Hacettepe Journal of
Mathematics & Statistics

Hacet. J. Math. Stat.
Volume 53 (1) (2024), 230 – 250
DOI : 10.15672/hujms.1190807

Research Article

LS-14 test suite for long sequences

Ziya Akcengiz[1], Melis Aslan[2], Ali Doğanaksoy[3], Fatih Sulak[4], Muhiddin Uğuz*[5]

[1] *UEKAE, TÜBİTAK, Kocaeli, Turkey*
[2,3,5] *Department of Mathematics, Middle East Technical University, Ankara, Turkey*
[4] *Department of Mathematics, Atılım University, Ankara, Turkey*

Abstract

Random number sequences are used in many branches of science. Because of many technical reasons and their practicality, pseudo random sequences are usually employed in place of true number sequences. Whether a sequence generated through a deterministic process is a pseudo random, in other words, random-looking sequence or it contains certain patterns, can be determined with the help of statistics and mathematics. Although, in the literature there are many statistical randomness tests for this purpose, there is no much work on test suites specialized for long sequences, that is sequences of length 1,000,000 bits or more. Most of the randomness tests for long sequences use some mathematical approximations to compute expected values of the random variables and hence their results contain some errors. Another approach to evaluate randomness criteria of long sequences is to partition the long sequence into a collection short sequences and evaluate the collection for the ran- domness using statistical goodness of fit tests. The main advantage of this approach is, as the individual sequences are short, there is no need to use mathematical approximations. On the other hand when the second approach is preferred, partition the long sequence into a collection of fixed length subsequences and this approach causes a loss of information in some cases. Hence the idea of dynamic partition should be included to perform a more reliable test suite. In this paper, we propose three new tests, namely the entire R2 run, dynamic saturation point, and dynamic run tests. Moreover, we introduce a new test suite, called LS-14, consisting of 14 tests to evaluate randomness of long sequences. As LS-14 employs all three approaches: testing the entire long sequence, testing the collection of fixed length partitions of it, and finally, testing the collection obtained by the dynamic partitions of it, the proposed LS-14 test suit differs from all existing suites. Mutual comparisons of all 14 tests in the LS-14 suite, with each other are computed. Moreover, results obtained from the proposed test suite and NIST SP800-22 suite are compared. Examples of sequences with certain patterns which are not observed by NIST SP800-22 suite but detected by the proposed test suite are given.

Keywords. Randomness, random number, statistical tests, cryptography, NIST SP800-22, dynamic partitioning

Mathematics Subject Classification (2020). 11B50, 03D80, 5A19, 97K10, 05A15, 68P25

*Corresponding Author.
Email addresses: ziya.akcengiz@tubitak.gov.tr(Z. Akcengiz), melisa@metu.edu.tr(M. Aslan), aldoks@metu.edu.tr(A. Doğanaksoy), fatih.sulak@atilim.edu.tr(F.Sulak), muhid@metu.edu.tr (M. Uğuz)
Received: 03.11.2022; Accepted: 03.08.2023

1. Introduction

A sequence without any pattern or any order is considered as a random sequence. True Random Number Generators (TRNG) produce such sequences by extracting randomness from complicated physical phenomena. However, TRNGs are usually not useful for cryptographic purposes as it is impossible to reproduce a generated sequence without providing the same physical environment. Moreover, transmission and storage of the true random data have a high cost. Another type of random number generator is Pseudo-Random Number Generators (PRNGs). In the literature, there are examples of PRNGs [1, 41, 44] The structure of PRNGs depends on some deterministic algorithms and hence the sequences generated by PRNGs are generally periodic and depend on the seed. Thus, outputs of PRNGs are not truly random sequences. However, since the regeneration of sequences is possible whenever providing the same seeds, they are used for cryptographic purposes. It is important to test the outputs of PRNGs for being indistinguishable from truly random number sequences. Randomness tests are also used in evaluation of outputs of cryptographic algorithms which are supposed to look like random mappings.

Therefore, PRNGs and cryptographic algorithms should be designed well so that outputs are indistinguishable from an output of a TRNG. Although it is not possible to decide for sure that whether a sequence under consideration is an output of a truly random generator or not, using statistical properties, one can claim that the sequence looks random with a certain probability with the help of statistical tests depending on some combinatorial aspects. Golomb's Postulates are one of the good examples for this purpose [11]. Although these postulates are essential for the philosophy of randomness, they are too strict. A non-periodic finite partition of an infinite true random sequence may not provide these postulates. In the literature, rather than Golomb's postulates, statistical approaches are studied to decide whether the finite sequence under consideration is random with a certain probability. Statistical randomness tests use different random variables, mostly inspired by Golomb's postulates [7, 10, 13, 14, 16, 26, 30, 34, 35, 38, 42].

While testing a sequence, either single or more generally, a group of statistical tests, called as a test suites, can be used. One of the oldest test suites, suggested by [18], involves four basic tests: *frequency, serial, poker and gap tests.* The main idea of most of the statistical test suites was suggested by [20] in his book. He proposed ten statistical tests: *birthday spacings, collision, coupon collector's, frequency, gap, maximum of t, permutation, poker, run and serial test.*

P. L'Ecuyer introduced some statistical tests in [21]. After, P. L'Ecuyer and Simard defined a test suite *TESTU01* which is the suite of test batteries [22]. Marsaglia et al. defined some new basic tests in [25], and then introduced a test suite called *DIEHARD* in [24]. In this test suite, there are twelve statistical randomness tests. Later, Brown introduced DIEHARDER test suite, including all tests in DIEHARD and ten more tests [3]. Rukhin defined a test suite including eleven statistical tests in [28]. National Institute of Standards and Technology (NIST) proposed a test suite using these tests and some others [29]. NIST test suite involves fifteen tests. For testing the randomness of outputs of Advanced Encryption Standard (AES) candidates, Soto [31] use this suite. In years, after some corrections and improvements for some tests, this test suit were proposed [5, 12, 23]. Walker proposed a test suite ENT including five tests [43] and Srinivasan et al. proposed a test suite test problems in parallel implementations of PRNGs [33]. Although generated sequences can be long or short, in all of these test suites, the length of the sequences was disregarded and usually some approximations are used in calculations. Koçak [19] and Sulak [36] proposed tests that can be used to evaluate short sequences. In doing so, they calculated the exact probabilities up to the necessary decimal point of the test statistics. Fan et al. proposed an algorithm to test chaotic binary sequences from a new perspective of periodicity [8].

A randomness test is defined in three steps. The first step is to define a random variable $X : \Omega \to \mathbb{R}$, on the set of sequences Ω, and then to compute its probability distribution function $F(X)$. The second step is computing $X(\sigma) = t_p$ where $\sigma \in \Omega$ is the sequence under consideration. The last step is to decide whether the sequence under consideration looks like a random sequence.

While performing statistical tests, the null hypothesis H_0, stating that the number generator under consideration can not be distinguished from a true random number generator, is tested. The alternative hypothesis H_a is number generator is not random. For each case, there exists an error type given in the Table 1 [29].

Table 1. Hypothesis testing

Random sources	Accept H_0	Accept H_a
Generator random	Correct	Type 1 error
Generator non-random	Type 2 error	Correct

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

$$P(X(\sigma) = t_p) \begin{cases} \leq \alpha & \sigma \quad \text{is not random} \\ > \alpha & \sigma \quad \text{passes the test.} \end{cases} \tag{1.1}$$

When the sequence is reasonably short, computing $X(\sigma)$ and $P(X(\sigma) = t_p)$ can be easy, but as the length of sequence gets longer, computing $P(X(\sigma) = t_p) = F(t)$ is generally getting more and more difficult, unless it is impossible in practice. In the literature, to overcome this computational problems, combinatorial properties are used. To compute $F(t)$ for large values of n, some alternative approaches are required. For example, for the weight test, computing exact value of $P((w(\sigma) = k) = \frac{1}{2^k}\binom{n}{k}$ directly is easy until length of the sequence n is about 1000. As n gets larger, this computation is not feasible. Since, in most of the cases, direct computation is infeasible for large values, obtaining an explicit expression of F is not enough. A feasible way of computing $F(t)$ is by making use of recursions which is an advantageous method for reasonably long sequences [19]. However, as the length of the sequence gets even larger, recursions are not helpful either. For this reason, an accurate approximation of statistics of the random variables should be derived to compute test results. For the weight test, when n is large enough, the distribution of F approaches to the normal distribution. Although like weight and run [29], for some tests using approximations give results, there is no accurate approximation for many other tests used for cryptographic purposes.
Besides direct computation and making use of approximations, the third approach is testing a collection of short subsequences of the sequence since all computations are much easier for short sequences.

Let $\sigma_1, \sigma_2, \ldots \sigma_N$ be N overlapping or non-overlapping parts of σ with an order, and $X(\sigma_1), X(\sigma_2), \ldots, X(\sigma_N)$ be $t-values$ of statistical randomness test depending on random variable of X. Then, distribution values of X over Ω_N and distribution values of X over $\{\sigma_1, \sigma_2, \ldots, \sigma_N\}$ should be similar by means of χ^2 distribution.

Definition 1.1. Let $\sigma = s_1, s_2, \ldots, s_n$ and

$$\sigma_1 = s_1, s_2, \ldots, s_{t_1}, \quad \sigma_2 = s_{t_1+1}, s_{t_1+2} \ldots, \sigma_{t_1+t_2}, \quad \sigma_N = s_{t_{N-1}+1}, s_{t_{N-1}+2}, \ldots, s_n$$

be a partition of σ. If $t_i = t_j$ for all $i \neq j$, then length of all subsequences are equal and it is called **fixed length partition** otherwise **variable length partition**.

If the given partition is a fixed length partition, then the χ^2 distribution can be used directly. One can also define a variable length partition by making use of the random variable under consideration. Let $\xi \in X(\sigma)$ be any element from the range of the random variable X, and $l_0 \in \mathbb{Z}^+$ be fixed as maximal length. We define a new random variable $X_\xi : \Omega \mapsto \mathbb{Z}^+$ as

$$X_\xi(\sigma_t) \mapsto \begin{cases} l_0 & \text{if} \quad X(s_{b(t)}, \dots, s_{b(t)+j}) \neq \xi \quad for \quad j = 0, \dots, l_0 - 1 \\ 1 + min_j\{j \mid X(s_{b(t)}, \dots, s_{b(t)+j}) = \xi\} & \text{otherwise} \end{cases} \tag{1.2}$$

where b_t is defined by setting $b_1 = 1$ and $b_{t+1} = b_t + X_\xi(\sigma_t)$ for $t > 1$. We call the subsequence $\sigma_t = s_{b(t)}, \dots, s_{b(t)+1}$ the t^{th} part of σ. Notice that lengths of each part in this variable length partition are determined by the random variable X.

To make the definition of the new random variable more clear, we give an example and hence from a random variable X, how the new random variable X_ξ is defined and how this new variable defines a variable length partition of a sequence σ will be more clear. Let the random variable X be the weight function, that is the number of 1's in the given sequence σ. Fix $l_0 = 8$ as maximal length and also fix $\xi = 3$. Values of the new random variable $X_\xi : \Omega \mapsto \mathbb{Z}^+$ on various sequences are given below:
$X_\xi(1,1,0,1,0,1,1,0,0,1,0,1,1,1,0,0,1) = 4$, since the sequence gets X value equal to 3 at the index 4 for the first time,
$X_\xi(0,1,0,0,0,1,1,0,0,1,0,1,1,1,0,0,1) = 7$, since the sequence gets X value equal to 3 at the index 7 for the first time,
$X_\xi(1,1,0,0,0,0,0,0,0,1,0,1,1,1,0,0,1) = 8$, since the sequence does not get X value equal to 3 until the predefined maximal index $l_0 = 8$.
Also notice that X_ξ defines a variable length partition of each given sequence. As an example with above parameters, the sequence $\sigma = (1,1,0,1,0,1,1,0,0,1,1,1,1,0,0,1,1,0,0,0,0)$ will be partitioned as
$\sigma_1 = (1,1,0,1), \quad \sigma_2 = (0,1,1,0,0,1) \quad \sigma_3 = (1,1,1), \quad \sigma_4 = (0,0,1,1,0,0,0,0)$
of lengths $b_1 = 4, \quad b_2 = 6, \quad b_3 = 3, \quad b_4 = 8$ respectively.

In this paper, tests are defined and classified in three groups:

- those applied to the entire sequence directly,
- those applied to the collection obtained from the sequence using a fixed-length partition,
- a new approach called dynamically partitioned subsequences defined by [2].

In the last approach, each bit of the sequence effects the test result, no part of the sequence is discarded. This approach is constructed by inspiring one of the entropy estimation method recommended by NIST; the collision estimate method [39].

In this paper, we propose a new test suite to test long sequences. We use tests defined in [2] and tests for the entire sequence and fixed-length partitioning method in order to complete the test suite. In the entire sequence method, two basic tests named as weight and run [29] are used, while in the fixed-length partitioning method, five different tests named weight test, run test, linear complexity test, and integer coverage test [20] are used. Weight, run, and linear complexity tests are selected to make a meaningful comparison, and coverage tests are selected to show the difference between dynamic partition methods and fixed-length partition methods. In addition to available tests in the literature, three new tests, named R2 run test, dynamic run test, and dynamic saturation point test are also proposed to complete the test suite for long sequences. The main contribution of this paper is to indicate some weaknesses in the suites in the literature by propose a test suite for long sequences which employs all three approaches: testing the entire sequence, testing the fixed length partitions and testing the dynamic partitions, that can detect non random sequences which other test suites can not detect.

According to [32], independence and coverage are important issues. Turan et al. [40] observed that some tests for short sequences on the NIST test suite are correlated . Doğanaksoy et al. [6] and Sulak et al. [37] observed those correlations and dependencies in NIST test suite. In addition to these, there are many studies in the literature about correlations, independence, and coverage of statistical randomness tests [9, 15, 17]. When forming proposed test suite LS14, mutual correlations between the tests are also considered.

The rest of this paper is organized as follows. In the second chapter, all the definitions and the statistical calculations of the propose tests are given. In the third chapter, mutual correlations of propose tests are given. In the fourth chapter, the comparisons of propose test suite and NIST test suite are given [29]. In the last chapter, the conclusion is given.

2. Randomness tests in the LS14 test suite

In this section, definitions and necessary computations of statistical randomness tests that are used in the LS14 test suite are given. The suite consists of the following 14 tests

(1) Entire Weight Test (E. Weight)
(2) Entire Run Test (E. Run)
(3) Entire R2 Run Test (E. R2 Run)
(4) Fixed Length Weight Test (F.L Weight)
(5) Fixed Length Run Test (F.L Run)
(6) Fixed Length Linear Complexity Test (F.L LC)
(7) Fixed Length Integer Coverage Test (F.L Coverage)
(8) Dynamic Partition Weight Test (Dyn. Weight)
(9) Dynamic Partition Run Test (Dyn. Run)
(10) Dynamic Partition Linear Complexity Test (Dyn. LC)
(11) Dynamic Partition Integer Collision Index Test (Dyn Coll. Ind)
(12) Dynamic Partition Integer Collision Distance Test (Dyn. Coll. Dist)
(13) Dynamic Partition Saturation Point Test (Dyn. Saturation)
(14) Dynamic Partition Index Coincidence Point Test (Dyn. ICT)

2.1. Literature tests

In this subsection, brief definitions of randomness tests in the literature used in the test suite are given.

2.1.1. Weight test.

Definition 2.1. The number of terms equal to 1 in a binary sequence is called the weight of the sequence.

Weight test is one of the most basic tests in the literature and is used in almost every test suite. In all three approaches, we used weight test as follows:

- Entire weight test [29]: Evaluate the weight of the long sequence by using normal distribution to approximate binomial distribution.
- Fixed-length weight test [19]: Partition the long sequence to obtain a collection of fixed length short sequences and evaluate the collection using χ^2 distribution.
- Dynamic partition weight test [2]: Partition the long sequence into short sequences of variable length determined by a prefixed value for the weight. Then evaluate the collection of the lengths of subsequences using χ^2 distribution.

2.1.2. Run test.

Definition 2.2. In a binary sequence, an uninterrupted sub-sequences of identical bits are called run [20].

We use run test for all three approaches: Two of them are the entire total run test [29], and fixed-length run test [19]. Apart from these two tests,in section 2.4 we propose the third one as run test in dynamic partitioning method, in this paper. In addition to these there, as a fourth approach, R2 run test is proposed in this paper about the length of runs.

2.1.3. Linear complexity test.

Definition 2.3. Let σ be a binary sequence, linear complexity of σ is the length of the shortest linear feedback shift register(LFSR) that can produce σ.

We used linear complexity test in two ways, fixed length linear complexity test [19], [29] and dynamic partition linear complexity test [2].

2.1.4. Integer coverage test.

Definition 2.4. Let $\tilde{S} = \tilde{S}_1, \ldots, \tilde{S}_n$ be an integer sequence. The cardinality of the set $I = \{\tilde{S}_1, \ldots, \tilde{S}_n\}$, that is, the number of distinct terms of the sequence $\tilde{S}$ is called the coverage of the sequence $\tilde{S}$.

In the literature, Knuth [20] proposed a version of coverage test namely the coupon collector test in his book. Sulak defined it as a test in [36].

Since we will use the computation of integer coverage test in Section 2.5, namely the dynamic saturation point test, we will give a detailed explanation of this test.

Consider a binary sequence S of length $n = b2^b$ to be converted to $b-$bit integer sequence $\tilde{S}$ of length 2^b. Let c be the number of different integer in $\tilde{S}$. Then, each b-bit integer can take 2^b different values. Since we know c different integers in the sequence, there can be $\binom{2^b}{c}$ different integer sets. Since the length of the sequence is equal to 2^b and there are exactly c different integers, the total number of each integers' frequency is 2^b. The number of the piece of the integers separated in the sequence is $\left\{{2^b \atop c}\right\}$, where $\{\}$ is the Stirling numbers of the second kind. Finally, there are $c!$ different orders of integers. Then, the probability of the sequence $\tilde{S}$ having c different integers is:

$$P_C(\tilde{S} = c) = \frac{c!}{2^{b2^b}} \binom{2^b}{c} \left\{{2^b \atop c}\right\}. \tag{2.1}$$

2.1.5. Integer collision tests.

Definition 2.5. Let $\tilde{S}$ be an integer sequence with the first t entries that are different and $(t+1)^{th}$ entry is equal to one of the first t entries where $t = 1, 2..., 2^b - 1$ then *the index* $(t+1)$ and the $(t+1)^{th}$ entries are called *collision index* and *colliding integer*, respectively [2].

In the literature, collision test is one of the most appropriate randomness tests for the dynamic partitioning method. Two types of collision test [2] are used in the test suite.

2.1.6. Index coincidence point test.

Definition 2.6. The point where the entry of the sequence and the index of the sequence coincide is called index-coincidence.

This test is proposed in [2] as a new test. We used this test in order to show the benefits of this test.

2.2. Proposed tests

In the literature, there are lots of statistical randomness tests. Only a few of them can directly be applied to long sequences. For example, the frequency test and the run test in the NIST test suites [29] directly depends on the number of 1's and the number of runs, respectively, and hence can directly be used for long sequences. However, evaluating sequences like in these tests for most of the other tests can be misleading. In the paper [2], the dynamic partitioning method is defined, and four tests are proposed. In this paper, we used this approach and proposed two more tests. Moreover, we proposed one more test without using any partitioning method.

2.3. R2 run test

In this chapter, we proposed a new test depending on the number of runs. Recall Golomb's second postulates: if the sequence length is n, the expected number of runs is $n/2$. At least half of the total number of runs of 0's or 1's should have length one, at least one-fourth should have length 2, at least one-eighth should have length 3, and so on.

There are many run tests in the literature. Some of them are designed according to the total number of runs or lengths of the runs. However, neither of them fully corresponds to the second postulate of Golomb. When testing long sequence, this postulate of Golomb is usually modified as follows cited: In a random binary sequence, the expected number of runs of length one is half of the total number of runs, expected number of runs have length two one-fourth of the total number of runs and the like [11]. This test aims to evaluate the sequence exactly like in Golomb's original postulate and determine whether frequencies of the length of runs are uniformly distributed or not. Since this test depends on the total number of runs, if the sequence fails the total number of run test [29], this test should be omitted. To apply χ^2 test properly with meaningful probability, bin values should be appropriately distributed. It is recommend that, R2 run test should be applied only for long sequences.

Test description:

The following 4 steps should be performed;

(1) Determine the total number of runs.
(2) Determine the total number of runs of length i, R_i, for $i = 1, 2, 3, 4, 5, 6, 7, \geq 8$.
(3) Using bin values given table 2, compute χ^2 value .
(4) Compute the p-value $\Gamma(\chi^2, 7)$. Note that last bin can be calculated by using the other values.

Note that, in cryptographic applications, degree of freedom usually chosen as 7 or 9 depending on the sensitivity of the tests.

Bin bounds and probabilities

The expected number of runs of length i, that is $E(r_i)$, of a sequence σ of length n can be computed as follows.

$E(r_1)$**:** Let $\sigma' = s'_1, s'_2, \ldots, s'_{n-1}, 1$ be derivative of σ. Then each run in σ' is in one to one correspondence with $1's$ in σ' that is $s'_i = 1$. Except from first and the s'_{n-1} term runs of length 1 means that $s'_i = s'_{i+1} = 1$. For the first term and and the s'_{n-1} term it is enough to $s'_1 = 1$and $s'_{n-1} = 1$. Then expected number of runs of length 1, that is $E(r_1)$ can be computed as sum of

$$P(s'_1 = 1) = P(s'_{n-1} = 1) = \frac{(r-1)}{n-1} \tag{2.2}$$

$$P(s'_i = s'_{i+1} = 1) = \frac{(n-2)(n-1)(r-2)}{(n-1)(n-2)} \tag{2.3}$$

and hence

$$E(r_1) = \frac{r(r-1)}{n-1} \tag{2.4}$$

$E(r_2)$**:** Except from the first two and the last three terms, runs of length 2 means that $s'_i = s'_{i+2} = 1$ and $s'_{i+1} = 0$. For the first term and and the s'_{n-2}, s'_{n-1} term , it is enough to $s'_1 = 0$ and $s'_2 = 1$ and $s'_{n-1} = 0$ and $s'_{n-2} = 1$. The expected number of runs of length 2 can be computed as sum of

$$P(s'_1 = 0 \text{ } and \text{ } s'_2 = 1) = P(s'_{n-1} = 0 \text{ } and \text{ } s'_{n-2} = 1) = \frac{(r-1)(n-r-1)}{(n-1)(n-2)} \tag{2.5}$$

$$P(s'_i = s'_{i+2} = 1 \text{ } and \text{ } s'_{i+1} = 0) = \frac{(n-3)(r-1)(r-2)(n-r-1)}{(n-1)(n-2)(n-3)} \tag{2.6}$$

and hence

$$E(r_2) = \frac{r(r-1)(n-r-1)}{(n-1)(n-2)} \tag{2.7}$$

$E(r_i)$**:** For the general case, we have

$$E(r_i) = \frac{r(r-1)(n-r-1)\dots(n-r-(i-1))}{(n-1)(n-2)\dots(n-t)} \tag{2.8}$$

Table 2. Bin values of R2 run test

Bins	1	2	3	4	5	6	7	8
Length of Run	1	2	3	4	5	6	$7-8$	≥ 9
Expected Count	$E(r_1)$	$E(r_2)$	$E(r_3)$	$E(r_4)$	$E(r_5)$	$E(r_6)$	$E(r_7)$	$E(r_{\geq 8})$

2.4. Dynamic partition run test

As in the weight test in [2], we determine the partition point as the number of runs reaches the predestined values. Let S be an n bit binary sequence, and r is the predestined number of runs. In each step, we record the first index k when the number of runs reaches r and this subsequence is deleted. This process is repeated throughout the sequence. When the number of runs does not reach to number r before the index t, we record the index t and the subsequence is deleted. t is chosen so that the probability of not reaching the number of runs r till index t is negligible. The process is familiar with weight test in this method.

Example 2.7. Let $S = 0011100001010000000111$ and take $r = 2$, $t = 7$. (Notice that the probability of not reaching the number of runs $r = 2$ until $t = 7^{th}$ term is $\frac{1}{2^8}$)
Since end of each run in the sequence S corresponds to term 1 in the derivative sequence, it is easy the count the number of runs there. Lets compute the derivative sequence S'

$S' = 010010001111000000100$ then

$S'_1 = 01001$, $S'_2 = 00011$, $S'_3 = 11$, $S'_4 = 0000001$, the rest is deleted. Hence, $r_2 = 1$, $r_5 = 2$, $r_{\geq 7} = 1$

Bin bounds and probabilities

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

Let S be a binary sequence. Then compute S' to determine the number of runs. If S is random, then it is expected that S' is also random. Let r be the predestined number of runs to define a partition of the sequence. In other words, r is the prefixed weight of S' that will be used to partition the S'. The only difference between this test and to previous weight test is since S' is generated from the original sequence S by computing $\sigma_i \oplus \sigma_{i+1}$, when i^{th} term of S' is generated in the original sequence the $(i+1)^{th}$ index is generated. Other computations are the same with weight test. However, it can be seen from 3 bin bounds are different.

The suggested parameter values for the r and t are 128 and 281, respectively. The range with respect to $r = 128$ is $\{128, 129, \cdots\}$. Bounds and corresponding probabilities of bins can be seen in the following table 3

Table 3. Bin values of total run test in Dynamic Partition Method

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

2.5. Dynamic saturation point test

Definition 2.8. Let $\tilde{S}$ be a sequence where its terms are from the set $I = \{0, 1, \ldots, t\}$. The first index i at which all elements of I appear in $\tilde{S}_1$, $\tilde{S}_2$, ..., $\tilde{S}_i$, that is $\{\tilde{S}_1, ..., \tilde{S}_i\} = I$, is called the saturation point.

A sequence can be dynamically partitioned by using its saturation points. The entire sequence testing method should not be used as the saturation point will be at the very beginning of the sequence and hence its tail will be discarded leading loss of information. The fixed-length partition method should not be used either, as when the cardinality of the set I is big, most of the short-length subsequences cannot achieve saturation. In both cases, the test result will be misleading.

As in the collision test, the random variable used in the collision estimate method, described in [39], suggests partitioning sequences dynamically. The idea of this test depends on [36]. Koçak proposed saturation point as a test in [19]. In both, saturation point tests are applied to short sequences. However, in this paper, we proposed saturation test for long sequences. From the definition of saturation point test, the rest of the sequences after saturation point is not tested in a single part of the sequence. However, by using dynamic partitioning method, all part of sequences is tested. From the example 2.10 it will be understood well.

Remark 2.9. Residual part for a long sequence will not change the result. It will be the negligible.

Example 2.10. Let $S = 10010111110001100101100111001100011 00$ and fix the block size as $b = 2$, and obtain an integer sequence with terms in $I = \{0, 1, 2, 3\}$. Then, $\tilde{S} = 2113301211213030 12$ and delete residual part, that is the last term, which is negligible. Then, subsequences are $\tilde{S}_1 = 211330, \tilde{S}_2 = 12112130, \tilde{S}_3 = 3012$ are generated from the definition of saturation point 2.8. Length of subsequences are 6, 8, 4 respectively.

Notice that, this gives the dynamic partition of S as 100101111100, 0110010110011100, 11000110.

Bin bounds and probabilities

Proposition 2.11. *Let $\tilde{S}$ be a $b-bit$ integer sequence of length n. Let $\tilde{S}_k$ be number of sequences with the first saturation point at the k^{th} index. Then until the k^{th} index there should be $2^b - 1$ different integers and the last integer should appear at the k^{th} index. This means that the first $k-1$ terms cover $2^b - 1$ integer. At the end the k^{th} term should be equal to the non appearing integer in the first $k-1$ terms, probability of this is $\frac{1}{2^b}$. From the equation 2.1 we get;*

$$P_K(\tilde{S}_k = k) = \frac{1}{2^b}\frac{(2^b-1)!}{2^{b^{k-1}}}\binom{2^b}{2^b-1}\left\{\begin{matrix} k-1 \\ 2^b-1 \end{matrix}\right\}. \tag{2.9}$$

Saturation test is one of the most appropriate tests for the dynamic partition method. However, in a binary sequence, there may not be any saturation point. In this paper, to overcome this problem, we introduce a fixed-length index t and if saturation point does not happen until index t, partition the sequence from there and start to search for the next saturation point again.

Table 4. Bin values and expected probabilities of saturation point test for $b = 8$

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

3. Mutual correlation of tests

In this paper, we proposed a new test suite. According to [6] correlations of tests in a test suite is important and affect the coverage of the test result. In this section, correlation of the tests in the test suite with each other are given. To find the correlation between the test that can be applied to long sequences, we choose a sample set of tests that dynamic partition method can be applied or fixed-length partition can be applied, or entire sequence can be tested without any partitioning. In this paper, we computed the correlation of entire, fixed-length, and dynamic length weight and run tests, entire R2 run test, fixed-length, and dynamic length linear complexity, fixed-length coverage test, dynamic collision tests, saturation point tests, and index point coincidence tests [2,19,29]
.

For discovering correlations of proposed tests, 1000 random number sequences of length $4,000,000$ are generated. In other words, approximately 4 billion bits are generated. For each generated sequence, p-values are evaluated from all proposed tests. Pearson Correlation [27] is used as a measure of correlation. Results of all tests are given in table 5.

According to Table 5, the weight tests (entire, fixed length and dynamic partition) are correlated with each other and the run tests (entire, fixed length and dynamic partition) are correlated with each other when the correlation boundary is 0.1 as in [6], and they are half correlated according to [27]. This result is expected as they use the same random variable.

3.1. Proposed test suite

It is better to apply all proposed tests. All suggested tests run in insignificant time for a 32 million bit sequence, except for the dynamic linear-complexity test. But, because of long execution time of the dynamic linear-complexity test, to get a quick result, the test suite can be used excluding this test. The reference program is available in [46].

Table 5. Pearson correlation results of tests

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

Table 6. Test results of sequences generated by Quantis QRNG

Sequence Length	1000000	2000000	4000000	8000000	16000000	32000000
E. Weight	0.394214	0.574977	0.709520	0.525901	0.817120	0.639456
E. Total run	0.925109	0.852469	0.606553	0.322371	0.823146	0.869414
E. R2 run	0.843440	0.585994	0.446997	0.591846	0.403512	0.245524
F.L Weight	0.079366	0.26239	0.102383	0.221871	0.771908	0.144076
F.L Run	0.784187	0.946985	0.158243	0.486801	0.203090	0.254156
F.L LC	0.405698	0.986085	0.042022	0.044577	0.939479	0.362193
F.L Coverage	0.289969	0.794233	0.671157	0.780034	0.999115	0.274191
Dyn. Weight	0.406396	0.128787	0.444505	0.893928	0.706713	0.629052
Dyn. Run	0.941925	0.579412	0.213063	0.391849	0.450734	0.519474
Dyn. LC	0.483495	0.978775	0.350613	0.917682	0.256289	0.828645
Dyn. Saturation	0.288721	0.516936	0.426526	0.213987	0.619955	0.415656
Dyn. Coll. index	0.466814	0.343573	0.069285	0.768003	0.367716	0.213964
Dyn. Coll. distance	0.417817	0.928356	0.463415	0.745665	0.827976	0.331304
Dyn. ICP	0.013587	0.257699	0.171938	0.685735	0.152750	0.415301

3.2. Test results of quantis QRNG and glitter lamps

In this section, LS-14 test suite results of Quantis QRNG [45] and Glitter Lamps RNG [4] are given. For this purpose, six different length, 1000000 bit, 2000000 bit, 4000000 bit, 8000000 bit, 16000000 bit, 32000000 bit, sequences are generated from [45] and Glitter Lamps RNG tested. The results are given in Table 6 and 7.

4. Sensitivities of tests and comparison of test suites

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

Table 7. Test results of sequences generated by Glitter Lamps

Sequence Length	1000000	2000000	4000000	8000000	16000000	32000000
E. Weight	0.569356	0.345537	0.421685	0.318239	0.941209	0.889632
E.T. Run	0.256794	0.456096	0.781012	0.97095	0.607776	0.701795
E. R2 Run	0.379692	0.214387	0.081357	0.647288	0.669653	0.496604
F.L Weight	0.934969	0.680207	0.378161	0.251712	0.691331	0.902557
F.L Run	0.215943	0.328728	0.514668	0.809655	0.536009	0.333519
F.L LC	0.123035	0.150581	0.147389	0.076084	0.368838	0.406806
F.L Coverage	0.865755	0.545644	0.681228	0.823591	0.322887	0.197411
Dyn. Weight	0.325934	0.031267	0.127835	0.249216	0.965399	0.847572
Dyn. Run	0.428734	0.382647	0.813936	0.780547	0.911485	0.975468
Dyn. LC	0.453966	0.080612	0.322055	0.009997	0.041799	0.308587
Dyn. Saturation	0.550701	0.838141	0.938623	0.771977	0.369905	0.433015
Dyn.Coll.Ind.	0.227801	0.361846	0.037257	0.747914	0.635151	0.210667
Dyn.Coll.Dist	0.964845	0.607877	0.586102	0.516497	0.24237	0.708378
Dyn. ICP	0.315094	0.08462	0.01512	0.055629	0.000796	0.015863

Table 8. Test results of biased sources

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

4.1. Test results of bias sources

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

Case 1 : Modifying the probability of generating 1 and 0

In a random binary sequence, it is expected that 1's and 0's are generated with equal probability. In this section to determine to evaluate sensitivities, the same random source is manipulated by generating 1 with probability $\frac{1}{2} + p$ where $-\frac{1}{2} \leq p \leq \frac{1}{2}$. The generated sequences are 16000000-bit length.

In order to evaluating sensitivities and power of the tests against non balanced sequences, the same random number generator is manipulated. From table 8, among all methods, weight tests are the first tests detecting the bias since it measure directly number of ones. All tests except linear complexity tests detected non-random sequences when the bias is 0.02. Because of the test structure, linear complexity tests could not detect biased sequences. In Table 8, column names show bias probability.

Case 2 : Modifying a generating template

In this case, we give bias to generating templates. For example, for each 10^{th} "1000" template generated, we change it with the "1100" template. As generated non-random sequences, we use the first 4000000 bits of π. For different cases, we change non-overlapping templates and overlapping templates. The results are given in table 9.

By manipulating the sequences in this way, we able to detect the sensitivity of the tests in the proposed test suite LS14 to unbalanced sequences in terms of the number of templates and to sequences with disrupted run counts.

For non-overlapping templates, integer tests are more sensitive, and for overlapping tests, run tests, especially R2 Run Test, are more sensitive. When different weight templates are chosen to be modified, biased sequences fail; in other cases, biased sequences pass from the weight tests.

In Table 9, columns are results of biased 4000000 bit of π. Biased methods are given in followings

(1) At the first column for each third 0111000 template changed with 00111000.
(2) At the second column for each fifth 00001000 template changed with 00000010
(3) At the third column for each tenth 0001 template changed with 0010
(4) At the fourth column for each twentieth 0001 template changed with 0010

Table 9. Test results of biased 4000000 bit of π

	1	2	3	4
E. Weight	0.761508	0.761508847	0.761508847	0.761509
E. Total Run	0.953748	0.953748629	1.49E-19	4.97E-10
E. R2 Run	2.78E-12	2.52E-32	1.46E-68	2.95E-16
F.L. Weight	0.766230653	0.794147626	0.842721336	0.856401
F.L. Run	0.467805	0.978681911	9.08E-111	7.56E-16
F.L. LC	0.7695767	0.709798666	0.180798564	0.55676
F.L. Coverage	0.3860949	0.131851751	0.394313052	0.148469
Dyn. Weight	0.351313	0.325022708	0.271946719	0.289364
Dyn. Run	0.848894	0.812718761	9.48E-111	3.96E-16
Dyn. LC	0.52973	0.857565413	0.71828367	0.665208
Dyn. Saturation	0.301674	0.80528541	0.367098526	0.154132
Dyn. Coll. Index	0.012233	0.090185203	1.52E-09	0.000613
Dyn. Coll. Distance	0.2900747	0.640132634	0.315079052	0.379265
Dyn. ICP	0.306376207	0.868683345	1.45E-01	0.047406

Case 3 : Repeating sequence with long period

In this case, we use two different methods; one is repeating the random sequence, and the other is repeating the random sequence with each term of the sequence inverted. For this case, we use bits of π.

By generating sequences in these ways, we measured the sensitivity of the tests in the proposed test suite LS14 to periodic sequences.

In Table 10, columns are results of repeating bit of π. Biased methods are given in followings

(1) At the first column, first 2000000 bit of π is added end of the first 2000000 bit of π. By the way 4000000 bit sequence is generated.
(2) At the second column, 1000000 bit of π is added end of the first 2000000 bit of π. By the way 3000000 bit sequence is generated.
(3) At the third column, first 4000000 bit of π is added end of the first 4000000 bit of π. By the way 8000000 bit sequence is generated.
(4) At the fourth column, first 2000000 bit of π is added end of the first 4000000 bit of π. By the way 6000000 bit sequence is generated.

Table 10. Test results of repeating bit of π

	1	2	3	4
E. Weight	0.739123	0.975128	0.667767	0.701161
E. Total run	0.887474	0.87931	0.934627	0.991856
E. R2 run	0.739134	0.403088	0.525452	0.642347
F.L Weight	0.497328	0.671859	0.397206	0.738422
F.L Run	0.848001	0.665962	0.05919	0.584432
F.L LC	0.037603	0.096186	0.654556	0.315588
F.L Coverage	0.001103	0.01482	0.000686	0.000392
Dyn. Weight	0.719222	0.898615	0.232539	0.364713
Dyn. Run	0.420331	0.805917	0.957034	0.826496
Dyn. LC	0.104981	0.410588	0.060226	0.199087
Dyn. Saturation	0.006804	0.050247	0.784898	0.359986
Dyn. Coll. index	0.045062	0.35241	0.000529	0.007022
Dyn. Coll. distance	0.000904	0.034323	0.021276	0.024306
Dyn. ICP	0	4.21E-07	0	3.05E-06

Table 11. Test results of repeating complement bit of π and e

	1	2	3	4	5
E. Weight	1	0.910934	1	1	0.746134
E. Total run	0.887869	0.991531	0.934908	0.788704	0.756356
E. R2 run	0.066397	0.641874	0.525182	0.009200	0.034696
F.L. Weight	0.841868	0.935787	0.918687	0.105233	0.191092
F.L. Run	0.848001	0.584432	0.05919	0.613499	0.659686
F.L. LC	0.128253	0.589482	0.801001	0.58827	0.594375
F.L. Coverage	0.001103	0.000456	0.000757	0.205925	0.252846
Dyn. Weight	0.803966	0.379659	0.3406745	0.05455	0.104959
Dyn. Run	0.434369	0.831704	0.95146	0.922812	0.904922
Dyn. LC	0.284296	0.173834	0.2743891	0.981879	0.975726
Dyn. Saturation	0.006917	0.374526	0.783355	0.208226	0.334155
Dyn. Coll. ind.	0.364399	0.088629	0.027327	0.00088	0.007712
Dyn. Coll. dist.	0.000863	0.024306	0.021276	0.642837	0.719439
Dyn. ICP	0.288143	0.602396	0.730529	0.339384	0.145964

From Table 10, for different non-random sequences of different lengths which are generated by repeating itself, at least one of the tests in the test suite fails.

In Table 10, columns are results of repeating bit of π and e. Biased methods are given in followings

(1) At the first column, complement of first 2000000 bit of π is added end of the first 2000000 bit of π. By the way 4000000 bit sequence is generated.
(2) At the second column, complement of first 2000000 bit of π is added end of the first 4000000 bit of π. By the way 6000000 bit sequence is generated.
(3) At the third column, complement of first 4000000 bit of π is added end of the first 4000000 bit of π. By the way 8000000 bit sequence is generated.
(4) At the fourth column, complement of first 4000000 bit of e is added end of the first 4000000 bit of e. By the way 8000000 bit sequence is generated.
(5) At the fifth column, complement of first 2000000 bit of e is added end of the first 4000000 bit of e. By the way 8000000 bit sequence is generated.

From Table 11, for different non-random sequences of different lengths which are generating by repeating complement of the sequence, at least one of the tests in the test suite fails.

4.2. Tests results of NIST test suite and comparisons of tests

In this section, test results of NIST test suite [29] and according to results of LS14 and NIST test suite comparisons are given. In NIST test suite originally there were 16 tests. Because of some computations problem Fast Fourier Test was removed from the test suite. Although in the NIST test suite there are 15 tests since cumulative sum test include 2 test, serial test include 2 test, random excursion test include 8 test, random excursion variant test include 18 test and non overlapping template matching test include 147 test, there are actually 186 tests. Since non overlapping template matching test and random excursion variant test include too many tests we give pass rate of these tests. In NIST test suite, if the sequence is not passed from Frequency test some tests are not applicable. Because of these reason, in some cell of Table 13 there are NA value.

In Table 9 we give the sensitivities of tests in LS14 to non random sequences generated with changing some templates to a fixed template. In Table 12 we give the test result of NIST test suite to same non random sequences.

Table 12. NIST test results of changed template of 4000000 bit of π

Test Name	1	2	3	4
Approximate Entropy	0	0	0	0
Frequency Test within a Block	0.153092	0.13298	0.149941	0.149941
Cumulative SUM 1	0.370864	0.370864	0.370864	0.370864
Cumulative SUM 2	0.533527	0.533527	0.533527	0.533527
Frequency	0.543851	0.543851	0.543851	0.543851
Linear Complexity	0.73799	0.44827	0.742285	0.741376
Test for the Longest Run of Ones in a Block	0.1898291	0.188291	0.205234	0.188291
Non-Overlapping Template Matching	135/147	121/147	64/147	107/147
Overlapping Template Matching	0.590462	0.590462	0.471968	0.667088
Random Excursion V(-1)	0.790082	0.796378	0.820586	0.821615
Random Excursion V(-2)	0.834858	0.904335	0.80511	0.76303
Random Excursion V(-3)	0.744884	0.876281	0.785522	0.843112
Random Excursion V(-4)	0.122652	0.064636	0.223694	0.098602
Random Excursion V(1)	0.609301	0.447682	0.835204	0.640652
Random Excursion V(2)	0.271901	0.329291	0.268735	0.319342
Random Excursion V(3)	0.177687	0.175985	0.240543	0.274554
Random Excursion V(4)	0.108394	0.097079	0.08507	0.134298
Random Excursion Variant	18/18	18/18	18/18	18/18
Run	0.908591	0.908591	0	0
Binary Matrix Rank	0.084759	0.825892	0.534422	0.919951
Serial1	0	0.0007	0	0.014521
Serial2	0.668664	0.863105	0.58817	0.574905
Universal	0.06759	0.043544	0.744355	0.08562

According to Table 12, approximate entropy, serial 1 and some of non-overlapping template matching tests are sensitive to changing templates. Moreover, if the template changing template length is 4 run test is also sensitive for biased sequences.

In Tables 8,10,11 we give the sensitivities of tests in LS14 to non random sequences. In Table 13, test result of random sources π, e and same sequences tested in Tables 8,10,11. The tested sequences explanations are given in followings;

(1) At the first column 16000000 bit π is tested.
(2) At the second column 16000000 bit e is tested.
(3) At the third column, first 4000000 bit of π is added end of the first 4000000 bit of π. By the way 8000000 bit sequence is generated.
(4) At the fourth column, complement of first 2000000 bit of π is added end of the first 4000000 bit of π. By the way 6000000 bit sequence is generated.
(5) At the fifth column, complement of first 4000000 bit of π is added end of the first 4000000 bit of π. By the way 8000000 bit sequence is generated.
(6) At the sixth column, complement of first 4000000 bit of e is added end of the first 4000000 bit of e. By the way 8000000 bit sequence is generated.
(7) At the seventh column, complement of first 2000000 bit of e is added end of the first 4000000 bit of e. By the way 6000000 bit sequence is generated.
(8) At the eighth column, random sequences generated from library of $C\#$ which is generate 0 with probability 0.49999 and 1 with probability 0.50001
(9) At the ninth column, random sequences generated from library of $C\#$ which is generate 0 with probability 0.4999 and 1 with probability 0.5001
(10) At the tenth column, random sequences generated from library of $C\#$ which is generate 0 with probability 0.499 and 1 with probability 0.501
(11) At the eleventh column, random sequences generated from library of $C\#$ which is generate 0 with probability 0.49 and 1 with probability 0.51

According to Table 13, random sources passed all tests. For none random sources,

(1) For the third column approximate entropy, 13 non overlapping template matching test, longest run with in a block test, and serial tests fail.
(2) For the fourth column only 3 non overlapping template matching test fail.
(3) For the fifth column only 2 non overlapping template matching test fail.
(4) For the sixth column none of the tests fails.
(5) For the seventh column none of the tests fails.
(6) For the eight column none of the tests fails.
(7) For the ninth column none of the tests fails.
(8) For the tenth and eleventh column most of the tests fail or not applicable.

Comparisons of the results obtained from the NIST test suite and LS14 test suite are given as follows:

- According to Table 8 and 8-11 columns of table 13, weight tests (frequency test and frequency test with in a block in NIST test suite and entire weight test, fixed length weight test and dynamic weight test in LS14), both test suites detect non-random sequences when the biases are greater then 0.001.
- Table 9 shows that non-random sequences fail from the entire R2 run test in LS14. In contrast to this, column 1 and 2 of Table 12, shows that non-random sequences pass the run test and test for the longest run of ones in a block in the NIST test suite. On the other hand, in the NIST test suite non-random sequences fail some other tests such as approximate entropy and serial test.
- Tables 10, 11 show that non-random sequences fail from at least one of the test in LS14. On the other hand, according to column 4 of 13, only 3 Non-overlapping Template Matching tests out of 147 tests fail, column 5 shows that only 2 Non-overlapping Template Matching test out of 147 test fail, and column 6 shows that none of the test in NIST test suite fail, in other words non-random sequence passes the NIST test suite.

  As a result, the proposed lightweight test suite, LS14, can be applied to test the randomness of a long sequence. However, as an alternative, the reliability of random number generators can be tested better by increasing the number of tests.

5. Conclusion

We propose a new test suite to evaluate long sequences using three different methods. As for the first method, that is for testing the entire sequence at once, without partitioning, weight and run tests are used from the literature. Also, the entire R2 run test is defined to test the entire sequence without partitioning. As for the second method, to test a collection of fixed-length sub-sequences obtained by partitioning a long sequence, weight, run, linear complexity, and integer coverage tests are used. Finally, the third method that is for testing dynamic length sub-sequences, weight, collision, linear complexity, and index coincidence tests are used from the literature. Moreover, run and integer saturation tests are proposed in the dynamic partitioning method to complete the test suite. Afterwards, random sources are tested with the introduced test suite. Then, non-random biased and modified sequences are tested to find the sensitivities of each test. The Pearson correlation is employed to show the correlation between the methods and tests in the test suite. Finally, the results of NIST and the proposed LS14 test suite are compared. It is observed that the tests we proposed in this paper give better results. Since the main motivation of this paper is to propose a test suite enriched with a new methods, for long sequences we only give results with selected parameters that test results in each method will be compatible with other methods. For some other parameters, test result can be better.

Table 13. Results of NIST test suite

	1	2	3	4	5	6	7	8	9	10	11
Approximate Entropy	0.673321	8.843267	0	0.995689	0.961303	0.515667	0.798712	0.926638	0.919633	0.66378	0
Frequency Test within a Block	0.317426	0.349606	0.063889	0.065913	0.063889	0.186986	0.247547	0.834348	0.829309	0.823249	0
Cumulative SUM 1	0.0834	0.966648	0.344156	0.556993	0.688386	0.736192	0.608994	0.250297	0.781625	0	0
Cumulative SUM 2	0.09682	0.96187	0.449681	0.762346	0.688386	0.736192	0.793969	0.312923	0.880578	0	0
Frequency	0.084889	0.721092	0.390656	0.827975	1	1	0.51732	0.22917	0.624488	0	0
Linear Complexity	0.086092	0.523025	0.038915	0.07041	0.189268	0.477112	0.257603	0.535236	0.511801	0.099133	0.621461
Test for the Longest Run of Ones in a Block	0.905001	0.939874	0.007627	0.094659	0.044635	0.15134	0.3816	0.83335	0.79915	0.613254	0
Non-Overlapping Template Matching	145/147	147/147	134/147	144/147	145/147	147/147	147/147	147/147	147/147	147/147	35/147
Overlapping Template Matching	0.077454	0.494517	0.260901	0.426579	0.289271	0.102043	0.172027	0.559957	0.548902	0.294963	0
Random Excursion V(-1)	0.838724	0.955627	0.838724	0.838724	0.164205	0.473296	0.698551	0.303803	0.238035	NA	NA
Random Excursion V(-2)	0.066521	0.0719915	0.66521	0.6521	0.238889	0.580376	0.421132	0.353353	0.632885	NA	NA
Random Excursion V(-3)	0.872502	0.961855	0.872502	0.872502	0.42613	0.850536	0.955509	0.972404	0.749159	NA	NA
Random Excursion V(-4)	0.076942	0.41475	0.076942	0.076942	0.088969	0.081184	0.259401	0.94828	0.326092	NA	NA
Random Excursion V(1)	0.4843	0.164315	0.4843	0.483	0.077178	0.327657	0.735338	0.689247	0.536326	NA	NA
Random Excursion V(2)	0.287652	0.680462	0.287652	0.287652	0.784358	0.138689	0.648486	0.822329	0.684887	NA	NA
Random Excursion V(3)	0.150719	0.356663	0.150719	0.150719	0.318133	0.792802	0.839016	0.209209	0.861968	NA	NA
Random Excursion V(4)	0.093088	0.228798	0.093088	0.093088	0.923043	0.27491	0.337083	0.487469	0.855487	NA	NA
Random Excursion Variant	18/18	18/18	18/18	18/18	18/18	18/18	18/18	18/18	18/18	NA	NA
Run	0.743863	0.296471	0.870455	0.982395	0.870806	0.592456	0.535554	0.348173	0.317098	NA	NA
Binary Matrix Rank	0.92969	0.566379	0.512248	0.287817	0.274832	0.782794	0.898147	0.838937	0.63319	0.550551	0.680892
Serial1	0.734071	0.438843	0	0.615485	0.589229	0.196616	0.572886	0.83616	0.841721	0.760733	0
Serial2	0.971832	0.539644	0	0.512372	0.648968	0.418589	0.399348	0.842891	0.827452	0.864801	0.634761
Universal	0.765338	0.156516	0.228463	0.225479	0.226388	0.588435	0.479789	0.963884	0.991721	0.928601	0.001982

References

[1] A.A. Abd EL-Latif, B. Abd-El-Atty and S.E. Venegas-Andraca, *Controlled alternate quantum walk-based pseudo-random number generator and its application to quantum color image encryption*, Phys. A: Stat. Mech. **547**, 2020.

[2] Z. Akcengiz, M. Aslan, Ö. Karabayır, A. Doğanaksoy, M. Uğuz and F. Sulak, *Statistical randomness tests of long sequences by dynamic partitioning*, 2020 International Conference on Information Security and Cryptology, Ankara, Turkey, 2020.

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

[4] M.B. Demirköz, B. Kocaoğlu, Glitter Lamps: A Cryptographically- Secure, Physical, Non-Deterministic Random Bit Generator, In Preparation.

[5] A. Doğanaksoy and F. Göloğlu, On lempel-ziv complexity of sequences, Sequences and Their Applications-SETA, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, **4086**, 180-189, 2006.

[6] A. Doğanaksoy, F. Sulak, M. Uğuz, O. Şeker and Z. Akcengiz, *Mutual correlation of NIST statistical randomness tests and comparison of their sensitivities on transformed sequences*, Turk. J. Electr. Eng. **25**, 655-665, 2017.

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

[8] C. Fan, Q. Ding and C.K. Tse, *Evaluating the randomness of chaotic binary sequences via a novel period detection algorithm*, Int J Bifurcat Chaos **32** (5), 2022.

[9] C. Georgescu, E. Simion, A. Petrescu-Nita and A. Toma, *A view on nist randomness tests (in)dependence*, 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), 14, 2017.

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

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

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

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

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

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

[16] J. Hernandez-Castro, J. Sierra and A. Seznec, *The sac test: A new randomness test, with some applications to prng analysis*, Computational Science and Its Applications (ICCSA), International Conference, Assisi, Italy, 2004.

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

[18] M.G. Kendall and B.B. Smith, *Randomness and random sampling numbers*, J. R. Stat. Soc. **101** (1), 147-166, 1938.

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

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

[21] P. LEcuyer, *Testing random number generators*, Theory Probab. its Appl. **35**, 305-313, 1992.

[22] P. LEcuyer and R. Simard, *Testu01: A C library for empirical testing of random number generators*, ACM Trans Math Softw **33** (4), 1-40, 2007.

[23] H. Li, Y. Liu, M. Su and G. Wang, *Jump and hop randomness tests for binary sequences*, Cryptogr Commun **14**, 483-502, 2022.

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

[25] G. Marsaglia and A. Zaman, *Monkey tests for random number generators*, Comput. Math. with Appl. **26**, 1-10, 1993.

[26] U.M. Maurer, *A universal statistical test for random bit generators*, J. Cryptol. **5**, 89-105, 1992.

[27] K. Pearson, *Notes on regression and inheritance in the case of two parents*, Proc. R. Soc. Lond. **58**, 1895.

[28] A. Rukhin, *Testing randomness: A suite of statistical procedures*, Theory Probab. its Appl. **45**, 2000.

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

[30] B. Ryabko, V. Stognienko and Y. Shokin, *A new test for randomness and its application to some cryptographic problems*, J. Stat. Plan. Inference **123**, 365-376, 2004.

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

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

[33] A. Srinivasan, M. Mascagni and D. Ceperley, *Testing parallel random number generators*, Parallel Comput. **29**, 69-94, 2003.

[34] F. Sulak, *New statistical randomness tests: 4-bit template matching tests*, Turk. J. Math. **41**, 80-95, 2017.

[35] F. Sulak, *A new statistical randomness test: Saturation point test*, Int. J. Inf. Secur. **2**, 81-85, 2013.

[36] F. Sulak, *Statistical analysis of block ciphers and hash functions*, METU, 2012.

[37] F. Sulak, M. Uğuz, O. Koçak and A. Doğanaksoy, *On the independence of statistical randomness tests included in the nist test suite*, Turk. J. Electr. Eng. **25**, 3673-3683, 2017.

[38] F. Sulak, A. Doğanaksoy, M. Uğuz, O. Kocak, *Periodic template tests: A family of statistical randomness tests for a collection of binary sequences*, Discret. Appl. Math. **271**, 191-204, 2019.

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

[40] M. Turan, A. Doğanaksoy and S. Boztaş, *On independence and sensitivity of statistical randomness tests*, Sequences and Their Applications-SETA, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, volume 5203, 1829, 2008.

[41] A.V. Tutueva, E.G. Nepomuceno, A.I. Karimov, V.S. Andreev and D.N. Butusov, *Adaptive chaotic maps and their application to pseudo-random numbers generation*, Chaos Solitons Fractals **133**, 2020.

[42] M. Uğuz, A.Doğanaksoy, F. Sulak, and O. Koçak, *R-2 composition tests: a family of statistical randomness tests for a collection of binary sequences*, Cryptogr Commun **11**, 921-949, 2019.

[43] J. Walker, *Ent. a pseudorandom number sequence test program*, Software anddocumentation, 2008.

[44] C. Zhu, S. Li and Q. Lu, *Pseudo-random number sequence generator based on chaoticlogistic-tent system*, 2019 IEEE 2nd International Conference on Automation, Electronics and Electrical Engineering (AUTEEE), Shenyang, China, 2019.

[45] https:www.idquantique.com/random-number-generation/products/quantis-random-number-generator/
[46] https://users.metu.edu.tr/muhid/netcoreapp3.1.rar