Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 626408, 14 pages
http://dx.doi.org/10.1155/2015/626408

Research Article

New Statistical Randomness Tests Based on Length of Runs

Ali Doğanaksoy,[1] Fatih Sulak,[2] Muhiddin Uğuz,[1] Okan Şeker,[1] and Ziya Akcengiz[1]

[1]*Institute of Applied Mathematics, Middle East Technical University, 06800 Ankara, Turkey*
[2]*Mathematics Department, Atılım University, 06836 Ankara, Turkey*

Correspondence should be addressed to Fatih Sulak; fatih.sulak@atilim.edu.tr

Received 27 September 2014; Accepted 17 March 2015

Academic Editor: Anna Vila

Copyright © 2015 Ali Doğanaksoy et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Random sequences and random numbers constitute a necessary part of cryptography. Many cryptographic protocols depend on random values. Randomness is measured by statistical tests and hence security evaluation of a cryptographic algorithm deeply depends on statistical randomness tests. In this work we focus on statistical distributions of runs of lengths one, two, and three. Using these distributions we state three new statistical randomness tests. New tests use χ^2 distribution and, therefore, exact values of probabilities are needed. Probabilities associated runs of lengths one, two, and three are stated. Corresponding probabilities are divided into five subintervals of equal probabilities. Accordingly, three new statistical tests are defined and pseudocodes for these new statistical tests are given. New statistical tests are designed to detect the deviations in the number of runs of various lengths from a random sequence. Together with some other statistical tests, we analyse our tests' results on outputs of well-known encryption algorithms and on binary expansions of e, π, and $\sqrt{2}$. Experimental results show the performance and sensitivity of our tests.

1. Introduction

Random numbers and random sequences are extensively used in many areas such as *game theory, numerical analysis, quantum mechanics,* and *cryptography*. In cryptography, need for random sequences emerges in many different applications such as *challenge and response authentication systems, generation of digital signatures,* and *zero-knowledge protocols*. Among those, the most important feature is key generators which highly depend on random values. Use of weak random values in key generations can cause a leakage in the system and hence an adversary can gain ability to break the whole cryptosystem. Therefore, randomness testing is an essential part of security evaluation of a cryptographic algorithm.

Random sequences and random numbers can be generated by true random sources, such as atmospheric noise and radioactive decay. However, using these sources in an algorithm is unpractical. It causes challenging problems in transmitting and storing large random bits since reproducing outputs of these sources is nearly impossible. Therefore, sequences and numbers, used as a key in cryptographic algorithms such as block ciphers and synchronous stream ciphers, should be pseudorandom, that is, *random looking* sequences of a specific length which are produced by deterministic processes [1]. Since proving randomness of these generators mathematically is nearly impossible, we use statistical randomness test for this purpose. Using statistical tests we try to detect the weaknesses that a generator could have.

Moreover, outputs of encryption algorithms should be indistinguishable from random mappings; that is, it should be random looking. This is another place where pseudorandom sequences play an important role. Also, deciding the round number of a block cipher algorithm, which is an essential part of design, is highly associated with concept of being random looking. Therefore, security of the system highly depends on production or testing of pseudorandom sequences. For these reasons, statistical randomness tests are considered as an important part of evaluating security of cryptographic algorithms.

Statistical tests are designed to test the null hypothesis H_0 which states that the sequence is randomly generated. Testing a binary sequence means that its degree of randomness is evaluated by a statistical test. The conclusion is that the sequence is random or not probabilistic; in other words the hypothesis H_0 is either *accepted* or *rejected*. A statistical test considers a random variable whose distribution function is known. Depending on the distribution, a real number

between 0 and 1, called p value, is calculated. If the p value of the sequence is evaluated as one, we say that the sequence is completely random. On the other hand, the sequence is completely nonrandom, if p value is determined as zero. If the p value exceeds a predefined real number $\alpha \in [0, 1]$, then H_0 is accepted; otherwise, it is rejected.

Usually result of one statistical test is not enough to decide the randomness of sequence. Therefore, it is better to use a collection of statistical tests, called statistical test suites, to measure different behaviours of the sequence under consideration. These suites should be well designed to give trustable results and should not be blindly populated.

In the literature, there exist various statistical test packages. Among those, the most important ones are given in Knuth's book [2], test suite presented by Rukhin [3], DIEHARD [4], CRYPT-X [5], TestU01 [6], and the test suite published by NIST [7] so far. Also there are works focusing on statistical tests individually such as a universal statistical test, stated by Maurer [8], a test based on diffusion characteristic of a block cipher [9], and topological binary test defined by Alcover et al. [10].

In this work, we propose three new statistical randomness tests which depend on famous postulates of Golomb. These tests are named as runs of length one, runs of length two, and runs of length three test. The rest of the paper is formed as follows. In Section 2, we explain Golomb's randomness postulates. Also we discuss run tests given in the literature. In Section 3, we give proofs of our fundamental theorems. Also in order to calculate the probabilities needed, we state corollaries and algorithms for each theorem. In Section 4, we state new run tests and give the pseudocodes. In Section 5, we apply new tests to binary expansion of e, π, and $\sqrt{2}$, which are obtained from NIST package [7] and outputs of five advanced encryption standard competition finalists. In the last part of implementation we generate some nonrandom data sets to emphasize the sensitivity of our tests. Finally, in Section 6, we summarize our results and state the topics for further research.

2. Preliminaries

2.1. Golomb's Randomness Postulates. Deciding the pseudorandomness of a sequence is a difficult task. The base for this task is constructed by Golomb's postulates. These postulates are one of the most important attempts to create some necessary properties for a finite (or periodic) pseudorandom sequence to be random looking. Sequences satisfying following three properties are called *pseudonoise sequence* [11].

Let $S = s_0, s_1, \ldots, s_{n-1}, \ldots$ be an infinite binary sequence periodic with n (or a finite sequence of length n). A run is defined as an uninterrupted maximal sequence of identical bits. Runs of 0's are called *gap*; runs of 1's are called *block*. R1, R2, and R3 are Golomb's randomness postulates which are given as follows.

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

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

(R3) The autocorrelation function $C(t)$ should be two-valued. That is, for some integer K and for all $t = 0, 1, 2, \ldots, n-1$,

$$C(t) = \sum_{i=0}^{n-1} (-1)^{s_i + s_{i+t}} = \begin{cases} n & \text{if } t = 0 \\ K & \text{if } 1 \le t \le n-1. \end{cases} \quad (1)$$

The first postulate states that, in an n-bit sequence, the difference of number of ones and zeros should be 1 or 0. In other words, the number of ones in a sequence, that is, weight of the sequence, should be approximately $n/2$. Frequency test, which measures the difference of number of ones and zeros in an n-bit sequence, is defined to check the first postulate of Golomb. Balancedness is a fundamental feature for an algorithm's output. Therefore, frequency test is used as an initial step for almost all test suites. If an algorithm fails the frequency test, then other tests are not even applied.

The second postulate of Golomb is about number of runs in sequences. Tests, which deal with number of runs, are called run tests and these are also included in many test suites as the frequency test. Since calculating the expected number of runs of specified length in a random sequence is a difficult task (especially when specified length becomes large), most of test suites consider only the total number of runs and do not consider the number of runs of different lengths.

Lastly, the third postulate gives information about amount of similarities between the sequence and shifted version of it. If S is a random looking sequence, the autocorrelation should be constant; that is, correlation between ith and $(i + t)$th bits should give no information about the sequence for $t = 1, 2, \ldots, (n-1)$. In this paper, we mainly focus on the first and second postulates, and the last one is not a matter of concern.

These postulates are theoretical, but difficult to check. Inspired by these postulates, we define new statistical randomness tests which are practical. In order to give the definitions, we calculate the exact probabilities. Before explaining these tests, first we give the mathematical background in order to compute the probabilities that we use in the following Section 3.

2.2. Run Test. Run tests depend on Golomb's second postulate and investigate number of runs in a sequence and their distribution. Run tests take place in most of the test suites. Almost all of these suites, run tests, consider only the total number of runs in a sequence. The most important ones of these are the suites given in [2, 4, 6, 7].

Knuth [2] and DIEHARD [4] test suites define the run test on random numbers. They define runs as *runs up* and *runs down* in a sequence. To illustrate their definition, consider a sequence of length 10, $S_{10} = 138742975349$. Runs are

indicated by putting a vertical line between s_j's when $s_j > s_{j+1}$. Hence, runs of the sequence 138742975349 can be seen as |138|7|4|29|7|5|349|. In other words, the run test examines the length of monotone subsequences. TestU01 [6] defines *run and gap tests* for testing the randomness of long binary stream of length n. This test collects runs of 1's and 0's until the total number of runs is $2r$. Then, for each length $j = 1, 2, \ldots, k$ the number of runs of 1's and 0's of length in this collection is counted and recorded. Then χ^2 test is applied on these counts. *Longest run of 1's test* is also defined for the collection of strings of length m which are obtained from the original long binary string of length n.

NIST [7] test suite consists of firstly 16 and then 15 various statistical tests. After its first publication, some revisions are made. In 2004, it is discovered that test setting of discrete fourier transform test and lempel-ziv test were wrong [12] and new test, which can be used instead of lempel-ziv test, is defined in [13] and correction of overlapping template matching is stated in 2007 [14].

In the suite, 2 of 15 tests are variations of run tests. They are called run test and longest run of ones in a block test. The first one deals with the total number of runs in a sequence. It calculates the total number of runs in a sequence and determines whether it is consistent with the expected number of runs, which is supposed to be close to $n/2$ in a sequence or not. The second one determines whether the longest run of ones in the sequence is consistent with the length of the longest runs of ones which is in a random sequence. In NIST test suite the reference distributions for the run tests are a χ^2 distribution.

In test suite, NIST assumed that sequence of length n is of order 10^3 to 10^7. For this reason, asymptotic reference distributions were derived and used for their tests. But, asymptotic reference distribution is misleading for smaller values of n; as stated 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". In other words, asymptotic reference distributions can lead to some errors in testing short sequences such as outputs of block ciphers or hash functions. In 1999, to overcome this problem, Soto and Bassham [15] propose to concatenate short sequences. This method is used for testing the randomness of Advanced Encryption Standard candidates. Another method has been proposed by Sulak et al. [16], in which distribution functions are used in NIST test suite, replaced by exact distribution and a similar method is used for producing the p values.

In this paper we use the method stated in [16]; thus we need the exact probabilities and exact distribution of tests statistics. Finding the number of sequences having a specified number of runs of length i is a hard problem. We find the number using combinatorial formulas. After that we calculate the desired probabilities by dividing the calculated number by the total number of sequences of length n. Calculating the exact probabilities of the number of runs of length i in a sequence enables us to define the new run tests. We calculate the probabilities for number of runs of lengths one, two, three and we give the detailed information in the following chapter. However, as the length grows, calculations are getting complex and time required for these calculations grows exponentially. Therefore tests involving number of runs of length j ($j > 3$) are unpractical for statistical test suites.

3. Computation of Probabilities

In this chapter, we give the theorems to find the number of sequences with specified properties and hence state the exact probabilities. The probabilities depend on the number of existing shorter runs. That is, probabilities for the number of runs of length two depends on both total number of runs and number of runs of length one; similarly number of runs of length three depends on total number of runs and number of runs of lengths one and two and so on. Since they have some dependencies with other variables, these probabilities are not directly used in tests. Therefore, after stating each theorem we give the corollaries and the algorithms to find the exact probabilities which are needed for describing the tests.

In the calculations of probabilities we frequently use the following combinatorial formulas.

Fact 1 (number of nonnegative integer solutions of linear equation [17]). The number of nonnegative integer solutions of $x_1 + x_2 + \cdots + x_r = n$, $n \in \mathbb{Z}^+$, is $\binom{n+r-1}{r-1}$.

Fact 2. The number of positive integer solutions of $x_1 + x_2 + \cdots + x_r = n$, $n \in \mathbb{Z}^+$, is $\binom{n-1}{r-1}$.

Proof. With the substitution $x_i = x_i' + 1$ we get

$$\begin{aligned} \left(x_1' + 1\right) + \left(x_2' + 1\right) + \cdots + \left(x_r' + 1\right) &= n, \\ x_1' + x_2' + \cdots + x_r' &= n - r. \end{aligned} \tag{2}$$

From Fact 1 it follows that the number of solutions is

$$\binom{(n-r) + (r) - 1}{r-1} = \binom{n-1}{r-1}. \tag{3}$$

□

3.1. Number of Runs. In the rest of the paper we denote the total number of runs and number of runs of lengths one, two, and three as r_t, r_1, r_2, and r_3 and we use samples of these variables, r, l_1, l_2, and l_3, respectively. We denote the probability of randomly chosen binary sequence with r runs by $\Pr(r_t = r)$. In the same way, $\Pr(r_i = l_i)$ is the probability of randomly chosen binary sequence with l_i runs of length i. Also we use subscripts $S_1, S_2, \ldots, S_m$ to differentiate the blocks of a long sequence or outputs of block ciphers and hash functions. Lastly, L_1, L_2, and L_3 are used to state the set of number of runs of lengths one, two, and three in the sequences accordingly. That is, $L_i = \{l_i^1, l_i^2, \ldots, l_i^m\}$ and l_i^j corresponds the number of runs of length i in the jth sequence.

Moreover, in order to illustrate the runs of a sequence we use the equation $x_1 + x_2 + \cdots + x_r = n$ for a sequence with length n and having r runs. x_i $(i = 1, 2 \ldots, r)$ represents the number

of bits in ith run. An important property of this illustration is that it gives no information about content of x_i's; that is, x_i can be a run of 0's or 1's. Thus, each positive integer solution of the equation $x_1 + x_2 + \cdots + x_r = n$ corresponds to two sequences: one starts with 1 and the other starts with 0. Hence, the number of sequences with length n and having exactly r runs is $2\binom{n-1}{r-1}$ by Fact 2.

Example 1. Let $S = 011000100111110011000111010100 00$ be a binary sequence of length 32 and having 15 runs. Then,

$$x_1 + x_2 + \cdots + x_{15} = 32,$$

$$\underbrace{0}_{x_1}\ \underbrace{11}_{x_2}\ \underbrace{000}_{x_3}\ \underbrace{1}_{x_4}\ \underbrace{00}_{x_5}\ \underbrace{11111}_{x_6}\ \underbrace{00}_{x_7}\ \underbrace{11}_{x_8}\ \underbrace{000}_{x_9}\ \underbrace{111}_{x_{10}}\ \underbrace{0}_{x_{11}}\ \underbrace{1}_{x_{12}}\ \underbrace{0}_{x_{13}}\ \underbrace{1}_{x_{14}}\ \underbrace{0000}_{x_{15}},$$

$$\begin{aligned} &x_1 = 1, \quad x_2 = 2, \quad x_3 = 3, \quad x_4 = 1, \\ &x_5 = 2, \quad x_6 = 5, \quad x_7 = 2, \quad x_8 = 2, \\ &x_9 = 3, \quad x_{10} = 3, \quad x_{11} = 1, \quad x_{12} = 1, \\ &x_{13} = 1, \quad x_{14} = 1, \quad x_{15} = 4. \end{aligned} \tag{4}$$

Probabilities are calculated in a similar way as in [16]. The main difference is that, in the previous approach, sequences are viewed in a circular form. Probabilities depend on weight of the sequence and parity of number of runs. We calculate the probabilities with the above notation, which is not based on circular form, and they depend on the number of runs and number of shorter runs.

Theorem 2. *Let $S = s_1, s_2, \ldots, s_n$ be a binary sequence of length n having total of R runs; then*

$$\Pr(r_t = r) = \frac{\binom{n-1}{r-1}}{2^{n-1}}. \tag{5}$$

Proof. We can illustrate the sequence of length n, having r runs, as follows:

$$x_1 + x_2 + \cdots + x_r = n. \tag{6}$$

From Fact 2 the number of all binary sequences $S = s_1, s_2, \ldots, s_n$ of length n, having total number of r runs, is $2\binom{n-1}{r-1}$. Since there are 2^n sequences, probability of a randomly chosen such sequence to have exactly r runs is

$$\Pr(r_t = r) = \frac{2 \cdot \binom{n-1}{r-1}}{2^n}. \tag{7}$$

□

3.2. Number of Runs of Length One. In this section, probabilities for a n-bit sequence having l_1 runs of length one is given in a combinatorial approach. We use the illustration defined in Section 3.1 to compute the number of sequences having total of r runs, l_1 of which are of length one, and hence we calculate the probabilities. Then we state the first new run test depending on the idea of Golomb's second postulate in the next chapter.

Theorem 3. *The probability of randomly chosen binary sequence $S = s_1, s_2, \ldots, s_n$ with length n, having total of r runs, l_1 of which are runs of length one, is*

$$\Pr(r_t = r, r_1 = l_1) = \frac{\binom{n-r-1}{r-l_1-1} \cdot \binom{n}{l_1}}{2^{n-1}}. \tag{8}$$

Proof. As in the proof of the Theorem 2, we illustrate the sequence as follows:

$$x_1 + x_2 + \cdots + x_r = n. \tag{9}$$

Let us first assume that the last l_1 runs are the runs of length one and the rest are of at least length two. That is,

$$\begin{aligned} &x_{r-l_1+1} = \cdots = x_{r-1} = x_r = 1, \\ &x_1 + x_2 + \cdots + x_{r-l_1} + \overbrace{1 + 1 + \cdots + 1}^{l_1} = n, \\ &x_1 + x_2 + \cdots + x_{r-l_1} = n - l_1. \end{aligned} \tag{10}$$

Notice that, here, $x_i \geq 2$, so we use the change of variable $y_i = x_i - 2$ for $i = 1, 2, \ldots, r - l_1$. Consider

$$\begin{aligned} &(x_1 - 2) + (x_2 - 2) + \cdots + (x_{r-l_1} - 2) = n - l_1 - 2(r - l_1), \\ &y_1 + y_2 + \cdots + y_{r-l_1} = n - 2r + l_1. \end{aligned} \tag{11}$$

The number of sequences having conditions, which are stated above, is equal to the number of nonnegative solutions of (11). Consequently, by the Fact 1, number of desired solutions is

$$\binom{n-r-1}{r-l_1-1}. \tag{12}$$

Selection of l_1 runs of length 1 gives us a factor of $\binom{r}{l_1}$. Since each positive integer solution of (9) corresponds two sequences (one starts with 1; the other starts with 0), 2 is stated as factor also. Therefore, the number of all binary sequences of length n, having total number of r runs, l_1 of which are of length one, is equal to $2\binom{n-r-1}{r-l_1-1}\binom{r}{l_1}$. Hence probability of a randomly chosen such sequence to have exactly r runs, l_1 of which are of length one, is

$$\Pr(r_t = r, r_1 = l_1) = \frac{2 \cdot \binom{n-r-1}{r-l_1-1} \cdot \binom{r}{l_1}}{2^n}. \tag{13}$$

□

Number of sequences having r runs, l_1 of which are of length one, can be found using the formula above. Our aim is to compute total number of sequences of length n having l_1 runs of length one without depending on the total number of runs. In order to compute aimed probabilities we use Corollary 4.

$l_1 \leftarrow 1, r \leftarrow 1, N_1(l_1) \leftarrow 0,$
while $l_1 \leq n$ **do**
  while $r \leq n$ **do**
    $N_1(l_1) \leftarrow N_1(l_1) + \binom{n-r-1}{r-l_1-1}\binom{r}{l_1}/2^{n-1}$
    $r \leftarrow r+1$
  end while
  $l_1 \leftarrow l_1 + 1$
end while
return N_1

Algorithm 1: Calculating $\Pr(r_1 = l_1)$ for $l_1 = 0, 1, \ldots, n$.

Corollary 4. *Let $N_1(l_1)$ denote the number of sequences with exactly l_1 runs of length one. Then,*

$$N_1(l_1) = \sum_{r=1}^{n} 2 \cdot \binom{n-r-1}{r-l_1-1} \cdot \binom{r}{l_1}. \tag{14}$$

Since the number of all sequences of length n is 2^n, probabilities follow immediately:

$$\Pr(r_1 = l_1) = \frac{N_1(l_1)}{2^n}. \tag{15}$$

Moreover, using Algorithm 1 we calculate the probabilities for a sequence of length n and l_1 runs of length one so that we can investigate number of length one independently.

After finding the exact probabilities we calculate the subinterval probabilities. Following example shows the calculations of subinterval probabilities for 128-bit sequences.

Example 5 (calculating the subinterval probabilties).

Step 1. Calculate $N_1(l_1)$ for $l_1 = 0, 1, 2, \ldots, 128$ by using Corollary 4 and Algorithm 1.

Step 2. Determine subintervals such that; $(\alpha_0, \alpha_1), (\alpha_1, \alpha_2), \ldots, (\alpha_4, \alpha_5)$ such that, $\Pr_i(\alpha_i < R < \alpha_{i+1}) \approx 0, 2$. In our example subinterval probability can be calculated as follows;

$$\text{Box } 1 = \sum_{l_1=0}^{27} \Pr_1(r_1 = l_1), \qquad \text{Box } 2 = \sum_{l_1=28}^{31} \Pr_1(r_1 = l_1),$$
$$\text{Box } 3 = \sum_{l_1=31}^{34} \Pr_1(r_1 = l_1), \qquad \text{Box } 4 = \sum_{l_1=35}^{38} \Pr_1(r_1 = l_1),$$
$$\text{Box } 5 = \sum_{l_1=39}^{128} \Pr_1(r_1 = l_1). \tag{16}$$

Step 3. Finally, we get the Table 1 for subinterval probabilities.

In the same way we calculate the subinterval probabilities for different block lengths. All subinterval probabilities for runs of length one test can be seen in Table 2.

Example 6. Let S_n be a random sequence of length 8, having 4 runs and 2 runs of length one.

Table 1: Subinterval probabilities for 128-bit sequences.

	Intervals	Probability
Box 1	0–27	0.219194
Box 2	28–31	0.230457
Box 3	32–34	0.184348
Box 4	35–38	0.259274
Box 5	39–128	0.106724

Since, we have exactly 4 runs, x_i's must be at least 1;

$$x_1 + x_2 + x_3 + x_4 = 8, \quad x_i \geq 1 \text{ for } i = 1, 2, 3, 4. \tag{17}$$

Fix $x_3 = x_4 = 1$ then;

$$x_1 + x_2 = 6 \quad x_i \geq 2 \text{ for } i = 1, 2. \tag{18}$$

We want $x_i \geq 2$. Define $x_i' = x_i + 2$ for $i = 1, 2$.

$$x_1 + x_2 = 6, \quad x_i \geq 2 \text{ for } i = 1, 2,$$
$$(x_1 + 1) + (x_2 + 1) = 2,$$
$$x_1' + x_2' = 2, \quad x_i' \geq 0 \text{ for } i = 1, 2,$$
$$x_1' = 2, \quad x_2' = 0$$
$$\Longleftrightarrow x_1 = 4, \quad x_2 = 2, \quad x_3 = 1, \quad x_4 = 1 \begin{cases} 11110010 \\ 00001101, \end{cases}$$
$$x_1' = 1, \quad x_2' = 1$$
$$\Longleftrightarrow x_1 = 3, \quad x_2 = 3, \quad x_3 = 1, \quad x_4 = 1 \begin{cases} 11100010 \\ 00011101, \end{cases}$$
$$x_1' = 0, \quad x_2' = 2$$
$$\Longleftrightarrow x_1 = 2, \quad x_2 = 4, \quad x_3 = 1, \quad x_4 = 1 \begin{cases} 11000010 \\ 00111101. \end{cases} \tag{19}$$

The above construction gives us 6 different sequences of length 8 with 2 runs of length one. Also selecting x_3 and x_4 gives us a factor of $\binom{4}{2}$. Hence, the total number of sequences of length 8 with 4 runs, 2 of which are of length one is $2 \cdot \binom{8-4-1}{4-2-1} \cdot \binom{4}{2} = 36$.

3.3. Number of Runs of Length Two. In this section, we calculate the number of sequences having l_2 runs of length two in a combinatorial approach. As in the previous section we use the same notation and the similar ideas in Section 3.1 to compute the number of sequences having total of r runs, l_2 of which are of length two and hence we calculate the probabilities. After that, using these calculations, we state the second new run test.

Table 2: Interval and probability values for runs of length one for 64-, 128-, 256-, and 512-bit blocks.

	$n = 64$		$n = 128$		$n = 256$		$n = 512$	
	Interval	Prob.	Interval	Prob.	Interval	Prob.	Interval	Prob.
Box 1	0–13	0.190082	0–27	0.173171	0–56	0.187255	0–117	0.193566
Box 2	14–16	0.238877	28–31	0.21426	57–61	0.189280	118–125	0.218630
Box 3	17-18	0.174560	32–34	0.186977	62–66	0.219859	126–132	0.217076
Box 4	19–21	0.211470	35–38	0.21339	67–72	0.218775	133–140	0.199515
Box 5	22–64	0.185009	39–128	0.21219	73–256	0.184827	141–512	0.171211

Theorem 7. *The probability of randomly chosen binary sequence $S = s_1, s_2, \ldots, s_n$ with length n, having r runs, l_1 of which are length one and l_2 of which are length and two is,*

$$\Pr\left(r_t = r, r_1 = l_1, r_2 = l_2\right) = \frac{\binom{n-2r+l_1-1}{r-l_1-l_2-1} \cdot \binom{n}{l_1} \cdot \binom{n-l_1}{l_2}}{2^{n-1}}. \tag{20}$$

Proof. As in the previous Theorems 2 and 3 we illustrate the sequence as follows;

$$x_1 + x_2 + \cdots + x_r = n. \tag{21}$$

Let us first assume that the last l_1 runs are of length one and l_2 runs are the runs of length two. The rest are of length at least three. That is,

$$\begin{gathered} x_{r-l_1+1} = \cdots = x_{r-1} = x_r = 1, \\ x_{r-l_1-l_2+1} = \cdots = x_{r-l_1-1} = x_{r-l_1} = 2, \\ x_1 + x_2 + \cdots + x_{r-(l_1+l_2)} + \overbrace{2 + 2 + \cdots + 2}^{l_2} + \overbrace{1 + 1 \cdots + 1}^{l_1} = n, \\ x_1 + x_2 + \cdots + x_{r-(l_1+l_2)} = n - l_1 - 2l_2. \end{gathered} \tag{22}$$

Notice that here, $x_i \geq 3$. We use the change of variables $y_i = x_i - 3$ for $i = 1, 2, \ldots, r - (l_1 + l_2)$

$$\begin{gathered} \left(x_1 - 3\right) + \left(x_2 - 3\right) + \cdots + \left(x_{r-(l_1+l_2)} - 3\right) \\ = n - \left(l_1 + 2l_2\right) - 3\left(r - l_1 - l_2\right), \\ y_1 + y_2 + \cdots + y_{r-(l_1+l_2)} = n - 3r + 2l_1 + l_2. \end{gathered} \tag{23}$$

The number of sequences having conditions, which are stated above, is equal to the number of nonnegative solutions of (23). Consequently, by the Fact 1, number of desired solutions is,

$$\binom{n - 2r + l_1 - 1}{r - l_1 - l_2 - 1}. \tag{24}$$

Selection of l_1 and l_2 runs of length 1 and length 2 give us a factor of $\binom{r}{l_1}\binom{r-l_1}{l_2}$. Since, each positive integer solution of (21) corresponds two sequences (one starts with 1, the other starts with 0) 2 is stated as factor also. Therefore, the number of all binary sequences of length n, having total number of runs, l_1 and l_2 of which length one and two respectively, is equal to,

$$2 \cdot \binom{n - 2r + l_1 - 1}{r - l_1 - l_2 - 1} \cdot \binom{r}{l_1} \cdot \binom{r - l_1}{l_2}. \tag{25}$$

Hence the probability of a randomly chosen sequence to have the above conditions is;

$$\begin{aligned} &\Pr\left(r_t = r, r_1 = l_1, r_2 = l_2\right) \\ &= \frac{2 \cdot \binom{n-2r+l_1-1}{r-l_1-l_2-1} \cdot \binom{r}{l_1} \cdot \binom{r-l_1}{l_2}}{2^n}. \end{aligned} \tag{26}$$

□

We find the number of sequences having r runs, l_1 and l_2 of which are length one and two respectively, using formula above. In order to define the second new run test, we need number of sequences of length n having l_2 runs of length two, without depending on the other variables such as, number of runs and number of runs of length one. Corollary 8 enables us to compute the probabilities that are needed for defining the new statistical test.

Corollary 8. *Let $N_2(l_2)$ denote the number of runs of sequences with exactly i runs of length two. Clearly, we have maximum $\lfloor n/2 \rfloor$ runs of length two. Otherwise sequence length exceeds n. Then, for $l_2 = 0, 1, 2, \ldots, \lfloor n/2 \rfloor$,*

$$N_2\left(l_2\right) = \sum_{l_1=0}^{n} \sum_{r=1}^{n} 2 \cdot \binom{n - 2r + l_1 - 1}{r - l_1 - l_2 - 1} \cdot \binom{r}{l_1} \cdot \binom{r - l_1}{l_2}. \tag{27}$$

Since the number of all sequences of length n is 2^n, probabilities follow immediately:

$$\Pr\left(r_2 = l_2\right) = \frac{N_2\left(l_2\right)}{2^n}. \tag{28}$$

Also Algorithm 2 enable the calculation for the number of sequences with desired conditions. Furthermore, subinterval probabilities can be stated in the same way as in Example 5. The subinterval probabilities can be seen in Table 3.

3.4. Number of Runs of Length Three. In the last section of this chapter, we focus on the number of sequences having exactly l_3 runs of length three. We use the same constructions with

$i \leftarrow 1, l_1 \leftarrow 0, r \leftarrow 1, N_2(l_2) \leftarrow 0.$
while $l_2 \leq \lfloor n/2 \rfloor$ **do**
  while $l_1 \leq n$ **do**
    while $r \leq n$ **do**
      $N_2(l_2) \leftarrow N_2(l_2) + \binom{n-2r+l_1-1}{r-l_1-l_2-1}\binom{r}{l_1}\binom{r-l_1}{l_2}/2^{n-1}$
      $r \leftarrow r+1$
    end while
    $l_1 \leftarrow l_1 + 1$
  end while
  $l_2 \leftarrow l_2 + 1$
end while
return N_2

Algorithm 2: Calculating $\Pr(r_2 = l_2)$ for $l_2 = 1, 2, \ldots, \lfloor n/2 \rfloor$.

Table 3: Interval and probability values for runs of length two test for 64-, 128-, 256-, and 512-bit blocks.

	$n = 64$		$n = 128$		$n = 256$		$n = 512$	
	Interval	Prob.	Interval	Prob.	Interval	Prob.	Interval	Prob.
Box 1	0–5	0.161344	0–12	0.167075	0–27	0.192579	0–57	0.188938
Box 2	6-7	0.260964	13-14	0.174075	28–30	0.194051	58–61	0.178794
Box 3	8	0.149093	15-16	0.209794	31–33	0.222923	62–65	0.210496
Box 4	9-10	0.245287	17–19	0.266590	34–36	0.187853	66–70	0.225615
Box 5	11–32	0.183309	20–64	0.182464	37–128	0.202591	71–256	0.196154

the previous sections to compute the number of sequences having total of r runs, l_3 of which are of length three, and hence we calculate the probabilities. Then using these calculations, we state the last new statistical test in the next chapter.

Theorem 9. *The probability of chosen binary sequence $S = s_1, s_2, \ldots, s_n$ with length n, having r runs, l_1 runs of length one, l_2 runs of length two, and l_3 runs of length three, is*

$$\Pr\left(r_t = r, r_1 = l_1, r_2 = l_2, r_3 = l_3\right) = \frac{\binom{n-3r+2l_1+l_2-1}{r-l_1-l_2-l_3-1}\cdot\binom{r}{l_1}\cdot\binom{r-l_1}{l_2}\cdot\binom{r-l_1-l_2}{l_3}}{2^{n-1}}. \tag{29}$$

Proof. As in Theorems 2, 3, and 7 we illustrate the sequence as follows:

$$x_1 + x_2 + \cdots + x_r = n. \tag{30}$$

Let us first assume that the last l_1 are of length 1, l_2 are of length 2, and l_3 are of length 3. The rest are of at least length four. Consider

$$x_{r-l_1+1} = \cdots = x_{r-1} = x_r = 1,$$
$$x_{r-l_1-l_2+1} = \cdots = x_{r-l_1-1} = x_{r-l_1} = 2,$$
$$x_{r-l_1-l_2-l_3+1} = \cdots = x_{r-l_1-l_2-1} = x_{r-l_1-l_2} = 3,$$
$$x_1 + x_2 + \cdots + x_{r-l_1-l_2-l_3} + \overbrace{3+3+\cdots+3}^{l_3} + \overbrace{2+2+\cdots+2}^{l_2} + \overbrace{1+1\cdots+1}^{l_1} = n,$$
$$x_1 + x_2 + \cdots + x_{r-l_1-l_2-l_3} = n - r - l_1 - 2l_2 - 3l_3. \tag{31}$$

Notice that $x_i \geq 4$ and we use the change of variables $y_i = x_i - 4$ for $i = 1, 2, \ldots, r - (l_1 + l_2 + l_3)$.

The number of cases is equal to the number of nonnegative solutions of the following equation:

$$\begin{aligned}\left(x_1 - 4\right) + \left(x_2 - 4\right) + \cdots + \left(x_{r-(l_1+l_2+l_3)} - 4\right) &= n - \left(l_1 + 2l_2 + 3l_3\right) - 4\left(r - l_1 - l_2 - l_3\right), \\ y_1 + y_2 + \cdots + y_{r-(l_1+l_2+l_3)} &= n - 4r + 3l_1 + 2l_3 + l_3.\end{aligned} \tag{32}$$

The number of sequences having conditions, which are stated above, is equal to the number of nonnegative solutions of (32). Consequently, by Fact 1, number of desired solutions is

$$\binom{n-3r+2l_1+l_2-1}{r-l_1-l_2-l_3-1}. \tag{33}$$

$l_3 \leftarrow 1, l_2 \leftarrow 1, l_1 \leftarrow 1, r \leftarrow 1, N_3(l_3) \leftarrow 1.$
while $l_3 \leq \lfloor n/3 \rfloor$ **do**
  while $l_2 \leq n$ **do**
    while $l_1 \leq n$ **do**
      while $r \leq n$ **do**

$$N_3(l_3) \leftarrow N_3(l_3) + 2\binom{n-3r+2l_1+l_2-1}{r-l_1-l_2-l_3-1} \cdot \binom{r}{l_1} \cdot \binom{r-l_1}{l_2} \cdot \binom{r-l_1-l_2}{l_3}$$

        $r \leftarrow r+1$
      end while
      $l_1 \leftarrow l_1 + 1$
    end while
    $l_2 \leftarrow l_2 + 1$
  end while
  $l_3 \leftarrow l_3 + 1$
end while
return N_3

Algorithm 3: Calculating $\Pr(r_3 = l_3)$ for $l_3 = 1, 2, \ldots, \lfloor n/3 \rfloor$.

Table 4: Interval and probability values for runs of length three test for 64-, 128-, 256-bit blocks.

	$n = 64$		$n = 128$		$n = 256$	
	Interval	Prob.	Interval	Prob.	Interval	Prob.
Box 1	0–2	0.207825	0–5	0.163209	0–13	0.248734
Box 2	3	0.204319	5–7	0.274500	14-15	0.207164
Box 3	4	0.216732	8	0.154854	16-17	0.213743
Box 4	5-6	0.283245	9-10	0.245059	18–20	0.222144
Box 5	7–21	0.087877	11–42	0.162376	20–85	0.108212

Selection of l_1, l_2, and l_3 runs gives us a factor of $\binom{r}{l_1}\binom{r-l_1}{l_2}\binom{r-l_1-l_2}{l_3}$. Therefore, the number of all binary sequences of length n with conditions stated above is

$$2 \cdot \binom{n-3r+2l_1+l_2-1}{r-l_1-l_2-l_3-1} \cdot \binom{r}{l_1} \cdot \binom{r-l_1}{l_2} \cdot \binom{r-l_1-l_2}{l_3}. \tag{34}$$

Hence, the probability of a randomly chosen sequence to have these conditions is

$$\Pr(R = r, R_1 = l_1, R_2 = l_2, R_3 = l_3) = \frac{2 \cdot \binom{n-3r+2l_1+l_2-1}{r-l_1-l_2-l_3-1} \cdot \binom{r}{l_1} \cdot \binom{r-l_1}{l_2} \cdot \binom{r-l_1-l_2}{l_3}}{2^n}. \tag{35}$$

□

We find the number of sequences having r runs, l_1, l_2, and l_3 of which are of lengths one, two, and three, using the formula above. In order to use probabilities in tests we need numbers of sequences with length n and l_3 runs of length two, without depending on the other variables. Corollary 10 enables us to compute the probabilities that are needed for defining the new statistical test.

Corollary 10. *Let $N_3(l_3)$ denote the number of runs of sequences with exactly l_3 runs of length three. Clearly, we have maximum $\lfloor n/3 \rfloor$ runs of length three. If $l_3 > \lfloor n/3 \rfloor$ sequence length exceeds n, then, for $l_3 = 0, 1, 2, \ldots, \lfloor n/3 \rfloor$,*

$$N_3(l_3) = \sum_{l_2=0}^{n} \sum_{l_1=0}^{n} \sum_{r=1}^{n} 2\binom{n-3r+2l_1+l_2-1}{r-l_1-l_2-l_3-1} \cdot \binom{r}{l_1} \cdot \binom{r-l_1}{l_2} \cdot \binom{r-l_1-l_2}{l_3}. \tag{36}$$

Since the number of all sequences of length n is 2^n, probabilities follow immediately:

$$\Pr(r_3 = l_3) = \frac{N_3(l_3)}{2^n}. \tag{37}$$

Since the number of all sequences of length n is 2^n, probabilities follow immediately: $\Pr(r_3 = l_3) = N_3(l_3)/2^n$. And Algorithm 3 enables the calculations of the number of sequences of length n and l_3 runs of length three and hence subinterval probabilities can be stated in the same way as in Example 5. The subinterval probabilities can be seen in Table 4.

In this chapter we formulate the exact numbers of sequences with given conditions and hence corresponding probabilities are given. As we mentioned before calculating the probabilities for number of runs of length more than three is unpractical. The probabilities can be stated theoretically in the same way. However the time consumption of algorithms

to find the exact values grows exponentially. Therefore, it is inconvenient to use them in test suites.

4. Tests Descriptions

Golomb's first postulate is about the weight of a sequence and in many test suites the postulate is implemented with a proper generalization. On the other hand, the second postulate, which is about runs of a sequence, is mostly implemented according to the total number of runs regardless of their lengths. In this chapter, we define three new statistical tests as a proper generalization of Golomb's second postulate which are runs of length one test, runs of length two test, and runs of length three test. The subjects of new run tests are r_1, r_2, and r_3 as their names state.

We test the null hypothesis (H_0) which states that the sequence is randomly produced. There are two type of errors which are called *type I* and *type II* errors. Type I error occurs when the data is random and H_0 is rejected and the second one occurs when the data is nonrandom and H_0 is accepted. Probability of type I error is called *level of significance* and denoted by α. A statistical test evaluates the sequence against this predefined number α. If p value, produced by statistical test, is greater than α, then H_0 is accepted. Level of significance is decided based on the applications. We set α as 0.01, as in many test suites.

We use χ^2 as reference distribution. The measurements are compared with the expected values. In order to make a comparison we divide number of runs of lengths one, two, and three into subintervals, as explained in Section 3. New tests use the subintervals with the following property: $\Pr_i(\alpha_i < R < \alpha_{i+1}) \approx 0.2$. For example, probabilities of 128-bit sequences for runs of length two test can be divided into 5 subintervals as follows:

$$\begin{aligned}
&\Pr_1\left(1 \le r_2 \le 12\right) = 0.167075,\\
&\Pr_2\left(13 \le r_2 \le 14\right) = 0.174075,\\
&\Pr_3\left(15 \le r_2 \le 16\right) = 0.209794,\\
&\Pr_4\left(17 \le r_2 \le 19\right) = 0.266590,\\
&\Pr_5\left(20 \le r_2 \le 32\right) = 0.182464.
\end{aligned} \tag{38}$$

After calculating the subinterval probabilities, we count the number of runs of length i in the m different sequences and increment the corresponding subinterval counter by one according to the counted number of runs. To denote the number of sequences in the given subinterval we use F_i. Before the last step we calculate the χ^2 using the following formula [16]. Also N denotes the number of sequences. Consider

$$\chi^2 = \sum_{i=1}^{5} \frac{\left(F_i - N \cdot \Pr_i\right)^2}{N \cdot \Pr_i}. \tag{39}$$

Lastly p value is calculated according to the given values:

$$p \text{ value} = \texttt{igamc}\left(\frac{5-1}{2}, \frac{\chi^2}{2}\right). \tag{40}$$

We test the H_0 by comparing the produced p value with the level of significance α and accept or reject the H_0. That is, if p value $> \alpha$, H_0 is accepted; otherwise it is rejected.

New tests can be implemented on sequences of length $n = m \cdot 25$ (where m is the block size). This number is a direct consequence of creating subintervals. In order to get reliable results, in each subinterval we need at least 5 blocks of sequences. In NIST test suite it is suggested that the sequences should be about 20.000 bits long. Therefore, new run tests can be implemented on short sequences also.

Remark 11 (derivative of a sequence). Let $S = s_0, s_1, \ldots, s_{n-1}$ be a binary sequence of length n; then, derivative of S, denoted by $\Delta S = \Delta s_0, \Delta s_1, \ldots, \Delta s_{n-1}$, is defined as follows.

For $i = 0, 1, \ldots, n-1$,

$$\Delta s_i = \begin{cases} s_i \oplus s_{i+1} & \text{if } i = 0, 1, \ldots, n-2\\ 1 & \text{if } i = n-1. \end{cases} \tag{41}$$

Counting runs of a sequence by using the definition is unpractical. So we use the derivative of a sequence to count the runs. By the definition, all 1's in the derivative of a sequence indicate the end of a run. So the number of runs of a sequence can be defined as the weight of its derivative.

Also we use a variation of derivative $\Delta S'$ of length $n+1$ by adding 1's at the beginning the sequence ΔS. The variation of derivative is an important part of new defined run tests, since the number of runs of different length is determined by this sequence.

Remark 12. Let $S = s_0, s_1, \ldots, s_{n-1}$ be a binary sequence and derivative of S is denoted by $\Delta S = \Delta s_0, \Delta s_1, \ldots, \Delta s_{n-1}$. Then $\Delta S' = \Delta s_0', \Delta s_1', \ldots, \Delta s_n'$ is defined as follows:

$$\Delta s_i' = \begin{cases} \Delta s_{i-1} & \text{if } i = 1, \ldots, n\\ 1 & \text{if } i = 0. \end{cases} \tag{42}$$

In order to count the runs at the beginning, we use a variation of derivative instead of the original derivative definition. Number of runs of length one in a sequence is indicated by the number of overlapping occurrences of *11* in its variation of derivative. In the same way number of runs of lengths 2 and 3 in a sequence is indicated by the number of overlapping occurrences of *101* and *1001*, respectively. More generally we can say that number of runs of length n is indicated by the overlapping number of occurrences of $1\underbrace{00\cdots0}_{n-1}1$.

Example 13. Let $S = 01100010011111001100011101010000$ be a binary sequence of length 32, having 15 runs, 6 runs of

length one, 4 runs of length two, and 3 runs of length three. Then

$$\Delta s_0 = s_0 \oplus s_1, \Delta s_1 = s_0 \oplus s_2, \ldots, \Delta s_{31} = s_{31} \oplus s_{32}, \Delta s_{32} = 1,$$
$$\Delta S = 10100110100001010100100111110000 1,$$
$$\Delta S' = 11010011010000101010010011111000 01. \tag{43}$$

(i) Weight of ΔS is 15 which corresponds to number of runs.

(ii) Number of overlapping occurrences of 11 is 6 which corresponds to number of runs of length one: $\Delta S' = \underbrace{11}_{1}0100\underbrace{11}_{1}0100001010100100\underbrace{1111}_{4}00001$.

(iii) Number of overlapping occurrences of 101 is 4 which corresponds to number of runs of length two:

$$\Delta S' = 1\underbrace{101}_{1}001\underbrace{101}_{1}0000\underbrace{10101}_{2}001001111100001. \tag{44}$$

(iv) Number of overlapping occurrences of 1001 is 3 which corresponds to number of runs length three:

$$\Delta S' = 110\underbrace{1001}_{1}10100001010\underbrace{1001001}_{2}111100001. \tag{45}$$

Before defining new statistical tests, we give the general idea of the test by following example.

Example 14. Let S be a binary sequence of length 2^{21}. Let F_i and $\Pr_i$ be the number of sequences in given subinterval and probability of it, respectively.

Step 1. Choose a block size m. In our example we choose m as 128.

Step 2. Then divide the sequence into m-bit sequence. Then we get the set of sequences as follows: $\mathbf{S} = \{S_1, S_2, \ldots, S_{2^{14}}\}$.

Step 3. For each S_i count the number of runs of lengths one, two, and three. And increment the corresponding boxes by 1. Consider

$$\begin{aligned} S_1 &= [0,1,0,0,\ldots,1] \longrightarrow l_1^1 = 33,\ l_2^1 = 15,\ l_3^1 = 8,\\ S_2 &= [0,1,1,0,\ldots,0] \longrightarrow l_1^2 = 32,\ l_2^2 = 17,\ l_3^2 = 9,\\ &\vdots\\ S_{2^{14}} &= [0,1,0,0,\ldots,1] \longrightarrow l_1^{2^{14}} = 30,\ l_2^{2^{14}} = 16,\ l_3^{2^{14}} = 8. \end{aligned} \tag{46}$$

Table 5: Number of sequences in given intervals for runs of length one test, runs of length two test, and runs of length three test.

(a) Runs of length one test

	Interval	Count
F_1	0–27	3.699
F_2	28–31	3.744
F_3	32–34	3.016
F_4	35–38	3.155
F_5	39–128	2.770

(b) Runs of length two test

	Interval	Count
F_1	0–12	2.806
F_2	13-14	2.838
F_3	15-16	3.476
F_4	17–19	4.331
F_5	20–64	2.933

(c) Runs of length three test

	Interval	Count
F_1	0–5	2.634
F_2	5–7	4.447
F_3	8	2.532
F_4	9-10	4.082
F_5	11–42	2.689

Step 4. Then, we get Table 5. Count rows of each test corresponding to the number of sequences whose number of runs of length one, two, or three is in given interval.

Step 5. χ^2 is calculated by the given formula and p value is computed accordingly:

$$\chi^2 = \sum_{i=1}^{5} \frac{\left(F_i - 2^{14} \cdot \Pr_i\right)^2}{2^{14} \cdot \Pr_i},$$
$$p \text{ value} = \texttt{igamc}\left(\frac{5-1}{2}, \frac{\chi^2}{2}\right). \tag{47}$$

Step 6. Finally, we get the p value for each test.

(i) Number of runs of length one test: p value = 0.357056.

(ii) Number of runs of length two test: p value = 0.462207.

(iii) Number of runs of length three test: p value = 0.627001.

4.1. Runs of Length One Test. The subject of the first new run test is runs of length one in the sequences. Test uses the probabilities calculated in the previous chapter. First, we collect the algorithms output and generate the data set **S**. If the given sequence of length n is a long binary sequence, the sequence is divided into m-bit blocks and gets a set of

$\Delta S'_k = \Delta s'_{k,0}, \Delta s'_{k,1}, \ldots, \Delta s'_{k,m}$
$i \leftarrow 0, l_1^k \leftarrow 0$
while $i \leq m - 1$ **do**
  temp $= \Delta s'_{k,i} \cdot 2^1 + \Delta s'_{k,i+1} \cdot 2^0$
  if temp $= 3$ **then**
    $l_1^k \leftarrow l_1^k + 1$
  end if
  $i \leftarrow i + 1$
end while
Apply χ^2 of Goodness of Fit test to the values in L_1.
return p-value.

Algorithm 4: Runs of length one test $(S_1, S_2, \ldots, S_N)$, $L_1 = \{l_1^1, l_1^2, \ldots, l_1^N\}$.

$\Delta S'_k = \Delta s'_{k,0}, \Delta s'_{k,1}, \ldots, \Delta s'_{k,m}$
$i \leftarrow 0, l_2^k \leftarrow 0$
while $i \leq m - 2$ **do**
  temp $= \Delta s'_{k,i} \cdot 2^2 + \Delta s'_{k,i+1} \cdot 2^1 + \Delta s'_{k,i+2} \cdot 2^0$
  if temp $= 5$ **then**
    $l_2^k \leftarrow l_2^k + 1$
  end if
  $i \leftarrow i + 1$
end while
Apply χ^2 of Goodness of Fit test to the values in L_2.
return p-value.

Algorithm 5: Runs of length two test $(S_1, S_2, \ldots, S_N)$, $L_2 = \{l_2^1, l_2^2, \ldots, l_2^N\}$.

sequences and generates $\mathbf{S} = \{S_1, S_2, \ldots, S_N\}$ where $N = \lfloor n/m \rfloor$. In our test m can be 64, 128, 256, or 512. After generating the data set, the set L_1 is formed by counting the number of runs of length one in each sequence. In order to find the number of runs of length one, first we find the derivative of the binary sequence ΔS_k and then we count the overlapping occurrences 11 in $\Delta S'_k$ for $k = 1, 2, \ldots, N$. After that we apply χ^2 of goodness of fit test to the values in L_1. We propose new run test to implement the idea of Golomb's second postulate in statistical randomness test. The pseudocode of the test is given in Algorithm 4.

4.2. Runs of Length Two Test. After giving the first new run test, we define runs of length two test. Test uses the probabilities calculated in the previous chapter. As in the runs of length one test first, we generate the data set **S**. Also in the second test the block size m can be 64, 128, 256, or 512. From the data set **S**, the set L_2 is formed by counting the number of runs of length two in each sequence. Like in the previous test we get the derivative of the binary sequence ΔS_k. In order to find the number of runs of length two, we count the overlapping occurrences 101 in $\Delta S'_k$. Then we apply χ^2 of goodness of fit test to the values in L_2. The second new run test constitutes another approach to Golomb's second postulate. The pseudocode of the test is given as in Algorithm 5.

$\Delta S'_k = \Delta s'_{k,0}, \Delta s'_{k,1}, \ldots, \Delta s'_{k,m}$
$i \leftarrow 0, l_3^k \leftarrow 0$
while $i \leq m - 3$ **do**
  temp $= \Delta s'_{k,i} \cdot 2^3 + \Delta s'_{k,i+1} \cdot 2^2 + \Delta s'_{k,i+2} \cdot 2^1 + \Delta s'_{k,i+3} \cdot 2^0$
  if temp $= 9$ **then**
    $l_3^k \leftarrow l_3^k + 1$
  end if
  $i \leftarrow i + 1$
end while
Apply χ^2 of Goodness of Fit test to the values in L_3.
return p-value.

Algorithm 6: Runs of length three test $(S_1, S_2, \ldots, S_N)$, $L_3 = \{l_3^1, l_3^2, \ldots, l_3^N\}$.

4.3. Runs of Length Three Test. The last new run test is runs of length three test. This test also uses the probabilities calculated in the previous chapter. Data sets are created as in the previous run tests. Also in the last new run test block size m can be 64, 128, or 256. The set L_3 is formed by using **S**. The counting phase of this test is done by finding the total number of the overlapping occurrences 1001 in $\Delta S'_k$. Then we apply χ^2 of goodness of fit test to the values in L_3. The pseudocode of the last new run test is given in Algorithm 6.

Together with three new run tests we implement the idea of Golomb's second postulate in statistical randomness tests. The new run tests, concerning runs of lengths one, two, and three, constitute a better proper generalization of Golomb's idea.

5. Implementations

In order to check the reliability of tests stated in the previous section, we implement new test together with well-known statistical tests included in NIST test suite.

In the first part of the experiments we select 5 encryption algorithms, which are Advanced Encryption Algorithms finalists, MARS [18], RC6 [19], Rijndael [20], Serpent [21], and Twofish [22]. 2^{16} pseudorandom sequences of length 128 are generated with encryption of noncorrelated data by using these algorithms. In other words, in the first experiment we test the outputs of AES finalists using our tests and NIST test suite. New run tests are implemented on 2^{14} pseudorandom sequences of length 128 as described in the previous section and NIST's tests are implemented on a binary sequence of length 2^{21} by concatenating the outputs of algorithms. The results can be seen in Table 6.

In the second part of the experiments, we use the binary expansions of e, π, and $\sqrt{2}$. The binary expansions can be found within the NIST test suite. As in the first part we also use well-known tests that are included in NIST test suite. We collect first 2^{19} bits of the binary expansions. In order to apply new run tests, collected long sequence is divided into 128-bit blocks; hence we get 2^{12} sequences of length 128. Using the second implementation we show the performance of new run tests. The test results can be seen in Table 7.

Table 6: Test results for the 128-bit outputs of AES finalists.

Statistical tests	Rijndael	Serpent	Mars	RC6	Twofish
Frequency test	0.877073	0.385771	0.100285	0.813306	0.667550
Block frequency test	0.722551	0.159257	0.801489	0.475342	0.199609
Run test	0.703085	0.000651	0.003002	0.006542	0.006737
Longest run of ones in a block	0.031990	0.661453	0.229015	0.338937	0.308989
Universal statistical test	0.006504	0.048462	0.007328	0.108877	0.023687
Linear complexity test	0.308490	0.231002	0.159494	0.662083	0.452449
Serial test[1]	0.016532	0.249989	0.748831	0.307892	0.629330
Serial test[2]	0.444775	0.504040	0.226215	0.602572	0.923866
Approximate entropy test	0.001276	0.070437	0.322856	0.053931	0.220444
Cumulative sums test—backward	0.271617	0.627426	0.152360	0.822441	0.838133
Cumulative sums test—forward	0.362406	0.501622	0.057094	0.971814	0.877082
Random excursion test	0.949243	0.143578	0.455967	0.307333	0.409744
Random excursions variant test	0.816055	0.042998	0.515433	0.160018	0.041629
Runs of length one test	0.535513	0.076538	0.021622	0.055930	0.008255
Runs of length two test	0.095602	0.339466	0.051861	0.057043	0.309454
Runs of length three test	0.359483	0.213636	0.388663	0.318248	0.081348

[1,2]Two different versions of serial test in NIST test suite.

Table 7: Test results for the binary expansion of e, π, and $\sqrt{2}$.

Statistical test	e	π	$\sqrt{2}$
Frequency test	0.818668	0.393382	0.820816
Block frequency test	0.069195	0.191721	0.578760
Run test	0.489904	0.409869	0.894467
Longest run of ones in a block	0.328344	0.048248	0.537307
Universal statistical test	0.930374	0.915310	0.462562
Linear complexity test	0.927809	0.208269	0.396546
Serial test[1]	0.924970	0.232328	0.247445
Serial test[2]	0.719054	0.221747	0.037551
Approximate entropy test	0.707174	0.085060	0.837672
Cumulative sums test—backward	0.373319	0.333600	0.629320
Cumulative sums test—forward	0.242488	0.313745	0.838133
Random excursion test	0.892831	0.844143	0.270246
Random excursions variant test	0.388323	0.760966	0.461287
Runs of length one test	0.241279	0.097072	0.138194
Runs of length two test	0.092391	0.129520	0.158537
Runs of length three test	0.215721	0.114384	0.076582

[1,2]Two different versions of serial test in NIST test suite.

In the last part of the experiments, we analyse the sensitivity of new run tests. In order to do the implementation, first we need to generate a nonrandom sequence.

A nonrandom sequence can be generated in two steps. First, we create a sequence of random numbers $R = \{r_0, r_1, \ldots, r_{n-1}\}$ such that $0 \le r_i \le 1$ for $i = 0, 1, \ldots, n-1$ using RNGCryptoServiceProvider classes of C#. After the generation we create nonrandom data by using the following important concept in cryptography defined in [23].

Let $R = r_0, r_1, \ldots, r_{n-1}$ be the outputs of a random number generator and $0 \le r_i \le 1$ for $i = 0, 1, \ldots, n-1$
$i \leftarrow 0$
while $i < n$ **do**
  if $r_i \le 0.5 + q$ **then**
    $s_i \leftarrow 0$
  else
    $s_i \leftarrow 1$
  end if
  $i \leftarrow i + 1$
end while
return S^q

Algorithm 7: Generation of biased sequence $S^q = s_0^q, s_1^q, \ldots, s_{n-1}^q$.

Definition 15. Let S be a binary sequence of length n and ith element of it is represented as s_i; then bias q is defined as follows:

$$\Pr(s_i = 1) = \frac{1}{2} + q \qquad \Pr(s_i = 0) = \frac{1}{2} - q. \tag{48}$$

Clearly, we can say that in a true random sequence we expect bias as 0. That is, $\Pr(s_i = 1) = \Pr(s_i = 0) = 1/2$. Moreover, this is the main idea of Golomb's first postulate. To generate nonrandom sequence we need to increase the bias. Finally using Algorithm 7 we can generate a nonrandom sequence.

Example 16. Let $R = r_0, r_1, \ldots, r_{n-1}$ be a random sequence with $0 \le r_i \le 1$ for $i = 0, 1, \ldots, n-1$; from this sequence we construct a binary sequence with bias 0.05. The generation of nonrandom sequence can be summarized as follows:

$$s_j^q = \begin{cases} 0 & \text{if } r_i \le 0.5 + 0.05 \\ 1 & \text{if } r_i > 0.5 + 0.05, \end{cases}$$

Table 8: Test results for nonrandom data sets.

Statistical test	$q = 0.0$	$q = 0.01$	$q = 0.03$
Frequency test	0.375269	0.040143	0.000475
Block frequency test	0.760739	0.802281	0.777309
Run test	0.794303	0.903035	0.859454
Longest run of ones in a block	0.562918	0.257811	0.093295
Nonoverlapping template test ($M = 9$, $B = 000000001$)	0.436359	0.377016	0.328182
Overlapping template test ($M = 9$)	0.746164	0.642254	0.714769
Linear complexity test	0.693577	0.703492	0.670893
Serial test[1]	0.680524	0.681398	0.549883
Serial test[2]	0.538842	0.746869	0.615192
Approximate entropy test	0.372373	0.239482	0.308904
Cumulative sums test—backward	0.372373	0.032272	0.000333
Cumulative sums test—forward	0.429406	0.073315	0.000857
Runs of length one test	0.818485	0.832025	0.809327
Runs of length two test	0.944299	0.790180	0.852354
Runs of length three test	0.574298	0.782597	0.891987

[1,2]Two different versions of serial test in NIST test suite.

$$\Pr\left(s_j^q = 1\right) = 0.55, \qquad \Pr\left(s_j^q = 0\right) = 0.45. \tag{49}$$

In the last part of the experiments we generate nonrandom datum with different biases using the above construction. We observe the behaviour of new run tests with respect to the randomness of a sequence. The last results show the efficiency of the new tests. Moreover new run tests can detect the deviations in distributions of runs while other tests cannot. The test results can be seen in Table 8.

6. Conclusion

In cryptography almost all applications use random looking sequences. Therefore randomness is one of the most important issues for cryptographic algorithms. In fact, using weak random values enables an adversary to break the whole system.

In all applications, used values should be of sufficient size and be random, in such a manner that probability of any chosen quantity should be small enough to eliminate an adversary to gain any specific information. Therefore, sequences and numbers, used as a key in cryptographic algorithms, should be pseudorandom. Also these sequences should have good statistical properties. For these reasons statistical randomness is an important topic. While giving a mathematical proof that a generator is a random bit generator is nearly impossible, statistical tests are defined to detect weaknesses that a generator could have. Hence, they are considered as an important part of evaluating security of cryptographic algorithms.

In this work, we propose three new statistical tests based on Golomb's second postulate. Finding the real probabilities related to number of runs of lengths one, two, and three enables us to compare the observed values accordingly. New run tests can be used in test suites to test security of algorithms so that Golomb's second postulate is implemented in a proper way. Moreover, these tests can be used as an evaluation tool for short sequences such as outputs of block ciphers and hash functions. These tests can detect deviations in distribution runs which cannot be detected by other tests.

Also, we experiment with some standard encryption algorithms that behave like pseudorandom number generator and random sequences such as binary expansion of e, π, and $\sqrt{2}$. Implementations show the consistency of new statistical test with other well-known statistical tests. It is shown that, in order to detect the deviation from randomness (in the sense of distribution of runs), new statistical tests are more efficient than other statistical tests.

As a future work, we extend statistical tests to approach Golomb's randomness postulates more than now. And correlations between new statistical tests and also with other statistical tests can be examined.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, Fla, USA, 1st edition, 1996.

[2] D. E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, Addison-Wesley, Longman Publishing, Boston, Mass, USA, 3rd edition, 1997.

[3] A. L. Rukhin, "Testing randomness: a suite of statistical procedures," *Theory of Probability & Its Applications*, vol. 45, no. 1, pp. 111–132, 2001.

[4] G. Marsaglia, "The marsaglia random number CDROM including the diehard battery of tests of randomness," 1995, http://www.stat.fsu.edu/pub/diehard/.

[5] W. Caelli, "Crypt x package documentation," Tech. Rep., Information Security Research, 1992.

[6] P. L'ecuyer and R. Simard, "TestU01: a C library for empirical testing of random number generators," *ACM Transactions on Mathematical Software*, vol. 33, no. 4, article 22, 2007.

[7] L. E. Bassham III, A. L. Rukhin, J. Soto et al., "A statistical test suite for random and pseudorandom number generators for cryptographic applications," Tech. Rep. Sp 800-22 rev.1a, NIST, Gaithersburg, Md, USA, 2010.

[8] U. M. Maurer, "A universal statistical test for random bit generators," *Journal of Cryptology*, vol. 5, no. 2, pp. 89–105, 1992.

[9] V. Katos, "A randomness test for block ciphers," *Applied Mathematics and Computation*, vol. 162, no. 1, pp. 29–35, 2005.

[10] P. M. Alcover, A. Guillamón, and M. D. C. Ruiz, "A new randomness test for bit sequences," *Informatica*, vol. 24, no. 3, pp. 339–356, 2013.

[11] S. W. Golomb, *Shift Register Sequences*, Aegean Park Press, Laguna Hills, Calif, USA, 1982.

[12] S.-J. Kim, K. Umeno, and A. Hasegawa, "Corrections of the nist statistical test suite for randomness," *International Association for Cryptologic Research*, p. 18, 2004.

[13] K. Hamano and H. Yamamoto, "A randomness test based on T-codes," in *Proceedings of the International Symposium on Information Theory and its Applications (ISITA '08)*, pp. 1–6, December 2008.

[14] K. Hamano and T. Kaneko, "Correction of overlapping template matching test included in NIST randomness test suite," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 90, no. 9, pp. 1788–1792, 2007.

[15] J. Soto and L. Bassham, "Randomness testing of the advanced encryption standard finalist candidates," NIST IR 6483, National Institute of Standards and Technology, 1999.

[16] F. Sulak, A. Doğanaksoy, B. Ege, and O. Koçak, "Evaluation of randomness test results for short sequences," in *Sequences and Their Applications—SETA 2010*, vol. 6338 of *Lecture Notes in Computer Science*, pp. 309–319, Springer, Berlin, Germany, 2010.

[17] S. Ross, *A First Course in Probability*, Prentice Hall, New York, NY, USA, 6th edition, 2002.

[18] C. Burwick, D. Coppersmith, E. D'Avignon et al., Mars—a candidate cipher for AES, NIST AES Proposal, 1999.

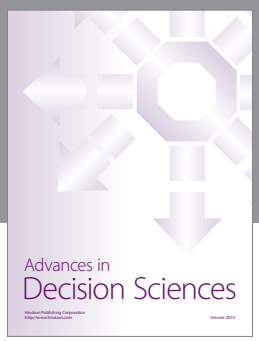
[19] R. L. Rivest, M. J. B. Robshaw, Y. Yin, and R. Sidney, *The RC6 Block Cipher*, 1998.

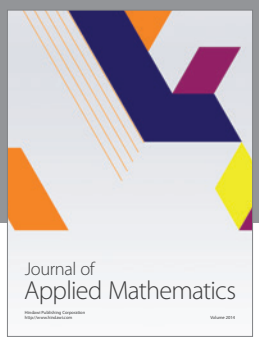
[20] J. Daemen and V. Rijmen, *The Design of Rijndael*, Springer, New York, NY, USA, 2002.

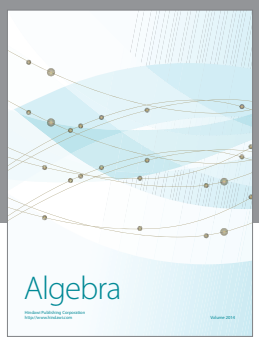
[21] E. Biham, R. J. Anderson, and L. R. Knudsen, "Serpent: a new block cipher proposal," in *Fast Software Encryption: 5th International Workshop, FSE' 98 Paris, France, March 23–25, 1998 Proceedings*, vol. 1372 of *Lecture Notes in Computer Science*, pp. 222–238, Springer, Berlin, Germany, 1998.

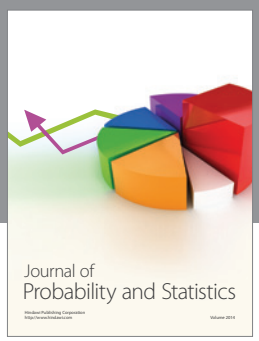
[22] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson, "Twofish: a 128-bit block cipher," in *Proceedings of the 1st Advanced Encryption Standard (AES) Conference*, Ventura, Calif, USA, August 1998.

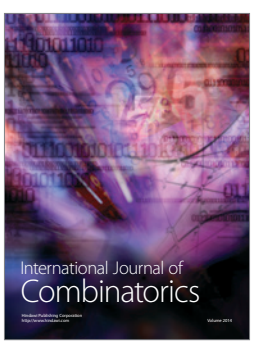
[23] H. M. Heys, "A tutorial on linear and differential cryptanalysis," *Cryptologia*, vol. 26, no. 3, pp. 189–221, 2002.

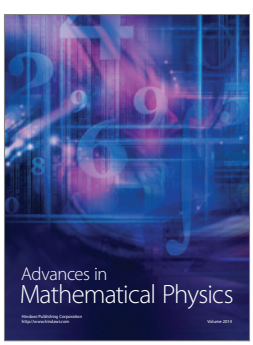


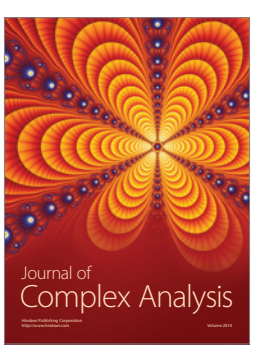


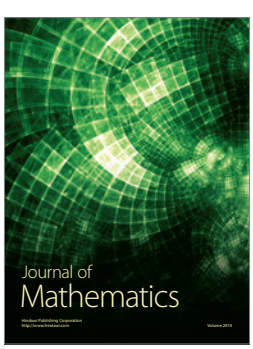


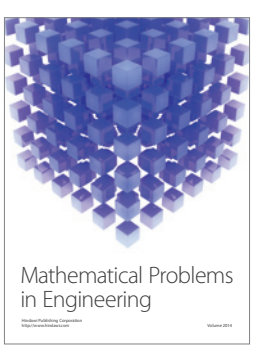


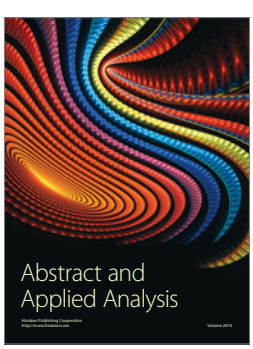


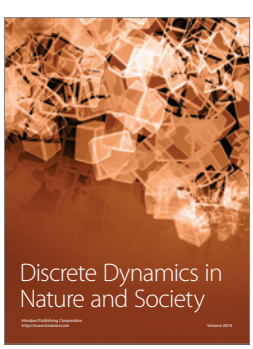


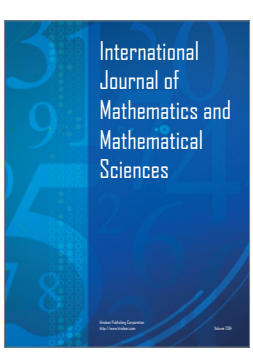












Hindawi

Submit your manuscripts at
http://www.hindawi.com