.mat)[Ind])

p <- v/n

SE <- sqrt(0.25/n)

p.val <- 2*(1-pnorm(abs(p-0.5)/SE))*10*100

round(p.val,1)
```

```
##Test 2: counting all the 1s in each column and applying 129 binomial tests##

Ind <- 2 # 1 for rows, 2 for columns, and 4 for bits

Dim.v <- dim(C.mat)[c(Ind,3)]

p.val.mat <- matrix(0,Dim.v[1],Dim.v[2])

n <- prod(dim(C.mat)[-c(Ind,3)]) # sample size for each test

SE <- sqrt(0.25/n)

k <- prod(Dim.v) # number of comparisons

if (Ind>3) Ind <- Ind - 1

for (Round in 1:10) {

  C.mat.R <- C.mat[,,Round,]

  v <- apply(C.mat.R, Ind, sum)

  p <- v/n

  p.val.mat[,Round] <- 2*(1-pnorm(abs(p-0.5)/SE))*k*100

}

apply((p.val.mat < 5),2,sum)

a <- apply(p.val.mat,2,min)

round(a,1)
```

```
##Test 3: counting all the 1s in each cell and applying 129*129 binomial tests for each round##

p.val.f <- function(mat){

 Ind <- 2 #1 for rows, 2 for columns, and 4 for bits

 Dim.v <- dim(C.mat)[c(Ind,3)]

 p.val.mat <- matrix(0,Dim.v[1],Dim.v[2])

 n <- prod(dim(C.mat)[-c(Ind,3)]) # sample size for each test

 SE <- sqrt(0.25/n)

 k <- prod(Dim.v) # number of comparisons

 for (Round in 1:10) {

  C.mat.R <- C.mat[,,Round,]

  v <- apply(C.mat.R, Ind, sum)

  p <- v/n

  p.val.mat[,Round] <- 2*(1-pnorm(abs(p-0.5)/SE))*k*100

 }

 p.val.mat}

load("C.mat.list")

C.mat <- C.mat.list$C
```

```
for (Round in 1:10) {

  C.mat.R <- C.mat[,,Round,]

  v <- apply(C.mat.R, 1, p.val.f)

  v}

a <-  apply((v < 5),1,sum)

a2 <- matrix(a,129,10)

apply(a2 ,2,sum)

b <- apply(v,1,min)

b2 <- matrix(b,129,10)

round(apply(b2,2,min),1)



############## Plotting  round 1 and round 9#############

Round <- 1

load("C.mat.list")

C.R1.mat <- C.mat.list$C[,,Round,]

par(mfrow=c(3,1))

Label <- c("rows", "columns","bits")
```

```
for (Ind in 1:3) {
  n <- prod(dim(C.R1.mat)[-Ind]) # sample size for each test
  k <- dim(C.R1.mat)[Ind] # number of comparisons
  SE <- sqrt(0.25/n)
  freq.p.v <- apply(C.R1.mat, Ind, sum)
  freq.p.v <- freq.p.v/n
  plot(freq.p.v,xlab=paste("Index for",Label[Ind]), ylab="p = frequency of 1",
       main=paste("Frequncy of 1's for",Label[Ind]))
  z.val <- qnorm(1-0.05/(2*k), sd=SE)
  abline(h = 0.5+z.val,lty=2)
  abline(h = 0.5-z.val,lty=2)
}
##################Coincidence Test###################
P.val.f <- function(Ind=1,V=c(1,1,1,1)) {
  Match.V.f <- function(t,V=1) {
    all(t==V)
  }
```

```
Coincid.vec.test.f <- function(x,V=1) {
  k <- length(V)
  m <- floor(length(x)/k)
  mat <- matrix(x[c(1:(k*m))],k,m)
  d <- apply(mat,2,Match.V.f,V=V)
  sum(d)
}
Coincid.mat.test.f <- function(mat,V=1) {
  Count.v <- apply(mat,1,Coincid.vec.test.f,V=V)
  sum(Count.v)
}
load("C.mat.list")
C.mat <- C.mat.list$C
Dim.v <- dim(C.mat)[c(Ind,3)]
Dim.test.v <- dim(C.mat)[-c(Ind,3)]
m <- length(V)
n <- Dim.test.v[1]*floor(Dim.test.v[2]/m) # sample size for each test
```

```
k <- prod(Dim.v) # number of comparisons

SE <- sqrt(0.25/n)

p.val.mat <- matrix(0,Dim.v[1],Dim.v[2])

p0 <- 2^(-m)

# Now calculating for each round:

if (Ind>3) Ind <- Ind - 1 # "manual" adjustment for Ind beyond Round index

for (Round in 1:10) {

  C.mat.R <- C.mat[,,Round,]

  # here need to apply coincidence test to the matrix (for now to rows only)

  v <- apply(C.mat.R, Ind, Coincid.mat.test.f,V=V)

  p <- v/n

  p.val.mat[,Round] <- 2*(1-pnorm(abs(p-p0)/SE))*k*100

  # browser()

  # This needs adjustment if Ind is a vector

  # *k because of Bonferroni

  # *100 to get percentages

}
```

```
  p.val.mat # typically 129 by 10 matrix

}

b <- P.val.f(Ind=1)

apply((b < 5),2,sum)



####################Uniformity Test##################

Unif.P.val.f <- function(Ind=1,V=c(1,1,1,1)) {

  Unif.vec.test.f <- function(x,V=V) {

    k <- 32

    m <- floor(length(x)/k)

    mat <- matrix(x[c(1:(k*m))],k,m)

    v1 <- apply(mat,2,sum)

    v2 <- k-v1

    v <- rbind(v1,v2)

    v <- t(v)

    p.v <- rep(0,m)

    for(j in 1:m){
```

```
    p.v[j] <- chisq.test(v[j,])$ p.val
  }
  sum(p.v)
}
Unif.mat.test.f <- function(mat,V=V) {
  Count.v <- apply(mat,1,Unif.vec.test.f,V=V)
  sum(Count.v)
}
load("C.mat.list")
C.mat <- C.mat.list$C
Dim.v <- dim(C.mat)[c(Ind,3)]
Dim.test.v <- dim(C.mat)[-c(Ind,3)]
m <- 32
n <- Dim.test.v[1]*floor(Dim.test.v[2]/m) # sample size for each test
k <- prod(Dim.v) # number of comparisons
SE <- sqrt(0.25/n)
p.val.mat <- matrix(0,Dim.v[1],Dim.v[2])
```

```
 p0 <- 2^(-1)

 # Now calculating for each round:

 if (Ind>3) Ind <- Ind - 1 # "manual" adjustment for Ind beyond Round index

 for (Round in 1:10) {

  C.mat.R <- C.mat[,,Round,]

  # here need to apply coincidence test to the matrix (for now to rows only)

  v <- apply(C.mat.R, Ind, Unif.mat.test.f,V=V)

  p <- v/n

  p.val.mat[,Round] <- 2*(1-pnorm(abs(p-p0)/SE))*k*100

 }

 p.val.mat # typically 129 by 10 matrix

}

a <- Unif.P.val.f (Ind=1)

apply((a < 5),2,sum)
```

Chapter 9

Bibliography

NIST; https://www.nist.gov/.

3gpp.org. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects

3G Security; General Report on the Design, Speification and Evaluation of 3GPP Standard Confidentiality and Integrity Algorithms.

Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J. B., Seurin, Y., & Vikkelsoe, C. 2007. PRESENT: An Ultra-Lightweight Block Cipher, vol. 4727: 450-466. Berlin, Heidelberg: Springer Berlin Heidelberg.

Boutin, C. 2005. Pi Seems A Good Random Number Generator - But Not Always The Best. Purdue University.

Brown, R., Eddelbuettel, D., & Bauer, D.; Dieharder: A Random Number Test Suite.

Daemen, J., Govaerts, R., & Vandewalle, J. 1993. A New Approach To Block Cipher Design; In: Fast Software Encryption: pp. 18-33: Springer-Verlag.

Dent, A. W. & Mitchell, C. 2005. ***User's guide to cryptography and standards***. Boston: Artech House.

Doganaksoy, A., Ege, B., Kocak, O., & Sulak, F. 2010. Cryptographic Randomness Testing of Block Ciphers and Hash Functions. (Institute of Applied Mathematics, Middle East Technical University, Ankara, Turkey).

Goldberg, I. & Wagner, D. 1996. Randomness and the Netscape browser, vol. 21: 66-66. SAN MATEO: MILLER FREEMAN, INC.

Hoyt, K. 2016. ***Structuring Statistical Tests for Validating Encryption: An Array-based Approach***. ProQuest Dissertations Publishing.

Hörmanseder, D. R. & Erik, S. 2007. randomness in cryptography.

Kaminsky, A.; The Coincidence Test: a Bayesian Statistical Test for Block Ciphers and MACs.

Kaminsky, A. & Sorrell, J. 2013. CryptoStat: a Bayesian Statistical Testing Framework for Block Ciphers and MACs. Rochester Institute of Technology, Rochester, NY.

Kessler, G. C. 2015. An Overview of Cryptography. 54.

L'Ecuyer, P. & Simard, R. 2007. TestU01: A C library for empirical testing of random number generators. ***ACM Transactions on Mathematical Software (TOMS)***, 33(4): 1-40.

Lai, X. & Massey, J. 1999. A Proposal for a New Block Encryption Standard. In: EUROCRYPT: pp. 389-404: Springer-Verlag (1991).

Marsaglia, G.; Diehard battery of tests of randomness; http://www.stat.fsu.edu /pub/diehard/linux.tar.gz.

Matthews, J. 2003. Network Security Basics. Cornell university.

NIST. 2001. *Announcing the advanced encryption standard (AES)*. Gaithersburg, MD: Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology.

Paar, C. & Pelzl, J. 2010. ***Understanding cryptography: a textbook for students and practitioners***. Berlin: Springer.

Rao, A. 2007. ***Randomness extractors for independent sources and applications***. ProQuest Dissertations Publishing.

Rukhin, A. L. & Bassham, L. E. 2008. A statistical test suite for random and pseudorandom number generators for cryptographic applications, vol. Revision 1. Gaithersburg, MD: U.S. Dept. of Commerce, National Institute of Standards and Technology.

Schneier, B. 1993. Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish); In: Anderson, R. (ed.) Fast Software Encryption, Cambridge Security: pp. 191-204: Springer-Verlag (1994).

Shamir, A. 2008. SQUASH – A New MAC with Provable Security Properties for Highly Constrained Devices Such as RFID Tags; In Nyberg, K. (ed.) 15th International Workshop on Fast Software

Encryption, vol. vol. 5086: pp. 144-157: Springer, Heidelberg (2008).

Soto, J. & Bassham, L. 2000. Randomness Testing of the Advanced Encryption Standard Finalist Candidates. Gaithersburg, MD: NIST.

Walker, J.; ENT: a pseudorandom number sequence test program; http://www.fourmilab.ch/random/.

Wright, M. A. 2001. The Advanced Encryption Standard. ***Network Security***, 2001(10): 11-13.