Simplification Algorithm and Analysis of Random Sequence Test Packets based on Filtering Ability

Zekun Wang¹² and Fuxi Zhang¹,*
¹College of Engineering Science and Technology, Shanghai Ocean University, Shanghai, 201306, China
²Department of Mechanical Engineering, Auburn University, Auburn, AL, 36949, U.S.
*Corresponding author email: fx-zhang@shou.edu.cn

Abstract. A method was presented for simplifying the random sequences by means of introducing the concept of “the ability of filtering”. On the basis of explication of the relationship between each test and In-depth analysis of the role and contribution of individual tests to the overall test, the algorithm and implementation plan of simplified test package were proposed. Three procedures for filtering descending permutation tables for simplifying links and the algorithm for simulation results were discussed.

1. Introduction
Random numbers play a vital role in cryptography systems which are widely used in key generation, digital signature, authentication protocol, timestamp, initialization vector, large prime number generation and so on [1]. A flawed sequence of random numbers could lead to the entire password system being broken by an attacker. Randomity testing is an important technical means to study the security of cryptographic algorithms by testing whether the output sequence of a cryptographic algorithm is random enough to determine its security. The requirement for random numbers should not only stop at the fact which the sequence is statistically randomly generated, but that the sequence of random numbers is unpredictable [2].

A random number generator is an algorithm, device, or function that produces random numbers, and its security is very important to the security of a cryptographic system. Commonly used random number generators include squared method, linear iso-residual method, chaos mapping method, DSA, and Intel generator. Current computers cannot generate a true sequence of random numbers, but only through simulations of some physical processes, such as quantum effect, thermal motion (various noise, Brown motion), isotope decay, potential penetration, time coordinates, and noise signals; pseudo-random number sequence generated by Monte Carlo, von Neumann, U (0, 1) random number, etc. However, both approaches have their own flaws: the physical processes that the former may seem random but actually imply a lot of random patterns, while the latter actually uses an unchanging recursive algorithm to generate random numbers, and the resulting sequences are periodic in nature [3]. Therefore, it is very important to test whether the random number sequence meets certain random conditions or detects the defects of the sequence by algorithm. Randomness is a probability attribute. Comparing a sequence with a truly random sequence determines whether it is random by examining the probability size of the sequence in a true random process. There are infinite numbers of possible tests that cannot be tested with all existing tests in order to detect the randomness of a test. It is reasonable to ensure that the tests included are as comprehensive as possible and that there are as few repetitions as possible. A reliable random test package must
ensure that the tests it contains are independent and unrelated to each other, and that each test must meet a certain level of coverage. At the same time, another basis for a test whether it should be included in the test package is the necessity of the existence of the test, or the ability to consider a sequence of obsolescations that distinguish it from other tests [4-6]. Considering the operational efficiency of the test package, it is perfectly possible to exclude tests in a test package that are overwritten by one or more other tests in the package [7]. This paper dedicates to effectively simplify the tests by introducing "screening capabilities".

2. Hypothetical Test
The basic idea of hypothetical test is the principle of small probability, that is, if a hypothesis is believed to be true, then an event with a very small probability of occurring under this assumption is assumed to be impossible in an experiment [8]. Based on this strategy, two assumptions are first made based on the actual problem, one called the original hypothesis or the zero hypothesis, recorded as H0, and the other is called the alternative hypothesis, which is recorded as H1. A statistic is then constructed, which is H0 when the data is true. Second, according to the known distribution, give the value of the significant level $\alpha$. According to H1, set $P\{\text{decline } H_0|\text{true } H_0\} = \alpha$, to determine the deny field for the statistic. The value of the statistic is finally calculated, and if the value falls into the deny field, then H0 rejected, otherwise H0 accepted. To determine the deny domain involves two types of errors in the hypothesis test: type 1 error rejects H0 assuming true results, calling an "abandoned" error; type 2 error assumes that H0 was accepted when it is false, calling a "false" error. The probability of introducing type 1 errors refers to a level of significantness $\alpha$ and the probability of creating type 2 errors refers to $\beta$. When the sample size is fixed, the probability of making another type of error tends to increase if the probability of creating one type of errors is reduced. Usually we select the sample size and control the value of $\alpha$ to determine the appropriate rejection domain so that $\beta$ is taken to a minimum. This test controls only the probability of making type 1 errors, regardless of the probability of making type 2 errors which is also named as the “significantity test”. A probability of B occurring under conditions where A has occurred, as defined below:

Assume A and B are two events, and $P(A) > 0$,

$$ P(B|A) = \frac{P(AB)}{P(A)}, \quad (1) $$

is a probability of B occurring under conditions where A has occurred.

Independence:

$$ P(AB) = P(A)P(B) \quad (2) $$

or

$$ P(B|A) = P(B) \quad (3) $$

Entropy is an important concept in Informatics theory, which is used to illustrate the amount of information that a random variable can provide by taking values. The possible value of the discrete random variable X is limited: $x_1, x_2, x_3 ... x_n$. The probability of taking each value is different: $p_i, 1 \leq i \leq n$.

The entropy of the random variable X is:

$$ H(X) = -\sum_{i=1}^{n} p_i \log p_i. \quad (4) $$

When $p_i = 0$, $p_i \log p_i = 0$.

The condition entropy of X under Y is:

$$ H(X|Y) = \sum_{y} P(Y = y)H(X|Y = y) \quad (5) $$

The entropy of mutual information between X and Y is:

$$ I(X \cdot Y) = H(X) - H(X|Y) \quad (6) $$

Consequently, the entropy of mutual information meets the following properties:

$$ I(X \cdot Y) = I(Y \cdot X) $$
3. "Simplified Test Package" Algorithm Based on Filtering Capabilities

3.1. Filtering Ability

As the relevant knowledge system in the field of random testing has not been established, we can't rule out unnecessary testing by means of mathematical theory [9, 10]. This paper proposes the "simplified test package" algorithm based on screening ability from the practical, that quantify the contribution of a test to the overall test package and then by comparing the contribution values of each test type to avoid unnecessary testing by determining whether to reject.

Filtering ability: For one test \( i \) in a test package machine that contains \( n \) tests, define the filtering capability of test \( i \):

\[
C_l(i) = \frac{H(X_i|X_1, X_2, ..., X_{i-1}, X_{i+1}, ..., X_n)}{H(X_i)}
\]  

According to Eq. (5), Eq. (7) can be written as:

\[
C_l(i) = \frac{H(X_1, X_2, ..., X_n) - H(X_1, X_2, ..., X_{i-1}, X_{i+1}, ..., X_n)}{H(X_i)}
\]

The higher the filtering capability, the more important this test \( i \) will be for the entire test package. In addition, because modern computers can't test all sequences with a sequence length of more than 25 bits in a short period of time, all the data in this paper is based on a sequence operation comparison with a sequence length of less than 25. It also expounds some laws found by observation and the reasonable inference of the effect of longer sequence test resulting from these laws.

3.2. Implementation

Randomness detection usually investigates whether the detected sequence meets certain characteristics of the random sequence by means of probability statistics to determine whether it is random. Statistical detection of random numbers is a commonly used method of random number detection. Each test item is divided into first-level detection and second-level detection. Some existing secondary tests are incomplete, i.e. random sequences through which they are used may still be flawed in the statistical characteristics detected. The results calculated by the detection of normally distributed Pvalue suitable for first-level detection. In secondary detection, even if the Pvalue evenly distributed, the sequence under test is detected, and the sequence does not necessarily satisfy the nature of the test. A significant imbalance in the appearance times of "0" and "1" in the sequence is the purpose of the test. When the number of 0s and 1s in a sequence is out of bounds, the test treats them as "non-random".

For the purpose of improving the efficiency of operation, scholars proposed various methods, among which the more common is to change the operation order of the test package internal test. The core idea of changing the order of operation is to have more sequences that are judged "non-random" by the test package to be rejected by the test package as early as possible. For example, if a test in a test package can exclude 90% of the number of sequences that the test package can exclude, putting the test in the first place in the order in which the test package is executed can greatly improve the speed of the entire test package. To get the fastest run sequence for a test package, we provide the following effective execution order:

1. The length of the selected sequence \( n \);
2. Traverses all test strings of \( n \) lengths for all tests in the test package. Select the test string with the largest number of exclusion sequences and place it first in the execution order. Organize the sequence collection \( K \), which contains all sequences with \( n \) lengths and are considered "random" by this test;
3. Transvers \( K \) with all the rest sequences. Set the largest ordered sequence to the standing by position, and update the contents of sequence collection \( K \) to all sequences that are considered "random" by all tests in the current test sequence;
4. Repeat step 3 until all tests in the test package have been placed in the order in which the test package was executed;
(5) Get the order executive order of the test package. This method is most effective because the number of test exclusions per step must be the largest of the tests that have not yet been performed. It ensures the best execution sequence with a length of n.

3.3. Test Methods Based on the Fastest Execution Order

The basic idea is to use the generation characteristics of random numbers to exclude the values that have been generated from random intervals, so as to ensure that the next random number generated must be new. Specifically, first, create an array of length N, with initial values 0... N-1; then a random number x1 is generated, \( x1 \in [0,N) \). Take the \( \text{num1}=\text{array}[x1] \) as the first member in the sequence. Next is the key step: swap \( \text{num1} \) with array \( [N-1] \); then generate a random number \( x2=\text{random.Next}(0,N-1) \), \( x2 \in [0,N-1) \). Since \( \text{num1} \) has been exchanged with array \( (N-1) \) and \( x2<N-1 \). So \( \text{num2}=\text{array}[x2] \) must not be equal to \( \text{num1} \) to avoid duplication. Then swap \( \text{num2} \) and array\( [N-2] \). According to this method, you can get the third and fourth in the sequence... Nth member. The resulting array is a non-repeating random sequence. Here's a graphical demonstration of the entire calculation process (assuming N=5):

![Diagram of the method based on the fastest execution order.]

The cursor starts from the very end.

2 is randomly chosen and exchanged with 4.

The cursor moves forward the 4th position.

0 is randomly chosen and exchanged with 3.

The cursor moves forward to the 3rd position.

1 is randomly chosen and exchanged with 4.

The cursor moves forward to the 2nd position.

4 is randomly chosen and exchanged with itself.

The cursor moves forward to the 1st position.

**Figure 1.** The method based on the fastest execution order.
On the basis of the above, we propose a test method based on the fastest execution sequence.
(1) Set the sequence length of n;
(2) Select the number of tests in the current test package m to generate all possible variables [x1, x2, ..., xm]. [1,0... 0] Represents the first number of the sequence is identified as "random", but the rest are identified as "not random", and so on;
(3) Traverse all sequences with a length of n and assign values to all variables obtained by step 2;
(4) Remove variables [1,1,...1], and arrange all other 2m−1 variables in a descending order;
(5) Select a sequence set that excludes the current at the top of the descending sequence table and add it to the execution sequence of the test package. If an even number of tests meet this condition, continue to the next item on the watch list until the last one remains.
(6) Repeat step (5) until all tests have been added to the execution list.

4. Results and Discussion
By performing the appropriate list generator (excluding tests 3, 4, 5, 6) on the test package, we find this fact: The resulting list of two execution orders, although small differences in detail, is generally the same trend; At the same time, the resulting execution order list is strikingly similar to the filtering capability descending sort table described in this article. Tests numbered 7, 8, 9, 15, 16 are usually prioritized, while tests numbered 1,2,12,13,14 are usually ranked in the second half of the order list. Tests numbered 1, in particular, have always been ranked at the bottom of the list. Its over-universal and concise nature has led to the ability of the test itself to be overshadowed by many other tests. The proposed method performs a fast and efficient filter capability of the descending list [11-13]. A test method based on operational efficiency better reflects the contribution of individual tests to the test package itself than a test method based on the fastest execution order. Once we have obtained the filtering capability descending list, there is a certain ability to modify the test package.

5. Conclusions
This paper mainly introduces the concept of "filtering ability". From a practical point of view, try to analyze the role and contribution of each test in the sequence randomness test package to the overall test package. This is used to explain the relationship between the various tests that cover each other, and to propose the behavior of the "simplified test package" and the specific implementation plan.

Acknowledgments
The authors would like to express their gratitude for the support of Shanghai Engineering Research Center of Marine Renewable Energy (Grant No. 19DZ2254800).

References
[1] D L. Banks. Statistical Testing of Random Number Generators, Proceedings of the 22nd National Information Systems Security Conference, October 19, 1999
[2] Md. Arshad Ali, Yuta Kodera, Takuya Kusaka, Satoshi Uehara, Yasuyuki Nogami, Robert H. Morelos-Zaragoza. Multi-Value Sequence Generated over Sub Extension Field and Its Properties, Journal of Information SecurityVol.10 No.03(2019)
[3] Abdul Rahim Khan. Random iterative algorithms and almost sure stability in Banach spaces, Filomat 31(12):3611-3626 • January 2017
[4] Zhou, Xiao, Shi, Ruizhi, Li, Da. FM Screening Algorithm of Multiplicative Congruence Pseudorandom based on Dot Gathering Model, NIP & Digital Fabrication Conference, 2014 International Conference on Digital Printing Technologies, pp. 100-103(4)
[5] Arredondo-Alonso S, Rogers MRC, Braat JC, Verschuuren TD, Top J, Corander J, Willems RJJL, Schürch AC. mlplasmids: a user-friendly tool to predict plasmid- and chromosome-derived sequences for single species, Microb Genom. 2018 Nov;4(11). doi: 10.1099/mgen.0.000224. Epub 2018 Nov 1.
[6] A Chambaz, MVD Laan. Estimation and testing in targeted goup sequential covariate-adjusted randomized clinical trials, University of California - Berkeley, 2013, 41(1):104-140
[7] Liu, Gang, Guo, Zhaolu. A clustering-based differential evolution with random-based sampling and Gaussian sampling, Neurocomputing 205 • May 2016

[8] Hiroaki Kikuchi, Xuping Huang, Shigeta Ikuji, Manami Inoue. Privacy-Preserving Hypothesis Testing for Reduced Cancer Risk on Daily Physical Activity, Journal of Medical Systems volume 42, Article number: 90, 2018

[9] Daner Chen. A Collaborative Filtering Recommendation Algorithm Based on User Information, Communication Systems and Information Technology, Part of the Lecture Notes in Electrical Engineering book series (LNEE, volume 100): 993-998

[10] Zhaochen Dong, Wei Yu, Wangli Xu. A modified combined p -value multiple test, Journal of Statistical Computation and Simulation, 85(12): 2479-2490, 2015

[11] Felix Miedaner, Ludwig Kuntz, Christian Enke, Bernhard Roth, Anika Nitzsche. Exploring the differential impact of individual and organizational factors on organizational commitment of physicians and nurses, BMC Health Services Research volume 18, Article number: 180 (2018)

[12] Zhang, Ruining, Jiang, Xuemei, Li, Ruifang. Improved decomposition-based multi-objective cuckoo search algorithm for spectrum allocation in cognitive vehicular network, Physical Communication, Volume 34, June 2019, Pages 301-309

[13] Yi Wen. Dynamic Adaptive Multi-cuckoo Search Algorithm, Journal of Computers • January 2018