Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2017) 25: 3673 – 3683

doi:10.3906/elk-1605-212

On the independence of statistical randomness tests included in the NIST test suite

Fatih SULAK[1,*], Muhiddin UĞUZ[2], Onur KOÇAK[3,4], Ali DOĞANAKSOY[2]

[1]Department of Mathematics, Faculty of Arts and Sciences, Atılım University, Ankara, Turkey
[2]Department of Mathematics, Faculty of Arts and Sciences, Middle East Technical University, Ankara, Turkey
[3]TÜBİTAK National Research Institute of Electronics and Cryptology (UEKAE), Gebze, Turkey
[4]Department of Cryptography, Institute of Applied Mathematics, Middle East Technical University, Ankara, Turkey

Received: 20.05.2016 • **Accepted/Published Online:** 31.05.2017 • **Final Version:** 05.10.2017

Abstract: Random numbers and random sequences are used to produce vital parts of cryptographic algorithms such as encryption keys and therefore the generation and evaluation of random sequences in terms of randomness are vital. Test suites consisting of a number of statistical randomness tests are used to detect the nonrandom characteristics of the sequences. Construction of a test suite is not an easy task. On one hand, the coverage of a suite should be wide; that is, it should compare the sequence under consideration from many different points of view with true random sequences. On the other hand, an overpopulated suite is expensive in terms of running time and computing power. Unfortunately, this trade-off is not addressed in detail in most of the suites in use. An efficient suite should avoid use of similar tests, while still containing sufficiently many. A single statistical test gives a measure for the randomness of the data. A collection of tests in a suite give a collection of measures. Obtaining a single value from this collection of measures is a difficult task and so far there is no conventional or strongly recommended method for this purpose. This work focuses on the evaluation of the randomness of data to give a unified result that considers all statistical information obtained from different tests in the suite. A natural starting point of research in this direction is to investigate correlations between test results and to study the independences of each from others. It is started with the concept of independence. As it is complicated enough to work even with one test function, theoretical investigation of dependence between many of them in terms of conditional probabilities is a much more difficult task. With this motivation, in this work it is tried to get some experimental results that may lead to theoretical results in future works. As experimental results may reflect properties of the data set under consideration, work is done on various types of large data sets hoping to get results that give clues about the theoretical results. For a collection of statistical randomness tests, the tests in the NIST test suite are considered. Tests in the NIST suite that can be applied to sequences shorter than 38,912 bits are analyzed. Based on the correlation of the tests at extreme values, the dependencies of the tests are found. Depending on the coverage of a test suite, a new concept, the coverage efficiency of a test suite, is defined, and using this concept, the most efficient, the least efficient, and the optimal subsuites of the NIST suite are determined. Moreover, the marginal benefit of each test, which also helps one to understand the contribution of each individual test to the coverage efficiency of the NIST suite, is found. Furthermore, an efficient subsuite that contains five statistical randomness tests is proposed.

Key words: Cryptography, random sequences, statistical randomness tests, NIST test suite, coverage, independence, correlation

*Correspondence: fatih.sulak@atilim.edu.tr

1. Introduction

Random numbers are used in many fields including cryptology, modeling simulations of a real phenomenon that is affected by an unpredictable process, or modeling a statistical event. In cryptography, they are mainly used in producing many cryptographic parameters such as encryption keys, challenges in protocols, salts and initialization vectors in cryptographic algorithms, and the like. Random numbers can be produced either by true random number generators (TRNGs) or by pseudorandom number generators (PRNGs). TRNGs use physical sources that are expected to be random such as atmospheric noise, radioactive decay, and the like. PRNGs are number generators that use relatively short random seeds to produce long pseudorandom sequences using a deterministic algorithm.

A sequence is called pseudorandom if it satisfies a set of predetermined properties that are observed from true random sequences. Consequently, the problem of evaluating a sequence in terms of randomness arises; that is, one has to be able to decide if the sequence really looks random or not. The set of properties through which pseudorandomness is defined is called a test suite. A test suite consists of several statistical randomness tests, each of which examines certain characteristics of the sequence and produces a P-value between 0 and 1. A P-value is the probability under a specified statistical model that a statistical summary of the data would be equal to or more extreme than its observed value [1]. P-values can indicate how the data diverge from a specified statistical model. It is also important to note that, by definition, P-values are uniformly distributed.

In cryptographic applications, a test suite can be used to discard certain sequences that are used in vital applications, such as encryption keys. Construction of a test suite is not an easy task. On one hand, the coverage of a suite should be wide; that is, it should compare the sequence under consideration from many different points of view with true random sequences. On the other hand, an overpopulated suite is expensive in terms of running time and computing power. Unfortunately, this trade-off is not addressed in detail in the most of the suites in use. Hence, an efficient suite should avoid use of similar tests, while still containing sufficiently many.

There are many statistical test suites and among these one of the mostly considered and used was developed by NIST [2]. It consists of 15 statistical randomness tests and has been used in many applications. For example, in the evaluation of advanced encryption standard (AES) candidate algorithms, one of the criteria was their performance as PRNG. The NIST test suite was used in order to determine the number of rounds in which the algorithms behave like a PRNG, to understand the security level of the algorithms roughly [3].

It is natural to expect the tests in a test suite to be independent and the P-values produced by these tests to be uncorrelated. There are various studies on the independence of tests and correlations between tests. In 2008, Turan et al. focused on five different tests and determined the dependencies and the correlations between the tests by defining transformations and analyzing the effects of these transformations on the P-values [4]. Doğanaksoy et al. extend the idea in [4] to the tests in the NIST test suite [5]. Moreover, they examined the relations between the tests by observing the distribution of the P-values that are less than a certain bound. Fan et al. gave a general method to evaluate the correlations between the tests in the NIST suite [6]. Hellekalek et al. showed that three tests in the NIST suite are highly correlated using defective sources [7]. Recently Doğanaksoy et. al. found the correlations between the tests in the NIST suite using the Pearson correlation test. They observed that the sets formed by "frequency test, cumulative sums test, serial test, and approximate entropy test" and the sets formed by "random excursion test and random excursion variant test" are correlated. Moreover, they discovered some transformation methods to which some tests react in similar manner.

In this work, the tests used in the NIST suite are investigated. Rather than the correlation values of

the entire set, the correlation of the tests at extreme values are concentrated on to decide the dependencies of the tests. Depending on the coverage of a test suite, a new concept, the coverage efficiency of a test suite, is defined, and using this concept the most efficient, the least efficient, and the optimal subsuites of the NIST suite, involving k of the tests in the package for each $k < 9$, are determined. Moreover, the marginal benefit of each test, which also helps one to understand the effects of each individual test on the overall coverage efficiency of the NIST suite, is found. Furthermore, considering all methods, an efficient subsuite for each k is proposed.

The organization of the paper is as follows. In Section 2, the preliminaries and the motivation of the paper are given. In Section 3, coverage of a test suite and coverage efficiency of a test suite are defined, which are used to measure the dependencies between the statistical randomness tests. In Section 4, the details of the experiment and the parameters are given. Three different evaluation methods for independence of tests and a detailed analysis of the experiment are proposed in Section 5. Finally, the paper is concluded in Section 6.

2. Preliminaries and motivation

A statistical randomness test examines certain characteristics of a sequence and produces a P-value between 0 and 1. For cryptographic applications, usually a threshold value is determined and the sequences that get P-values of less than the threshold value, denoted by α, are considered as nonrandom. Assuming the P-values are uniformly distributed, for each statistical randomness test, the expected value of the percentage of the number of sequences that get P-values of less than the threshold value is $100 \cdot \alpha$.

Assume that a statistical randomness test SRT_1 examines a set consisting of sequences $s_1, s_2, ..., s_n$ and produces P-values $(SRT_1(s_1), SRT_1(s_2), ..., SRT_1(s_n))$. Similarly, assume that a statistical randomness test T_2 examines the same set and produces P-values $(SRT_2(s_1), SRT_2(s_2), ..., SRT_2(s_n))$. Most of the conventional methods consider the correlation of arrays $(SRT_1(s_1), SRT_1(s_2), ..., SRT_1(s_n))$ and $(SRT_2(s_1), SRT_2(s_2), ..., SRT_2(s_n))$, and decide whether SRT_1 and SRT_2 are correlated or not.

The motivation to change this approach for cryptographic applications is explained by an example. Let SRT_1, SRT_2, and SRT_3 be three tests in the suite. To investigate if there are any correlations between tests SRT_1, SRT_2, and SRT_3 by conventional methods, first the matrix of P-values whose (i,j)th entry is the P-value of the sequence s_i from the test SRT_j; $1 \leq i \leq 10$, $1 \leq j \leq 3$, is constructed. An example of such a matrix is presented at Table 1.

Table 1. Matrix of P-values.

	SRT_1	SRT_2	SRT_3
s_1	0.1	0.1	0.9
s_2	0.2	0.2	0.3
s_3	0.3	0.3	0.8
s_4	0.4	0.4	0.2
s_5	0.5	0.5	0.6
s_6	0.6	0.001	0.1
s_7	0.7	0.7	0.3
s_8	0.8	0.8	0.2
s_9	0.001	0.1	0.007
s_{10}	0.002	0.1	0.00

Then, using one of the correlation functions, it is decided whether any pairs of tests are correlated or not. For example, using the Pearson correlation method, whose main concern is linear relations, the correlation

value between SRT_1 and SRT_2 is 0.745356, where the correlation value between SRT_1 and SRT_3 is –0.03558. This measure may advise to exclude one of SRT_1 or SRT_2 from the test suite.

An alternative approach is to consider the set of sequences that obtain P-values of less than a threshold value, rather than the complete set, when one considers relations for cryptographic applications such as choosing secure encryption keys. For this purpose a new matrix, the pass-fail matrix, of 0s and 1s is constructed instead of the matrix of P-values. In this matrix, the (i, j)th entry is 0 if the corresponding P-value is less than the threshold value $\alpha = 0.01$, and it is 1 otherwise. The corresponding pass-fail matrix of the matrix given in Table 1 is presented in Table 2.

Table 2. Pass-fail matrix.

	T_1	T_2	T_3
s_1	1	1	1
s_2	1	1	1
s_3	1	1	1
s_4	1	1	1
s_5	1	1	1
s_6	1	0	1
s_7	1	1	1
s_8	1	1	1
s_9	0	1	0
s_{10}	0	1	0

Notice that the values in the first and third columns of the pass-fail matrix are all the same. Since for cryptographic applications, the values below α are very important, one can conclude that tests SRT_1 and SRT_3 are highly correlated, and thus not independent, whereas the relation between SRT_1 and SRT_2 could not be seen. In this work, new methods to determine the dependencies of the tests in a suite are defined.

3. Coverage and coverage efficiency of a test suite

In light of the observations made in Section 2, the dependencies between two tests are defined in terms of coverage at extremity. To define coverage, first recall that, if the significance level α is fixed as 0.01, one says that this sequence "fails" a test if the P-value obtained from that test is less than 0.01. Coverage of a test suite is defined using this concept.

Definition 1 *Coverage of a test suite is defined as the ratio of the number of all sequences that fail at least one of the tests in the suite to the total number of sequences.*

When α is fixed to 0.01, 1% of the random sequences are expected to fail each test, assuming the P-values are uniformly distributed. Hence, if two tests are independent, by the product rule, 0.01% of the random sequences will fail both. By the inclusion-exclusion principle [8], one has the following theorem:

Theorem 1 *Let α be the significance level of k independent statistical randomness tests. Then the expected coverage of this collection is:*

$$\sum_{i=1}^{k} (-1)^{i+1} \left(\frac{k}{i}\right) \alpha^i = 1 - (1-\alpha)^k$$

Using this theorem, the expected coverage values of k tests for k ∈ {1,2...,9} are given in Table 3 for α = 0.01. Now a measure is given to indicate how much the coverage of a suite under consideration deviates from the expected coverage of a suite consisting of k independent statistical randomness tests.

Table 3. Expected coverage ratios for $\alpha = 0.01$.

Number of tests	Coverage
1	0.010000
2	0.019900
3	0.029701
4	0.039404
5	0.049010
6	0.058520
7	0.067935
8	0.077255
9	0.086483

Definition 2 *Coverage efficiency of a test suite with k statistical randomness tests is the ratio of the coverage of the suite to the theoretical coverage.*

It is important to include independent tests while forming a test suite. If a test suite contains tests that are dependent, the coverage efficiency of the suite will be lower than it should be. Therefore, the coverage efficiency is an important measure for test suites. Usually the coverage efficiency is less than 1 and smaller values of coverage efficiency indicate stronger dependencies between the tests. Note that in certain cases the coverage efficiency can be larger than 1. This can happen for two different reasons. First, the intersection of the coverage sets of the tests may be less than expected, and in this case the tests are also dependent by definition. These kinds of dependencies are not of concern in this work as they can be discarded for cryptographic applications. Second, for a specific test, the ratio of the sequences that obtain P-values of less than the threshold value 0.01 may be greater than 0.01. For example, this ratio is 0.0122 for the approximate entropy test. As a result, the coverage efficiency may be greater than 1 for the subsuites containing fewer numbers of tests.

Definition 3 *The marginal benefit of a test for a test suite is the difference in the coverage of the subsuite before and after the test is included.*

For instance, let subsuite $S_1 = \{SRT_1, SRT_2\}$ with $C(S_1) = x$ and $S_2 = \{SRT_1, SRT_2, SRT_3\}$ with $C(S_2) = y$, where SRT_1, SRT_2, and SRT_3 are randomness tests and $C(S_i)$ denotes the coverage of suite S_i. Then the marginal benefit of SRT_3 for S_1 is $y - x$. If SRT_3 is correlated with SRT_1 or SRT_2, the marginal benefit of SRT_3 for S_2 will be lower than expected.

4. Experimental setup

In order to measure the coverage efficiency of a subsuite, the tests are run on pseudorandom data and the experimental results are compared with the expected ones. In spite of its simple definition, it is difficult to compute the coverage of the suite theoretically due to the complicated distribution functions of tests. A practical way is to work on a sufficiently large sample space. Since the probability that a random sequence fails

two different tests is 0.0001, the sample space should contain at least 100,000 sequences to be able to call it a sufficiently large sample space, so that the expected number of sequences that fail both tests is 10. Otherwise, the observed number of such sequences will be too small to give a clear idea about the dependencies of the tests.

Moreover, each test in the NIST test suite has a lower bound on the length of the sequence. These tests are not applicable on the sequences shorter than this bound, or, even if they are, the results are not reliable. One can classify these tests as "short sequence tests" and "long sequence tests" by the minimum required sequence length. The natures of the short sequence tests and long sequence tests are different and they should be examined separately. Therefore, in this work, it is chosen to examine the short sequence tests and in order to cover as many tests as possible "short" is defined as 38,912, which is the minimum required sequence length for the binary matrix rank test as the next lowest length is 387,840 for Maurer's universal test. The minimum required sequence length for each test included in the NIST test suite is stated in Table 4.

Table 4. Minimum sequence lengths for tests.

Test	Minimum sequence length (bits)
Frequency (monobit) test	100
Frequency test within a block	100
Runs test	100
Test for the longest run of ones in a block	128
Binary matrix rank test	38,912
Nonoverlapping template matching test	1,000,000
Overlapping template matching test	1,000,000
Maurer's "universal statistical" test	387,840
Lempel–Ziv compression test	1,000,000
Linear complexity test	1,000,000
Serial test	128
Approximate entropy test	128
Cumulative sums (Cusum) test	100
Random excursions test	1,000,000
Random excursions variant test	1,000,000

For these reasons, the tests that examine sequences longer than 100,000 bits are excluded and hence one is left with the frequency test, frequency test within a block, runs test, test for the longest run of ones in a block, binary matrix rank test, serial test (this test produces two P-values called serial-1 and serial-2), approximate entropy test, and cumulative sums test. A total of 100,000 distinct sequences of length 38,912 are generated and each sequence is tested with these nine statistical randomness tests.

The data used in each experiment are generated by concatenating the ciphertexts of the 128-bit plaintexts encrypted with fixed-key AES [9]. Since the output of the AES is 128 bits, one needs $100,000\frac{38,912}{128} = 30,400,000$ ciphertexts. The plaintexts can be selected in many ways; however, for the set of plaintexts used in distinct experiments to be disjoint, a 128-bit index plaintext $p0 = 0x00\ldots0$ is set for the experiment, incremented 30,400,000 times, and encrypted using a fixed key to get 30,400,000 distinct plaintexts. Then the ciphertexts are concatenated and divided into 100,000 sequences of length 38,912.

For the nine tests mentioned above, the parameters that are recommended in the NIST TEST SUITE for 38,912-bit sequences are used [2]. The block length in the frequency test within a block is chosen to be 2048. For the serial test and approximate entropy test, the lengths of the bit strings and the blocks are set to 8 bits.

The parameters in the other parameterized tests, the block length in the test for the longest run of ones in a block, and the matrix size in the binary matrix rank test are chosen automatically according to the sequence length in the implementation given by NIST, which are 10^4 and 32×32, respectively.

In order to determine the dependencies of tests, a 100,000 $\times$ 9 binary matrix M (pass-fail matrix) is constructed, whose (i,j)th entry is 0 if the P-value of the ith sequence obtained from the jth test is less than $\alpha = 0.01$; it is 1 otherwise. Let $S(SRT_1, ..., SRT_n)$ be the test suite consisting of tests $SRT_1, ..., SRT_n$ and $C(S)$ be the coverage of the test suite S. For each subset of k elements of the index set {2, 3,... , 9} , all subsuites containing k statistical randomness tests $SRT_{i1}, ...SRT_{ik}$ are considered. For each subsuite, the coverage $C(\{SRT_{i1}, ..., SRT_{ik}\})$ is computed using the matrix M. Afterwards, the coverage efficiency of the subsuite is evaluated using Table 3.

5. Evaluation methods

The pass-fail matrix is evaluated by three different methods. First, two tests are fixed and a set is formed consisting of all sequences that fail the first test. Then the ratio of the sequences in that set that also fail the second test is found. That is, let $F_{T_i} = \{\sigma \in S | T_i(\sigma) < 0,01\}$ where S is the sample space. Then,

$$Corr(T_i, T_j) = \frac{\left|F_{T_i} \cap F_{T_j}\right|}{|S|}$$

is the correlation between T_i and T_j.

After calculating the correlation values, a fail-fail ratio table is constructed to find the mutual dependencies of tests. For example, for the frequency test, there are 984 sequences that obtain P-values of less than 0.01, while this number is 958 for the frequency test within a block. If these two tests were independent, expectedly approximately 10 sequences would fail both of these two tests. However, 69 sequences fail both tests, which indicates a correlation between these two tests.

In the Freq column of Table 5, the value corresponding to Bl Freq row is $\frac{69}{958} \approx 0.072025$, and in the Bl Freq column of Table 5, the value corresponding to Freq row is $\frac{69}{984} \approx 0.070122$. Notice that this operation is not commutative and hence Table 5 is not symmetric. The abbreviations for the tests can be found in Table 6.

Table 5. Fail-fail ratio table.

	Freq	Bl Freq	Runs	Long Run	BMR	App Ent	Cu Sum	Ser1	Ser2
Freq		0.070122	0.00813	0.006098	0.010163	0.011179	0.009146	0.010163	0.011179
Bl Freq	0.072025		0.004175	0.011482	0.008351	0.012526	0.016701	0.009395	0.005219
Runs	0.007663	0.003831		0.017241	0.011494	0.025862	0.020115	0.023946	0.006705
Long Run	0.005848	0.010721	0.017544		0.009747	0.016569	0.035088	0.019493	0.008772
BMR	0.012034	0.009627	0.014440	0.012034		0.014440	0.009627	0.015644	0.012034
App Ent	0.009821	0.010714	0.024107	0.015179	0.010714		0.022321	0.306250	0.137500
Cu Sum	0.008964	0.015936	0.020916	0.035857	0.007968	0.024900		0.029880	0.004980
Ser1	0.009398	0.008459	0.023496	0.018797	0.012218	0.322368	0.028195		0.281955
Ser2	0.011168	0.005076	0.007107	0.009137	0.010152	0.156345	0.005076	0.304569	

When the results of the tests are analyzed carefully, it can be seen that sequences that fail some tests also fail specific tests with a higher probability than expected. This result indicates that the tests in question

Table 6. Abbreviations.

Abbreviation	Test
App Ent	Approximate entropy test
Bl Freq	Frequency test within a block
BMR	Binary matrix rank test
Cu Sum	Cumulative sums test
Freq	Frequency test
Long Run	Test for the longest run of ones in a block
Run	Runs test
Ser1	Serial 1 test
Ser2	Serial 2 test

are not dependent. Note that the expected value in the fail-fail ratio table is 0.01; thus, a threshold value of 0.05 is used and it is concluded that the pair of tests that get values greater than 0.05 in Table 5 are dependent. For example, 1064 sequences fail the serial-1 test and 343 of them also fail the approximate entropy test where the expected value is approximately 10, which indicates a strong dependency between these tests. As a result, the experiments suggest that the approximate entropy test, serial-1 test, and serial-2 test are dependent on each other. A similar dependency is observed between the frequency test and the frequency test within a block. Therefore, in order to build an efficient subsuite, one should avoid including dependent tests. These dependencies can be seen clearly in the Figure.

The second method suggests a procedure to find the most efficient, the least efficient, and the optimal subsuites consisting of k tests. All possible subsuites are considered and the experimental coverages of the subsuites are found, and the subsuites with the highest and the lowest coverage efficiency values for all subsuites including up to 8 tests are presented in Table 7 and Table 8, respectively. Let $F_{T_i}=\{\sigma\in S|T_i(\sigma)<0,01\}$ and T be a subsuite with k tests where S is the sample space. Then,

$$Cov(T)=\left|\bigcup_i F_{T_i}\right|$$

is the coverage of the subsuite T.

Table 7. Most efficient subsuites.

#	Subsuite	Coverage	Efficiency
2	Run, App Ent	0.02137	1.073
3	Run, App Ent, Long Run	0.03128	1.053
4	Run, App Ent, Long Run, Freq	0.04087	1.037
5	Run, App Ent, Long Run, Freq, Cu Sum	0.05003	1.021
6	Run, App Ent, Long Run, Freq, Cu Sum, Bl Freq	0.05853	1.000
7	Run, App Ent, Long Run, Freq, Cu Sum, Bl Freq, Ser2	0.06658	0.980
8	Run, App Ent, Long Run, Freq, Cu Sum, Bl Freq, Ser2, BMR	0.07425	0.961
9	All Tests	0.07896	0.913

Table 8. Least efficient subsuites.

#	Subsuite	Coverage	Efficiency
2	Ser1, Ser2	0.01749	0.879
3	Ser1, Ser2, App Ent	0.02481	0.835
4	Ser1, Ser2, App Ent, BMR	0.03283	0.833
5	Ser1, Ser2, App Ent, BMR, Bl Freq	0.04211	0.859
6	Ser1, Ser2, App Ent, BMR, Bl Freq, Freq	0.05097	0.871
7	Ser1, Ser2, App Ent, BMR, Bl Freq, Freq, Cu Sum	0.06024	0.887
8	Ser1, Ser2, App Ent, BMR, Bl Freq, Freq, Cu Sum, Long Run	0.06956	0.900
9	All Tests	0.07896	0.913

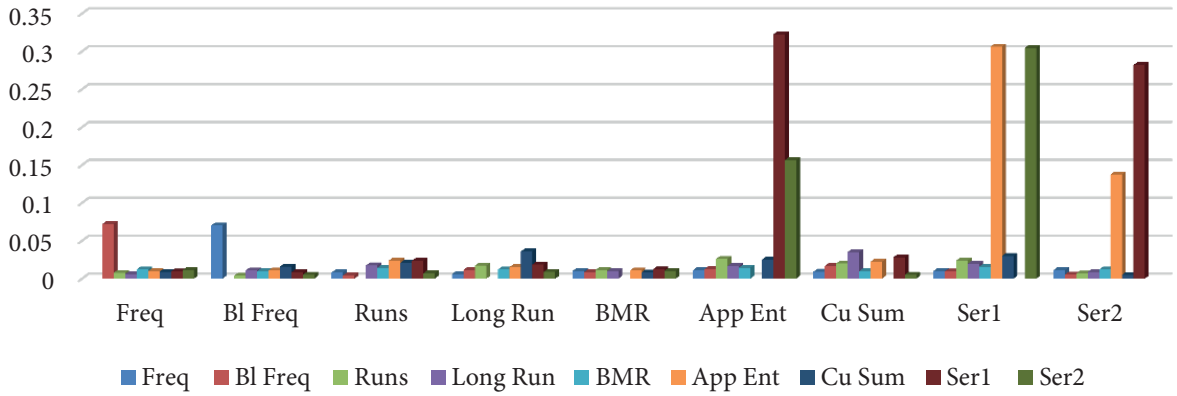

Figure. Fail-fail correlation graph.

Note that the experimental coverage of the test suite with all nine tests is 0.07896 and the coverage efficiency is 0.913. As mentioned in Section 3, the ratios of the sequences that obtain P-values of less than the threshold value of 0.01 are greater than 0.01 for some tests and the coverage efficiency may be greater than 1 for the subsuites containing smaller numbers of tests. For this reason, the optimal subsuites whose coverage is around 1 are also considered and optimal subsuites are presented in Table 9. Moreover, it is observed that there is an ordered structure of the tests for each mentioned subsuite. For example, the optimal subsuite with two tests consists of the frequency test within a block and the runs test. If one considers the optimal subsuite with three tests, the test for the longest run of ones in a block should be added to the previous subsuite and the ordered structure is valid for all possible cases.

Table 9. Optimal subsuites.

#	Sub-suite	Coverage	Efficiency
2	Bl Freq, Run	0.01998	1.004
3	Bl Freq, Run, Ser1	0.02971	1.000
4	Bl Freq, Run, Ser1, Long Run	0.03959	1.005
5	Bl Freq, Run, Ser1, Long Run, Cu Sum	0.04968	1.014
6	Bl Freq, Run, Ser1, Long Run, Cu Sum, App Ent	0.05780	0.988
7	Bl Freq, Run, Ser1, Long Run, Cu Sum, App Ent, Freq	0.06628	0.976
8	Bl Freq, Run, Ser1, Long Run, Cu Sum, App Ent, Freq, BMR	0.07425	0.961
9	All Tests	0.07896	0.913

In the third method, the marginal benefits of each test for the overall test suite are determined. Let $F_{T_i}=\{\sigma\in S|T_i\,(\sigma)<0,01\}$ and T be a subsuite with k tests where S is the sample space. Then,

$$Cov\,(T,T_i)=|Cov(T)\setminus Cov(T-T_i)|$$

is the marginal coverage of T_i within subsuite T. The results are presented at Table 10. It is seen that the runs test has the largest marginal benefit and the serial-2 test has the smallest marginal benefit.

Table 10. Marginal benefits of tests for the test suite containing 9 tests.

Test	Mar. ben.
Run	0.00940
Long Run	0.00915
Cu Sum	0.00876
Freq	0.00867
Bl Freq	0.00836
BMR	0.00761
App Ent	0.00670
Ser2	0.00611
Ser1	0.00471

The first method suggests that at most one test should be used from the set approximate entropy test, serial-1 test, and serial-2 test and from the set frequency test and frequency test within a block. Using the third method, it is seen that the marginal benefit of the approximate entropy test is larger than that of the serial-1 test and serial-2 test. Similarly, the marginal benefit of the frequency test is larger than that of the frequency test within a block. As a result, considering the optimal subsuites, it is proposed that it is more efficient to use the subsuite containing the five statistical randomness tests, the frequency test, runs test, test for the longest run of ones in a block, approximate entropy test, and cumulative sums test, instead of the overall suite that consists of the nine tests.

6. Conclusion

In this paper, the dependencies of nine statistical randomness tests included in the NIST test suite are examined. To have a manageable sample space, the tests in the suite that evaluate sequences longer than 100,000 bits are excluded. In order to determine the dependencies of tests, a pass-fail matrix is constructed and evaluated using three different approaches. First, all pairs of tests in the suite are considered separately. For each pair a set consisting of the sequences that fail one of the tests is formed. Then the ratio of the number of sequences in that set that also fail the other test is found. Afterwards, a fail-fail ratio table is constructed to find the mutual dependencies of tests. Experimental results suggest that the approximate entropy test, serial-1 test, and serial-2 test are dependent on each other. A similar dependency is observed between the frequency test and the frequency test within a block.

Second, a new concept, the coverage efficiency of a test suite, is defined depending on the coverage of a test suite. Using this concept, the most efficient, the least efficient, and the optimal subsuites of the NIST suite are determined. The results are presented in Tables 7–9. In all three tables, the ordered structure of the tests is observed. Furthermore, the marginal benefit of each individual test for the overall coverage efficiency of the NIST suite is found. It is observed that the runs test has the most marginal benefit and the serial-2 test has the least marginal benefit. Furthermore, an efficient subsuite is observed, which contains the five statistical

randomness tests, the frequency test, runs test, test for the longest run of ones in a block, approximate entropy test, and cumulative sums test.

As a future work, the dependencies between the tests that evaluate long sequences can be found. The idea in this paper can also be extended to find the coverage efficiency of other test suites.

References

[1] Wasserstein RL, Lazar NA. The ASA's statement on p-values: context, process and purpose. Am Stat 2016; 70: 129-133.

[2] Rukhin AL, Soto J, Nechvatal J, Smid M, Barker E, Leigh S, Levenson M, Vangel M, Banks D, Heckert A et al. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications Sp 800-22 Rev. 1a. Gaithersburg, MD, USA: Booz-Allen and Hamilton Inc., 2010.

[3] Soto J, Bassham L. Randomness Testing of the Advanced Encryption Standard Finalist Candidates. NIST IR 6483. Gaithersburg, MD, USA: National Institute of Standards and Technology, 1999.

[4] Turan MS, Doğanaksoy A, Boztaş S. On independence and sensitivity of statistical randomness tests. In: Sequences and Their Applications - SETA 2008: 5th International Conference; 14–18 September 2008; Lexington, KY, USA. Berlin, Germany: Springer. pp. 18-29.

[5] Doğanaksoy A, Ege B, Muş K. Extended results for independence and sensitivity of NIST randomness tests. In: Information Security and Cryptography Conference, ISC Turkey; 25–27 December 2008; Ankara, Turkey. pp. 190-194.

[6] Fan L, Chen H, Gao S. A general method to evaluate the correlation of randomness tests. In: Information Security Applications: 14th International Workshop, WISA Revised Selected Papers; 19–21 August 2013; Jeju Island, Korea. Berlin, Germany: Springer International Publishing. pp. 52-62.

[7] Hellekalek P, Wegenkittl S. Empirical evidence concerning AES. ACM T Model Comput S 2003; 13: 322-333.

[8] Cameron PJ. Combinatorics: Topics, Techniques, Algorithms. Cambridge, UK: Cambridge University Press, 1994.

[9] Daeman J, Rijmen V. The Design of Rijndael: AES - The Advanced Encryption Standard. Berlin, Germany: Springer-Verlag, 2002.