DOI: 10.37943/18LYCW2723

Ardabek Khompysh
PhD, Information Security Laboratory
ardabek@mail.ru, orcid.org/0000-0002-0702-9346
Institute of Information and Computational Technologies, Kazakhstan
PhD, Associate Professor
Egyptian University of Islamic Culture Nur-Mubarak, Kazakhstan

Kunbolat Algazy*
PhD, Information Security Laboratory
kunbolat@mail.ru, orcid.org/0000-0003-3670-2170
Institute of Information and Computational Technologies, Kazakhstan

Nursulu Kapalova
leading researcher, Information Security Laboratory
nkapalova@mail.ru, orcid.org/0000-0001-9743-9981
Institute of Information and Computational Technologies, Kazakhstan

Kairat Sakan
PhD, Information Security Laboratory
kairat_sks@mail.ru, orcid.org/0000-0002-6812-6000
Institute of Information and Computational Technologies, Kazakhstan

Dilmukhanbet Dyusenbayev
Researcher, Information Security Laboratory
dimash_dds@mail.ru, orcid.org/0000-0002-4835-1075
Institute of Information and Computational Technologies, Kazakhstan

STATISTICAL PROPERTIES OF THE PSEUDORANDOM SEQUENCE GENERATION ALGORITHM

Abstract: One of the most important issues in the design of cryptographic algorithms is studying their cryptographic strength. Among the factors determining the reliability of cryptographic algorithms, a good pseudorandom sequence generator, which is used for key generation, holds particular significance. The main goal of this work is to verify the normal distribution of pseudorandom sequences obtained using the generation algorithm and demonstrate that there is no mutual statistical correlation between the values of the resulting sequence. If these requirements are met, we will consider such a generator reliable. This article describes the pseudorandom sequence generation algorithm and outlines the steps for each operation involved in this algorithm. To verify the properties of the pseudorandom sequence generated by the proposed algorithm, it was programmatically implemented in the Microsoft Visual C++ integrated development environment. To assess the statistical security of the pseudorandom sequence generation algorithm, 1000 files with a block length of 10000 bits and an initial data length of 256 bits were selected. Statistical analysis was conducted using tests by D. Knuth and NIST. As shown in the works of researchers, the pseudorandom sequence generation algorithm, verified by these tests, can be considered among the reliable algorithms. The results of each graphical test by D. Knuth are presented separately. The graphical tests were evaluated using values obtained from each test, while the chi-squared criterion with degrees of freedom $2^k - 1$ was used to analyze the evaluation tests. The success or failure of the test was determined using a program developed by the Information Security Laboratory. Analysis

Received: 22.05.2024 Accepted: 25.06.2024 Published: 30.06.2024

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 18, JUNE 2024

of the data from the D. Knuth tests showed good results. In the NIST tests, the P-value for the selected sequence was calculated, and corresponding evaluations were made. The output data obtained from the NIST tests also showed very good results. The proposed pseudorandom sequence generation algorithm allows generating and selecting a high-quality pseudorandom sequence of a specified length for use in the field of information security.

Keywords: cryptography; algorithms; random sequence; pseudorandom sequence; statistical testing.

Introduction (Literary review)

The information security system must address tasks such as protecting the confidentiality of information or its critical components, verifying the authenticity of subjects and objects involved in information transmission, safeguarding information from unauthorized access by non-authorized users, protecting the rights of users who own the information, managing information, and operationally controlling the processes of information transformation and transmission. To solve these problems, cryptographic algorithms are used, including information hashing methods. The binary sequence obtained as a result of the hashing process can also be considered a pseudorandom sequence (PRS) [1,2].

Random numerical sequences are used in various fields of scientific research, one of which is cryptography. In cryptography, random sequences play an important role as they are used, for example, to generate initial parameters for cryptographic algorithms and protocols, as well as to create high-quality keys in stream cipher algorithms [3]. The random uniform distribution of ciphertexts obtained using algorithms that utilize high-quality keys ensures the statistical security of the algorithm [4,5]. An algorithm that generates pseudorandom numbers or independent sequences using certain mathematical methods is called a pseudorandom number generator or sequence generator (PRNG).

In most encryption algorithms, especially in stream ciphers, key sequence generators are used. A key sequence generator outputs a stream of bits that appears random but is actually deterministic and can be precisely reproduced on the recipient's side [6,7]. The more the generated stream resembles a random sequence, the longer it will take for a cryptanalyst to break the cipher. However, if the generator produces the same sequence each time it is initialized, breaking the cryptosystem becomes a trivial task. For example, in the case of stream ciphers, if an attacker intercepts two encrypted texts, they can XOR them to obtain the two original texts XORed together. This makes the system very easy to break. If an attacker has a plaintext-ciphertext pair, the task becomes even simpler. Therefore, it is assumed that all random sequence generators must be key-dependent. This dependency ensures that simple cryptanalysis is impossible. The structure of a key sequence generator can be represented as a finite state machine with memory, consisting of three blocks: a memory block that stores the state information of the generator, an output function that generates a bit of the key sequence based on the state, and a transition function that determines the new state the generator will transition to in the next step [8].

Currently, there are several thousand different variants of pseudorandom number generators. Let's consider the main methods of generating pseudorandom sequences that are most suitable for computer cryptography.

Linear Congruential Generator (LCG). The main advantages of linear congruential generators are their high speed due to the small number of operations per byte and their simplicity of implementation. Unfortunately, these generators are rarely used in cryptography because they are predictable. Specifically, LCGs cannot be used for constructing stream ciphers, as their predictability makes them vulnerable. For example, stream ciphers based on LCGs were first broken by Joan Boyar, who also successfully broke quadratic generators [9].

DOI: 10.37943/18LYCW2723

Among the most promising types of PRNGs are those based on shift registers with nonlinear feedback, specifically using so-called stochastic summators or R-blocks. This work generalizes the results obtained from studying PRNGs based on linear feedback shift registers (LFSR) to the generation of PRNGs using stochastic summators in the feedback loop (RFSR, Random Feedback Shift Register). Specifically, it examines the principles of constructing nonlinear PRNGs of length $2^n - 1$ and 2^n, as well as universal generators that ensure any predefined period and pre-period values of the generated sequences, where n is the number of memory elements in the generator consisting of N registers, each with a word length of n ($Q = nN$). LFSRs are decent random number generators but have undesirable properties. The bit sequences they generate are linear, which makes them unsuitable for encryption. For an LFSR of length n, the internal state can be determined from n output bits of the generator. Even if the feedback scheme is unknown, it is sufficient to have $2n$ output bits to deduce it. Large random numbers generated from consecutive bits of an LFSR are highly correlated and sometimes not truly random. Nonetheless, LFSRs are quite often used as fundamental encryption algorithms [10].

There are several tests used to evaluate the statistical properties of sequences, with the most commonly used ones being:

- DieHard Tests [11]: A suite of statistical tests proposed by George Marsaglia, a mathematician from Florida State University, USA.
- NIST Tests [12]: A set of statistical tests developed by scientists from the National Institute of Standards and Technology (NIST), including Andrew Rukhin and others.
- TestU01 [13]: A suite of statistical tests developed by Pierre L'Ecuyer and other researchers from the University of Montreal.
- RaBiGeTe [14]: A suite of statistical tests complemented by a graphical interface for use in Windows.
- Knuth's Tests [15]: A suite of statistical tests proposed by Donald Knuth, a scientist from Stanford University.

These tests evaluate the generated binary sequences against various statistical criteria to check for randomness.

Methods and Materials

Knuth's Tests. Knuth's tests for studying the statistical properties of sequences include both graphical and evaluative tests [15].

In graphical tests, the statistical properties of the generated sequences, which are used for qualitative assessment of the results, are represented visually.

Graphical Tests by Donald Knuth:

- Histogram of Sequence Elements Distribution;
- Distribution on the Plane;
- Runs Test;
- Monotonicity Check;
- Autocorrelation Function;
- Linear Complexity Profile;
- Graphical Spectral Test.

Let's take a closer look at the characteristics of the graphical tests proposed by Donald Knuth [16,17]:

Histogram Distribution Test of Sequence Elements. This test allows us to assess the uniformity of the distribution of symbols in the examined sequence, as well as to determine the frequency of occurrence of a specific symbol. The histogram is constructed as follows: the number of occurrences of each element in the examined sequence is counted, and then a graph is plotted

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 18, JUNE 2024

showing the dependence of the number of occurrences of the elements on their numerical representation.

Plane Distribution Test. This test is designed to identify relationships between the elements of the sequence we are studying. Distribution on the plane is performed as follows: points with coordinates $(e_i; e_{i+1})$ are plotted on a field of size $m \times m$, where $i = \overline{1,(n-1)}$, $0 \le e_i \le m$, e_i are the elements of the studied sequence, n is the length of the sequence, and m the size of the alphabet.

Runs Test. This test assesses the uniform distribution of symbols in the studied sequence by determining the frequency of occurrence of 0s and 1s in a series consisting of k bits. It identifies how many times zeros, ones, and series consisting of two, three, etc., bits appear in the examined sequence.

Monotonicity Test. This test counts the lengths of increasing and decreasing segments of the examined sequence elements. The studied sequence is graphically represented as consecutive, non-overlapping segments of non-increasing and non-decreasing elements.

Autocorrelation Function (ACF) Test. This test is designed to evaluate the correlation between shifted copies of the obtained sequence.

Bitwise ACF. The bitwise ACF is defined as follows: the examined sequence is considered as a bit sequence, after which the obtained bit sequence is normalized $(1 \to 1,0 \to -1)$, and correlation spikes are calculated using formula (1):

$$c_j = \frac{\sum_{i=0}^{n-1} b_i \cdot b_{(i+j) mod\, n}}{\sum_{i=0}^{n-1} b_i^2}, \quad (1)$$

where b_i are the elements of the normalized sequence, n is the length of the normalized bit sequence, $i = \overline{0,(n-1)}, j = \overline{0,n}$.

Bytewise ACF. First, the examined sequence E is normalized. Let $a_{r-1}\, a_{r-2} \ldots a_0$, (where r is the bit width of the number) be the binary representation of the sequence. The normalized value of the elements is then computed using formula (2), and the correlation spikes are computed using formula (3):

$$b_i = \sum_{j=0}^{r-1} \left((-1)^{a_i} \cdot 2^j \right) \quad (2)$$

Next, the autocorrelation spikes are calculated similarly to the bitwise ACF using the formula:

$$c_j = \frac{\sum_{i=0}^{n-1} b_i \cdot b_{(i+j) mod\, n}}{\sum_{i=0}^{n-1} b_i^2}, \quad (3)$$

Linear Complexity Profile Test. The following approach is used to create a linear complexity profile: suppose we have a binary sequence $t = t_1\, t_2\, t_3 \ldots t_n$ of length n. We sequentially consider subsequences $t^{(k)}$, which contain the first k elements of the sequence, and plot a graph showing the dependence of the linear complexity on the length of the subsequence N.

Graphical Spectral Test. This test allows for the assessment of the uniformity of the distribution of 0s and 1s in the examined sequence based on the analysis of the height of the peaks in the Fourier transform. Suppose $t = t_1\, t_2\, t_3 \ldots t_n$ is a binary sequence of length n. We transform it into a sequence $x_1\, x_2\, x_3 \ldots x_n$, where $x_i = 2t_i - 1$ (т. е. $1 \to 1,0 \to -1$).

Now, we apply the discrete Fourier transform to x and obtain the sequence of harmonics Sj as shown in formula (4):

$$S_j = \sum_{k=1}^{n} x_k \left[cos\left(\frac{2\pi j}{n} (k-1) \right) + i \cdot sin\left(\frac{2\pi j}{n} (k-1) \right) \right] \quad (4)$$

DOI: 10.37943/18LYCW2723

Knuth's Evaluation Tests. The following statistical tests by D. Knuth were used to determine the randomness properties of the sequence:

- Runs Test;
- Monotonicity Test;
- Intervals Test;
- Combinations Test;
- Coupon Collector's Test;
- Permutation Test;
- Correlation Test.

Let the test results be such that they can be divided into k категорий. categories. We conduct n независимых испытаний, где n is a sufficiently large number. Let p_s be the probability that the result of a trial falls into the sth category, and Y_s be the number of trials that actually fall into the sth category. The value of the test statistic is calculated using formula (5):

$$\chi^2(obs) = \sum_{s=1}^{k} \frac{(Y_s - np_s)^2}{np_s} \tag{5}$$

Chi-square distribution tables are used to assess the obtained result. In these tables, the rows correspond to the degrees of freedom v, and the columns correspond to probabilities p. If the table contains the number x in row v and column p, it means that the value of χ^2 (obs) will be greater than x with the probability of p.

Runs Test. Let $t = t_1\, t_2\, t_3 \ldots t_n$ be a binary sequence of length n and m be the length of a run. The number of occurrences $v_{i_1 \ldots i_k}$ of all possible non-overlapping runs of length m (extra bits are discarded) is counted, and the statistic is computed as per formula (6):

$$\chi^2(obs) = \sum_{v_{i_1 \ldots i_k}} \frac{\left(v_{i_1 \ldots i_k} - \left[\frac{n}{k}\right] \cdot \frac{1}{2^k}\right)^2}{\left[\frac{n}{k}\right] \cdot \frac{1}{2^k}} \tag{6}$$

Monotonicity Test. This test checks the uniformity of symbol distribution in a sequence by analyzing segments of non-increasing and non-decreasing elements. Let $v_i, i = \overline{1,t}$ be the number of segments of non-increasing (non-decreasing) length i. The statistic is calculated using formula (7):

$$\chi^2(obs) = \sum_{i=1}^{t} \frac{\left[v_i - (\sum_{j=1}^{t} v_j) \cdot p_i\right]^2}{(\sum_{j=1}^{t} v_j) \cdot p_i}, \tag{7}$$

where $p_i = \frac{1}{i!} - \frac{1}{(i+1)!}$.

Intervals Test. This test aims to check the uniformity of symbol distribution in the examined sequence by analyzing the lengths of subsequences where all elements fall within a specific numerical interval.

Let $t = t_1\, t_2\, t_3 \ldots t_n$ be a sequence of k- bit numbers of length n. Let α and β be two integers satisfying the inequality $0 \le \alpha < \beta \le 2^k - 1$. The length of intervals between numbers lying in the range $[\alpha; \beta]$ is computed. Then, the number of intervals $v_i, i = \overline{0,m}$, is determined, and the statistic is calculated using formula (8):

$$\chi^2(obs) = \sum_{i=0}^{m} \frac{\left(v_i - \eta \frac{\beta-\alpha}{2^k}\left(1 - \frac{\beta-\alpha}{2^k}\right)^i\right)^2}{\eta \frac{\beta-\alpha}{2^k}\left(1 - \frac{\beta-\alpha}{2^k}\right)^i} \tag{8}$$

where $\eta = \sum_{i=0}^{m} v_i$ is the total number of intervals, and the degrees of freedom equal m.

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 18, JUNE 2024

Combinations Test. This test checks the distribution of symbols in the examined sequence by analyzing various combinations of numbers in the considered subsequences. Let $t = t_1\ t_2\ t_3 \ldots t_n$ be a sequence of k-bit numbers of length n. We divide it into subsequences, each of which has a length m (additional bits are excluded). We count the number of subsequences $v_i, i = \overline{1,m}$, containing i distinct numbers, and then compute the chi-squared statistic using formula (9):

$$\chi^2(obs) = \sum_{i=1}^{m} \frac{\left[v_i - \left[\frac{n}{m}\right] \cdot p_i\right]^2}{\left[\frac{n}{m}\right] \cdot p_i} \tag{9}$$

where $p_i = \frac{d(d-1)\ldots(d-i+1)}{d^m} \left\{ {m \atop i} \right\}$, $d = 2^k - 1$, and $\left\{ {m \atop i} \right\}$ are the Stirling numbers of the second kind.

Coupon Collector's Test. In this test, the uniform distribution of symbols in the examined sequence is checked by analyzing various combinations of numbers in subsequences. Let $t = t_1\ t_2\ t_3 \ldots t_n$ be a sequence of k-bit numbers of length n. The number of subsequences v_i of length $i, i = \overline{2^k, r}$, containing a complete set of numbers from 0 to $2^k - 1$ is calculated. Then the χ^2 statistic is calculated using formula (10):

$$(obs) = \sum_{i=d}^{r} \frac{[v_i - \eta \cdot p_i]^2}{\eta \cdot p_i}, \tag{10}$$

where $p_i = \frac{d!}{d^i} \left\{ {i-1 \atop d-1} \right\}$, $i = \overline{2^k, r}$, $p_r = \frac{d!}{d^{r-1}} \left\{ {r-1 \atop d} \right\}$, $\eta = \sum_{i=d}^{r} v_i$, $d = 2^k - 1$, and the number of degrees of freedom is $r - 2^k + 1$.

Permutations Test. In this test, the relative arrangement of numbers in subsequences is analyzed to check the uniform distribution of symbols in the examined sequence.

The sequence is divided into subsequences, each of length k (additional bits are excluded). In each subsequence, there are $k!$ possible permutations of relative arrangements of numbers. The number of occurrences of each permutation, $v_i, i = \overline{1,k!}$, is counted, and the chi-squared statistic is calculated using formula (11):

$$\chi^2(obs) = \sum_{i=1}^{r!} \frac{\left[v_i - \left[\frac{n}{k}\right] \cdot \frac{1}{k!}\right]^2}{\left[\frac{n}{k}\right] \cdot \frac{1}{k!}} \tag{11}$$

Correlation Test. This test checks for the mutual independence of sequence elements. The statistic is calculated using formula (12):

$$C_j = \frac{n(t_0 t_j + t_1 t_{(1+j) mod n} + \cdots + t_{n-1} t_{(n-1+j) mod n}) - (t_0 + t_1 + \cdots + t_{n-1})^2}{n(t_0^2 + t_1^2 + \cdots + t_{n-1}^2)(t_0 + t_1 + \cdots + t_{n-1})^2} \tag{12}$$

For any j, the value C_j should lie within the interval:

$$[\mu_n - 2{,}43\sigma_n;\ \mu_n + 2{,}43\sigma_n],$$

where

$$\mu_n = -\frac{1}{n-1}, \sigma_n^2 = \frac{n^2}{(n-1)^2(n-2)}.$$

Results and discussion

Development of a pseudorandom sequence generation algorithm

The algorithm for generating pseudorandom sequences (PRS) utilized in this work was previously developed in the Information Security Laboratory. The PRS generation algorithm consists of three stages.

DOI: 10.37943/18LYCW2723

Stage 1. The initial parameters of the algorithm are as follows:
An initial numerical sequence $X=\{x_1, x_2,\ldots, x_n\}$, where $x_i \in Z_{256}$;
An irreducible polynomial $p(x)$ of degree 8, used as a polynomial modulus;
The length of the generated sequence $m \geq 1$.
The output is the sequence $Z=\{z_1, z_2,\ldots, z_m\}$, where $z_j \in Z_{256}$.
The transformation of the initial numerical sequence is carried out according to the following rules:

$$x_2^{(i)} = S\left(x_1^{(i-1)}\right), \ldots, x_{n-1}^{(i)} = S\left(x_{n-2}^{(i-1)}\right), x_n^{(i)} = S\left(x_{n-1}^{(i-1)}\right),$$

$$x_1^{(i)} = f_1\left(x_{n-1}^{(i-1)}, x_n^{(i-1)}\right) \oplus x_2^{(i)}, i = \overline{1,(n+m)},$$

$$z_j = f_1\left(x_{n-1}^{(i-1)}, x_n^{(i-1)}\right), i > n, j = (n-i),$$

where $f_1(a,b) = \left(S(a) \oplus (S(b) \cdot x^3)\right) \bmod p(x)$.
All generated $z_j, j = \overline{1,m}$, are saved as elements of the generated sequence, since the first n elements are not counted as results. Thus, the sequence generation starts from the $(n+1)$ th step. Here, the symbol $\oplus$ denotes the bitwise XOR operation and $\bmod p(x)$ represents the remainder of the division of the resulting polynomial by the given irreducible polynomial $p(x)$.
Stage 2. The input data (initial parameters) for this algorithm are as follows:
An initial numerical sequence $X=\{x_1, x_2,\ldots, x_n\}$, where $x_i \in Z_{256}$;
Irreducible polynomials $p(x)$ and $q(x)$ of degree 8, used as polynomial moduli, with $p(x) \neq q(x)$;
The length of the generated sequence $m \geq 1$.
The output is the pseudorandom sequence $Z=\{z_1, z_2,\ldots, z_m\}$, where $z_i \in Z_{256}$.
The transformation of the initial numerical sequence is carried out according to the following rules:

$$x_1^{(i)} = \left(\left(f_2\left(x_1^{(i-1)}, x_2^{(i-1)}\right) \cdot x^3\right) \oplus f_2\left(x_2^{(i-1)}, x_3^{(i-1)}\right)\right) \bmod q(x),$$
$$x_2^{(i)} = \left(\left(f_2\left(x_2^{(i-1)}, x_3^{(i-1)}\right) \cdot x^3\right) \oplus f_2\left(x_3^{(i-1)}, x_4^{(i-1)}\right)\right) \bmod q(x),$$
$$\ldots.$$
$$x_{n-1}^{(i)} = \left(\left(f_2\left(x_{n-1}^{(i-1)}, x_n^{(i-1)}\right) \cdot x^3\right) \oplus f_2\left(x_n^{(i-1)}, x_1^{(i-1)}\right)\right) \bmod q(x),$$
$$x_n^{(i)} = x_{n-1}^{(i)} \oplus \left(\left(\left(f_2\left(x_n^{(i-1)}, x_1^{(i-1)}\right) \cdot x^3\right) \oplus x_1^{(i)}\right) \bmod q(x)\right), i = \overline{1,(n+l)},$$
$$z_j = \left(\left(f_2\left(x_n^{(i-1)}, x_1^{(i-1)}\right) \cdot x^3\right) \oplus x_1^{(i)}\right) \bmod q(x), i > n, j = (n-i),$$

where $f_2(a,b) = S\left(\left(\mathrm{S}(a) \oplus (b \cdot x)\right) \bmod p(x)\right)$.
Stage 3. The input data (initial parameters) for this algorithm are as follows:
An initial numerical sequence $X=\{x_1, x_2,\ldots, x_n\}$, where $x_i \in Z_{256}$;
Irreducible polynomials $p(x)$ and $q(x)$ of degree 8, used as polynomial moduli, with $p(x) \neq q(x)$;
The length of the generated sequence $m \geq 1$.
The output is the pseudorandom sequence $Z=\{z_1, z_2,\ldots, z_m\}$, where $z_i \in Z_{256}$.
As mentioned above, this algorithm is a combination of the two previous stages, meaning the generated sequences are combined using bitwise XOR (modulo 2). The resulting sequence is then stored as the final PRS (Figure 1).

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 18, JUNE 2024

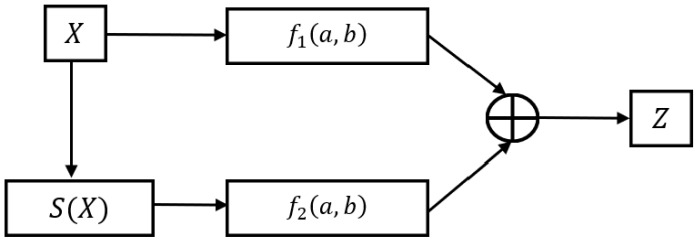

Figure 1. Schematic of the PRG_ISL Algorithm

Results of statistical evaluation of the PRG_ISL generator

One of the primary criteria for evaluating the security of a randomly generated sequence is the investigation of its statistical security. Statistical tests are empirical tests designed to assess the quality of PRNGs (Pseudorandom Number Generators), identify their weaknesses, calculate the statistical characteristics of the generated sequences, and compare these characteristics with those of truly random sequences. When testing the randomness properties of a generated sequence, the sequence is considered statistically secure if it is proven to be generated in a random manner. In this work, we examine the statistical characteristics of the output sequence of the PRG_ISL generator using NIST and D. Knuth's tests.

For the study of the statistical security of the PRNG using D. Knuth's graphical tests, a bit sequence of 98 KB, generated by the PRG_ISL, was selected. The results of the study are shown in Figures 2 and 3.

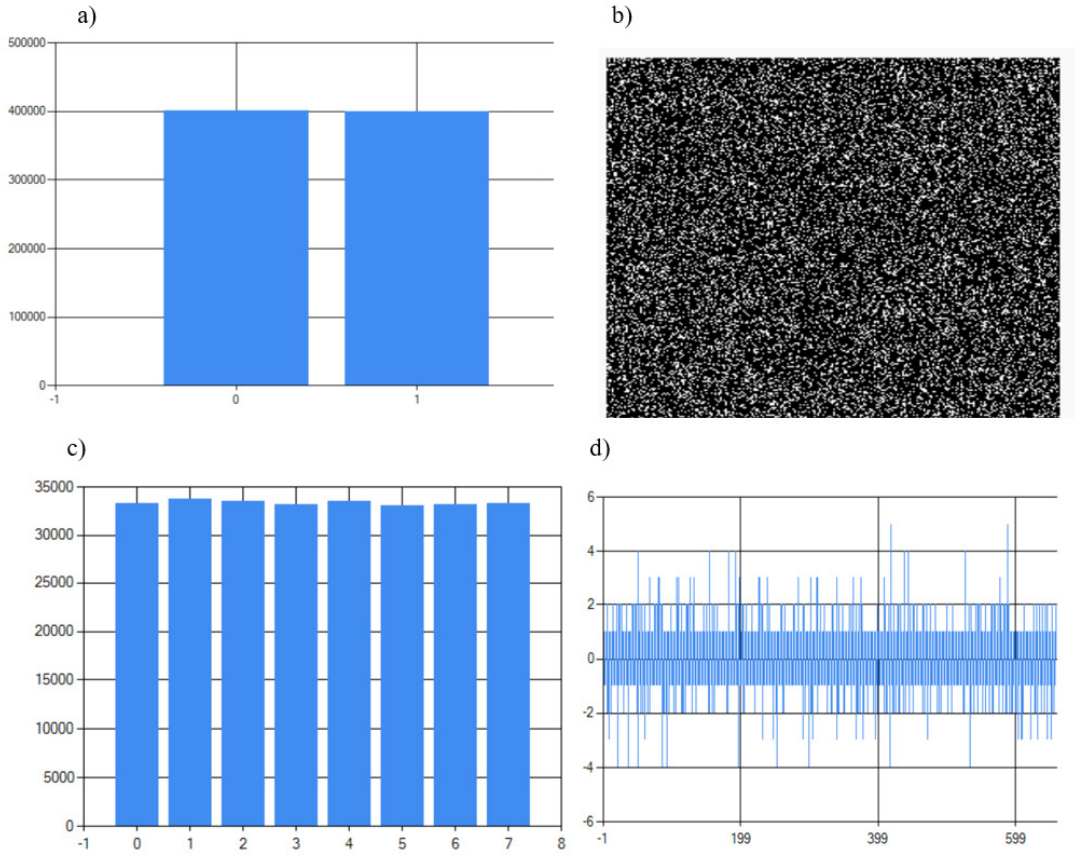

Figure 2. Results of D. Knuth's Graphical Tests:
a) – Histogram of sequence elements distribution;
b) – distribution on the plane; c) – runs test; d) – monotonicity check.

DOI: 10.37943/18LYCW2723

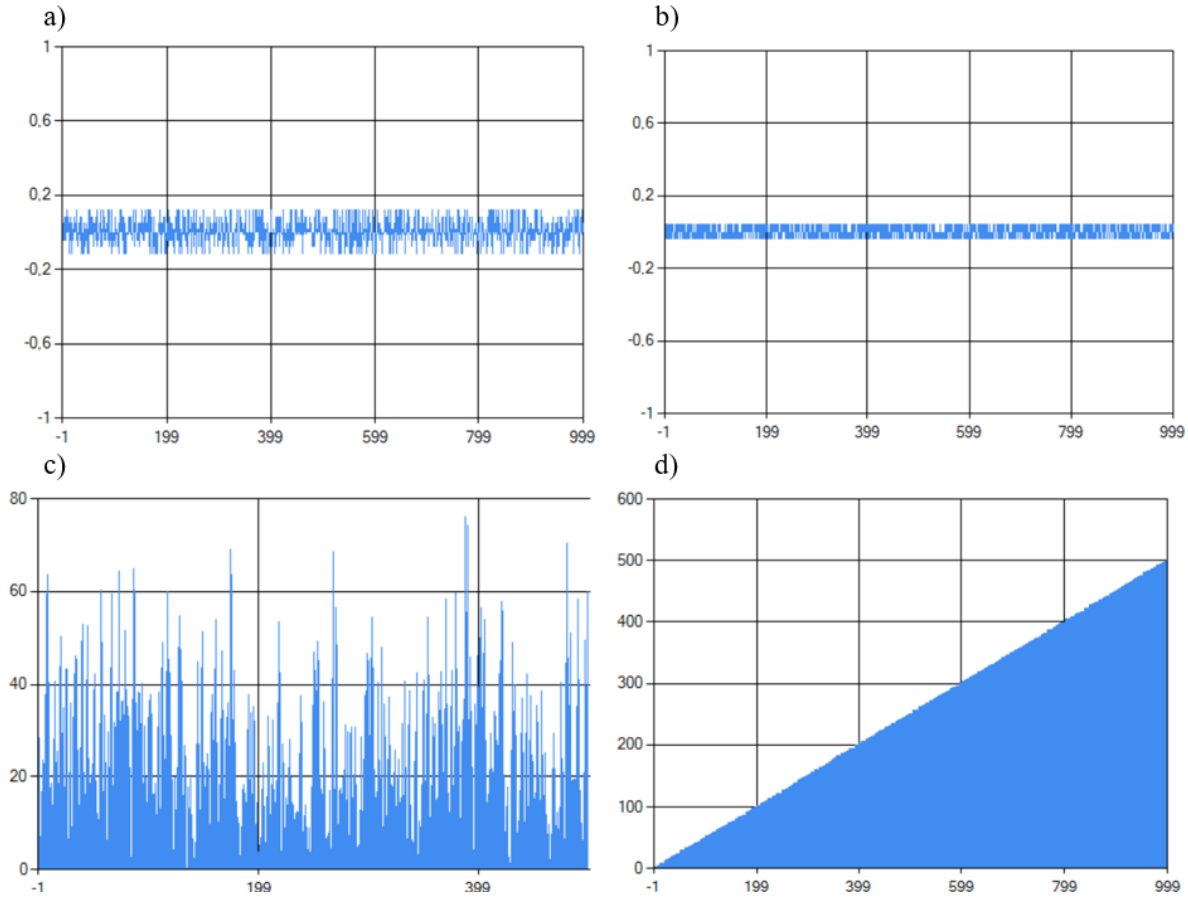

Figure 3. Results of D. Knuth's Graphical Tests:
a) – Bitwise autocorrelation function; b) – Bytewise autocorrelation function;
c) – Linear complexity profile; d) – Graphical spectral test.

For a thorough analysis of sequences generated by the PRG_ISL generator, we applied graphical tests developed by D. Knuth to 1000 files, each of which has a length of 10,000 bytes, with initial parameters of 256 bits. The results of this study are presented in Table 1.

Table 1. Results of Knuth's graphical tests

№	Graphical Tests	Number of files successfully passed testing
1	Histogram of sequence elements distribution	1000
2	Distribution on the plane	992
3	Runs Test	996
4	Monotonicity check	996
5	Bitwise autocorrelation function	997
6	Byte-wise autocorrelation function	996
7	Graphical spectral test	998
8	Linear complexity profile	1000

Unlike graphical tests, the interpretation of results in the case of estimation tests is determined as either 'passed' or 'failed' using specific numerical characteristics. To study the statistical properties of the generated sequences using estimation tests, we analyzed 1000 files obtained from the PRG_ISL generator. The results are shown in Figure 4.

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 18, JUNE 2024

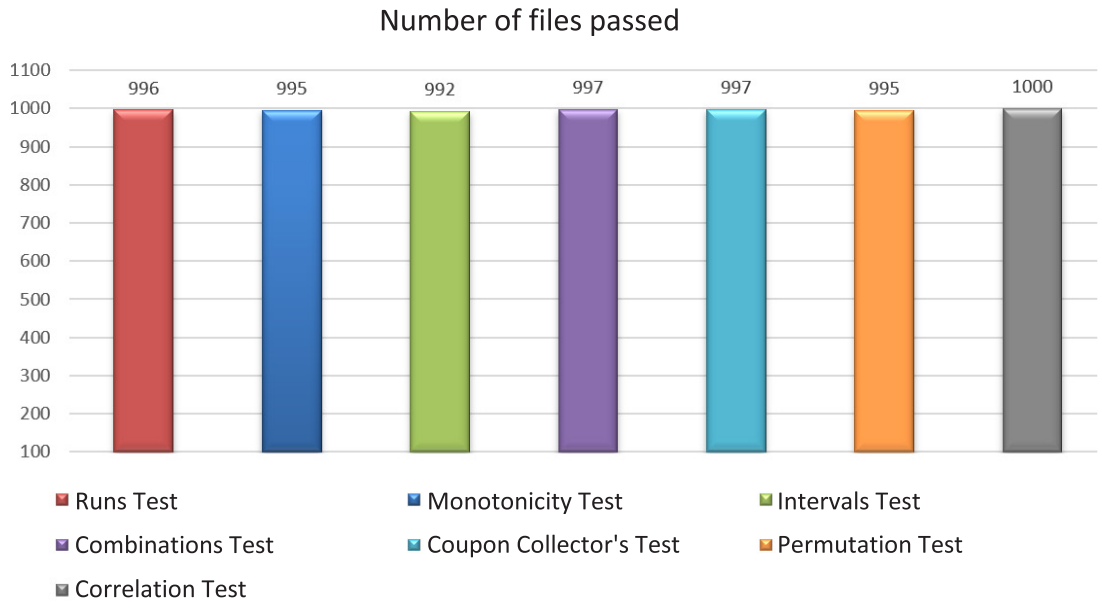

Figure 4. Results of Knuth's Evaluation Tests

According to the analysis of the number of files that did not pass the testing, significant deviations in the chi-square values were not detected.

Results of NIST Tests

Currently, the most widely used statistical tests are the NIST tests. The NIST tests consist of 15 statistical tests: frequency (monobit) test, frequency test within a block, runs test, longest run of ones in a block test, binary matrix rank test, spectral test, non-overlapping template matching test, overlapping template matching test, Maurer's universal statistical test, linear complexity test, approximate entropy test, serial test, cumulative sums test, and two different random excursions tests[18,19].

The result of each NIST test is determined by detecting various deviations during the hypothesis testing of the randomness of the examined sequence. For example, let's calculate the P-value, which determines the probability of obtaining a random sequence from the examined sequence. If the research, depending on the obtained P-value, is conducted in the area α=0,01, then the following conclusion is drawn. If $P \geq 0{,}01$, then the sequence generated from the PRG_ISL generator is 99% random, and if $P < 0{,}01$, then the sequence is 99% not random[20,21].

To investigate the statistical security of the output sequences of the PRG_ISL generator using the NIST tests, the same 1000 files were used as for the Knuth statistical tests. The results of the study are shown in Figure 5.

DOI: 10.37943/18LYCW2723

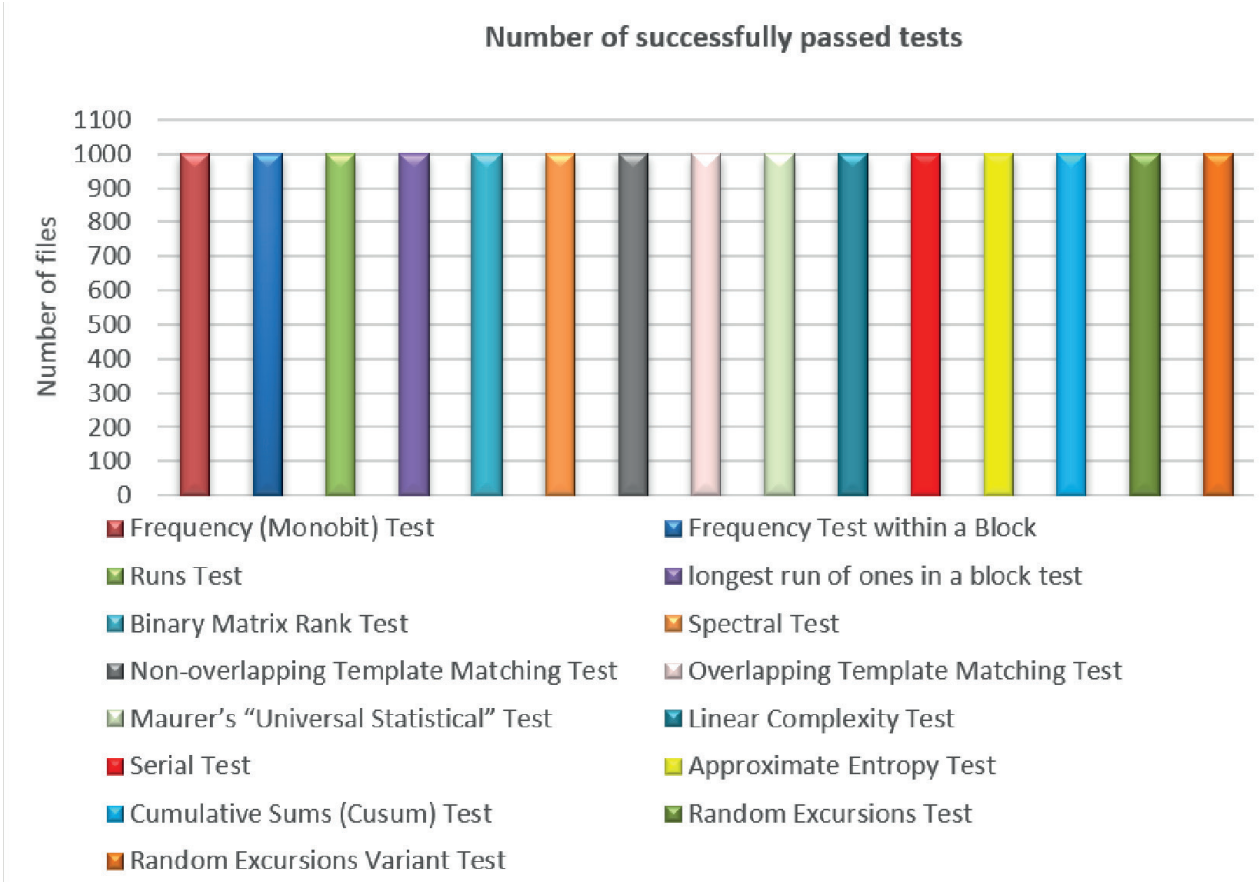

Figure 5. Results of NIST Tests

The results of the conducted tests show that all files passed the NIST testing successfully, as it was determined that the condition $P \geq 0{,}01$ is satisfied for all test types. Thus, it can be concluded that according to the NIST tests, the PRG_ISL generator is 100% secure, as it fully satisfies the criteria of statistical security.

Conclusion

Pseudorandom sequence generators can be utilized for various purposes, including addressing critical data security tasks. Ensuring data protection requires a reliable verification of the proximity properties of output sequences from generators to truly random ones in terms of their statistical properties and unpredictability of output values. The pseudorandom sequence generator proposed in the article is used for creating secure keys in a post-quantum digital signature algorithm based on hash functions developed under a grant project.

The PRG algorithm consists of three steps. At each step, the generated pseudorandom sequence performs its function in accordance with the normal distribution law. Tests by D. Knuth and NIST were used to verify the statistical security of the proposed PRG. NIST tests are considered the most suitable in terms of efficiently assessing the properties of pseudorandom sequences and their usability across different platforms. The results of the conducted NIST tests showed positive outcomes, and the findings obtained from the graphical and evaluative tests by D. Knuth demonstrated that the requirements for pseudorandom sequences are fully met.

According to the research results, it was established that the proposed PRG algorithm can be safely used in the field of information security for generating reliable secret keys, passwords, and pseudorandom sequences of various lengths.

Scientific Journal of Astana IT University
ISSN (P): 2707-9031 ISSN (E): 2707-904X
VOLUME 18, JUNE 2024

Acknowledgment. The work was carried out within the framework of the grant funding program AP14870719 «Development and study of post-quantum cryptography algorithms based on hash functions» of the Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan.

References

[1] Popereshnyak, S. (2020). Technique of the testing of pseudorandom sequences, Svitlana Popereshnyak. International Journal of Computing, 19(3), 387-398.
[2] Park, S., Kim, K., Kim, K., Nam, C. (2022). Dynamical Pseudo-Random Number Generator Using Reinforcement Learning. Appl. Sci., 12(3377). https://doi.org/10.3390/app12073377
[3] Pasqualini, L., Parton, M. (2020). Pseudo Random Number Generation: a Reinforcement Learning approach. International Workshop on Statistical Methods and Artificial Intelligence (IWSMAI), Procedia Computer Science, 170, 1122–1127.
[4] Kietzmann, P., Schmidt, T.C., Wählisch, M.A. (2022). Guideline on Pseudorandom Number Generation (PRNG) in the IoT. ACM Comput. Surv., 54, 1–38. https://doi.org/10.1145/3453159
[5] Dichtl, M., Golić, J.D. (2007). High-Speed True Random Number Generation with Logic Gates Only. In Cryptographic Hardware and Embedded Systems—CHES 2007: Lecture Notes in Computer Science Book Series, 2007, 45–62. https://doi.org/10.1007/978-3-540-74735-2_4
[6] Khalique, Aqeel & Lone, Auqib & Ashraf, Syed. (2015). A Novel Unpredictable Temporal based Pseudo Random Number Generator. International Journal of Computer Applications. 117. 23-28. http://doi.org/10.5120/20615-3301
[7] Haider, T., Blanco, S.A., Hayat, U. (2024). A novel pseudo-random number generator based on multivariable optimization for image-cryptographic applications, Expert Systems with Applications, 240(122446). https://doi.org/10.1016/j.eswa.2023.122446
[8] Maksymovych, V., Shabatura, M., Harasymchuk, O., Shevchuk, R., Sawicki, P., Zajac, T. (2022). Combined Pseudo-Random Sequence Generator for Cybersecurity. Sensors, 22(24),9700. https://doi.org/10.3390/s22249700
[9] Ofelius Laia et.al., (2019). Application of Linear Congruent Generator in Affine Cipher Algorithm to Produce Dynamic Encryption. International Conference of SNIKOM 2018. Journal of Physics: Conference Series, Vol. 1361.
[10] AL-khatib, Mohammed & Lone, Auqib. (2018). Acoustic Lightweight Pseudo Random Number Generator based on Cryptographically Secure LFSR. International Journal of Computer Network and Information Security, 10. 38-45. http://doi.org/10.5815/ijcnis.2018.02.05
[11] Feng, Yulong & Hao, Lingyi. (2020). Testing Randomness Using Artificial Neural Network. IEEE Access. 8. 163685-163693. http://doi.org/10.1109/ACCESS.2020.3022098
[12] Savelov, M. (2023). The limit joint distributions of statistics of three tests of the NIST package. Discrete Mathematics and Applications, 33(4), 247-257. https://doi.org/10.1515/dma-2023-0022
[13] Bassham, L., Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Leigh, S., Levenson, M., Vangel, M., Heckert, N., Banks, D. (2022). A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. Special Publication (NIST SP); National Institute of Standards and Technology: Available online: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=906762
[14] Popereshnyak, S. (2019). Analysis of pseudorandom small sequences using multidimensional statistics, Proceedings of the 2019 3rd IEEE International Conference on Advanced Information and Communication Technologies (AICT'2019), 541-544.
[15] Pierre, L'Ecuyer., Richard, Simard. (2007). TestU01: A C library for empirical testing of random number generators. ACM Transactions on Mathematical Software, 33(4):22-39. http://doi.org/10.1145/1268776.1268777
[16] Sun., Y., Lo, B. (2018). Random number generation using inertial measurement unit signals for on-body IoT devices, Living in the Internet of Things: Cybersecurity of the IoT - 2018, 1-9, http://doi.org/10.1049/cp.2018.0028
[17] Koçak, Onur. (2018). Modifications of knuth randomness tests for integer and binary sequences. Communications Faculty Of Science University of Ankara Series A1, Mathematics and Statistics. 67. 64-81. https://doi.org/10.1501/Commua1_0000000862
[18] Kapalova, N., Khompysh, A., Arici, M., Algazy, K., & Pham, D. (2020). A block encryption algorithm based on exponentiation transform. Cogent Engineering, 7(1). https://doi.org/10.1080/23311916.2020.1788292

[19] Khompysh, A., Kapalova, N., Lizunov, O., Dilmukhanbet, D., Kairat, S. (2023). Development of a new lightweight encryption algorithm. International Journal of Advanced Computer Science and Applications, 14(5), 452-459. https://doi.org/10.14569/IJACSA.2023.0140548
[20] Burciu, P., Simion, E. (2019). A systematic approach of NIST statistical tests dependencies. Journal of Electrical Engineering, Electronics, Control and Computer Science, 5(1), 1-6. https://jeeeccs.net/index.php/journal/article/view/113/93
[21] Sulak F., Uğuz M., Koçak O., Doğanaksoy A. (2017). On the independence of statistical randomness tests included in the NIST test suite. Turkish Journal of Electrical Engineering & Computer Sciences, 5(25), 3673-3683. http://doi.org/10.3906/elk-1605-212
[22] Kapalova, N., Algazy, K., Haumen, A., Sakan, K. (2023). Statistical analysis of the key scheduling of the new lightweight block cipher. International Journal of Electrical and Computer Engineering (IJECE), 13(6), 817-6826. http://doi.org/10.11591/ijece.v13i6.pp6817-6826
[23] Pikuza, M.O., & Mikhnevich, S. Yu. (2012). Testing of hardware random number generator using a set of NIST statistical tests. Reports of BSUIR, 19(4), 37-42. https://doi.org/10.35596/1729-7648-2021-19-4-37-42