INFORMATION AND CONTROLLING SYSTEM

The object of this study is the methods and algorithms for computing, evaluating, and comparing DNA-based pseudorandom sequences (PRSs) and random sequences (RSs). This paper addresses the task of extracting (P)RSs with the required stochastic and statistical properties from a DNA noise source, experimentally validating these properties in compliance with the requirements of current international standards, as well as evaluating and comparing the sequences obtained for different DNA samples. The results are the developed algorithms for generating DNA-based (P)RSs, improved algorithms for their comparison, and proposals for the process of evaluating their properties. For the output of implemented algorithms, more than 96 % of the bit streams of sequences successfully pass all statistical tests, and the entropy per bit for RSs is close to 1. A special feature of the developed generation algorithms is the use of validated conditioning component – block cipher in CTR mode – which explains the possibility of obtaining unique random data with required properties. The peculiarity of the proposed evaluation algorithm is the complexity of the tests and checks used: complete assessment of statistical properties and entropy values. Special features of the improved comparison algorithms are resource saving and the ability to evaluate much larger data sets. This is due to the use of structures and data types that are better in terms of memory usage and flexible cryptographic primitives with different modes.

The results could be practically applied to constructs where randomness is required: for computing the keys and system-wide parameters, when performing transformations as part of hash functions, while obtaining sequences of an arbitrary alphabet, in zero-knowledge protocols, etc.

Keywords: DNA, random sequences, randomness extractors, statistical testing, similarity evaluation, k-mers, MinHash

UDC 004.056.5
DOI: 10.15587/1729-4061.2024.314808

GENERATION, EVALUATION AND COMPARISON OF DNA-BASED (PSEUDO) RANDOM SEQUENCES

Ivan Gorbenko
Doctor of Technical Sciences, Professor*
Chief Designer**
Yaroslav Derevianko
Corresponding author
PhD Student*
Researcher-Consultant**
E-mail: yarik0009258@gmail.com
*Department of Cybersecurity of Information Systems, Networks and Technologies
V. N. Karazin Kharkiv National University
Svobody sq., 4, Kharkiv, Ukraine, 61022
**JSC "Institute of Information Technologies"
Profesora Otamanovskoho str., 15,
Kharkiv, Ukraine, 61166

Received date 16.08.2024
Received in revised form 07.10.2024
Accepted date 25.10.2024
Published date 12.11.2024

How to Cite: *Gorbenko, I., Derevianko, Y. (2024). Generation, evaluation and comparison of DNA-based (pseudo) random sequences. Eastern-European Journal of Enterprise Technologies, 6 (9 (132)), 6–24. https://doi.org/10.15587/1729-4061.2024.314808*

1. Introduction

Providing users with services of confidentiality, integrity, availability, non-repudiation, and crypto survivability, which directly depend on the cryptographic properties of the structures used, is the main requirement for modern information systems. They, in turn, clearly depend on the properties and quality of pseudo-random sequences (PRSs) and random sequences (RSs) used in crypto transformations. That is, an important feature of most crypto primitives, which is justified by the necessity of randomness, is the mandatory use of PRS and RS. Depending on the purpose, they are used to generate asymmetric and symmetric key pairs, to calculate parameters during the execution of algorithms and transformations, such as nonce or seed values, etc. [1].

Hereafter, the following definitions of RS, PRS, as well as the definition of a random number (RN) were used [1]. RS is a sequence of independent and equally distributed variables. PRS is a sequence of symbols obtained using a deterministic algorithm, which is computationally indistinguishable from a random one. RN are discrete values (bits, bit strings, integers) that are obtained from a noise source (NS) at separate time points.

Work and development in the direction of improving the randomness generation process is actively conducted at the international and national levels and requires special attention. The relevance of research and implementation of new algorithms for obtaining randomness is justified by the fact that the non-compliance of random data with modern requirements can with a high probability significantly weaken or compromise any cryptographic transformation or its components.

The US National Institute of Standards and Technology (NIST) and the German Federal Office for Information Security (BSI) emphasize the importance of developing algorithms, methods, and means for generating/calculating random data. They determine the relevance of building new structures to obtain randomness due to the need to improve cryptographic primitives that use random data (keys and parameters) and strengthen their security in order to counter existing and new attacks in the post-quantum period.

Considering the current state of development of cryptography in the field of randomness, there is a need both for new sources of randomness generation (noise sources), and for algorithms and tools that can obtain sequences with good randomness properties from samples from such sources. The urgency of building such structures is determined by the fact that usually the sample from NS does not have the necessary properties, for example, uniformity, and needs improvement.

Thus, certain physical or non-physical phenomena can be used as a source of randomness. In particular, DNA cryptography is an interesting area related to the new NS. However, scientific research on this topic is not very popular and developed, and even within it, obtaining data based on DNA is of secondary importance.

To make a decision about the need to improve the sample, as well as to be able to determine the degree of this improvement, it is important to be able to correctly and fully assess its properties. Therefore, the current area of modern research is also construction of new comprehensive evaluation methods that would make it possible to clearly determine to what extent the properties of the sequences meet the requirements, whether they are safe enough for use, and whether they need improvement.

Therefore, taking into account the lack of research into this area, as well as the recommendations by the main international organizations in the industry, the implementation of RNG and obtaining PRS and RS, in particular on the basis of DNA, is an actual and promising direction. Research on this topic is important because the use of unverified data with inappropriate values of statistics such as uniformity and low entropy per bit values is highly likely to significantly weaken or compromise cryptographic systems and primitives. This determines the need to calculate keys and parameters of cryptographic transformations based on the use of verified random data, the mandatory standardization and certification of algorithms and means for generating, evaluating, and comparing RS and RN.

The results of such studies are needed in practice as they could make it possible to create prerequisites for improving constructions that use randomness and improving their security. Also, owing to new advancements in this area, more accurate verification and standardization of received random data would be possible for further use both in the field of information security and in other fields.

2. Literature review and problem statement

The formation of RS is possible using both physical and non-physical NSs, and the formation of PRS occurs with the use of deterministic random number generators (DRNG), the input data for which are relatively short RSs [2]. Study [3] shows the theoretical possibility of using DNA as NS. According to [2], it can be classified as a non-physical NS, and therefore create a non-physical RNG.

In order to use the data obtained from NS or RNG, it is necessary to carry out their preliminary assessment. For a comprehensive and correct quantitative and qualitative assessment of randomness, it is suggested to use NIST recommendations [4, 5]. According to them, the sequence properties are divided into statistical and stochastic. Statistical ones include, for example, the uniformity of distribution, the presence of periodic patterns, the degree of compression, the frequency of sign changes, etc. The main stochastic indicator is a measure of the unpredictability of the information content, that is, in the case of sequences, the value of entropy per bit.

In the case of a partial non-compliance of sequence properties, for example, samples from NS, it is possible to use extraction [6] by means of encryption – block or stream ciphers [4].

Samples from DNA noise source have already been used as random data in some studies, but these works have some drawbacks. Paper [7] proposed the use of DNA as a random sequence, namely a random key for encryption using Rabin's cryptosystem and Feistel's design. The authors propose a simple conversion of a DNA sequence into a binary one according to a certain rule and the use of such a binary sequence as a key. However, issues related to checking the properties of such a key remained unresolved. The described method for obtaining the key is questionable since the statistical properties of the raw DNA sequence do not meet modern requirements, in particular [5], so that it can be used as an encryption key. Using key data with inappropriate properties could lead to the weakening of the cryptosystem or its complete breaking.

The criterion for evaluating the compliance of a certain statistical property is the P-value, which summarizes the strength of the evidence against the null hypothesis, which in the case of statistical testing is that the sequence being tested is random. The p-value is calculated as a specific function (depending on the test) of the test statistic. Test statistics are calculated differently for each test. The value of the test statistic is compared with the critical value. If the value of the test statistic exceeds the critical value, the null hypothesis of randomness is rejected. For example, for a monobit test that tests the ratio of zeros to ones, the test statistic would be the absolute value of the sum of zeros (–1) and ones (1) in the sequence divided by the square root of the length of the sequence. The p-value will be calculated by applying the complementary error function (erfc) [5].

For statistical tests, each P-value is the probability that a perfect random number generator would produce a sequence less random than the one being tested. The level of significance (α) is chosen for the tests. If the P-value $\geq\alpha$, then the null hypothesis is accepted, that is, the sequence is random. If the P-value $<\alpha$, then the null hypothesis is rejected, that is, the sequence is not random [5]. Usually, α is chosen in the range [0.001, 0.01].

A similar application example, which proposes obtaining secure keys using a DNA-based one-time pad (OTP) scheme, is provided in [8]. It is proposed to use the DNA of the selected gene as a codebook. Although the work, in contrast to [7], reports the results of the received keys successfully passing tests from the FIPS 140-2 standard, the unprocessed DNA sequence is still used as keys. The selection of the key is more random than in [7] since the place in the gene from which the key will be obtained is chosen randomly, and the length of the key depends on the length of the plaintext. The disadvantage of the encryption method described in [8] is that it does not take into account the possible similarity of the plaintext data and the value from the codebook – the key. Since the XOR operation is used for encryption, in this case their partial or complete mutual elimination will occur and, as a result, the ciphertext will not meet the necessary security requirements. The latest AIS 20/AIS 31 revision [2] recommends not using XOR, for example as a state transition function, because such an operation is not considered cryptographic and therefore cannot be used to obtain random or secret data. An option to overcome such a drawback is the use of more advanced or complex means of encryption: substitutions, permutations, etc. An important note regarding this method is also the need to use the key in the OTP scheme only once, which is quite difficult to achieve using DNA. This is due to the fact that different parts of the DNA sequences can be identical, and the same keys will be used to encrypt different messages. Another possible limitation of this method is the size of the received data, since long DNA sequences are needed to obtain rather long keys, i. e., random data, which are used as codebooks.

A possible option for overcoming the non-compliance of statistical indicators with the requirements [5] is the use

of substitutions, where different binary combinations are matched to the nitrogenous base according to certain rules. This approach is used in [9] and was presented in [10]. However, using such an approach simply changes the combinations to which the different nitrogenous DNA bases correspond, without changing the distribution and structure of the binary data itself. If different combinations are used for the same nitrogenous bases in one sequence, then the very essence of obtaining binary data from DNA is lost. Also, an unresolved issue in [9] is the correctness of testing the received binary sequences. In the work, sequences are split into 100-bit bit streams for further evaluation using the NIST statistical test suite [5]. However, [5] implies the use of significantly longer sequence lengths for most tests in the set (10^3–10^7 and more). In addition, the reported research results lack data from some tests from the set. Thus, the results in [9] cannot be considered completely reliable, and the described transformation method is not capable of forming sequences with the required statistical properties, which will be shown in this work.

Regarding the comparison of DNA sequences and binary sequences, works [11, 12] present tools for comparison by preliminary alignment. The method they use is called pairwise progressive sequence alignment. This heuristic method first performs a pairwise alignment of sequences for all pairs that can be constructed from a set of sequences. Then, a dendrogram (spanning tree) of sequences is constructed according to pairwise similarity. Finally, a multiple sequence alignment is constructed by aligning in the order defined by the tree.

Paper [13] also suggests the use of the Needleman-Wunsch algorithm [14] for preliminary alignment and comparison of sequences. The algorithm essentially divides a large problem (such as a complete sequence) into a number of smaller problems and uses the solutions of the smaller problems to find the optimal solution to the larger problem. It is also sometimes called the optimal matching algorithm and the global alignment technique.

The disadvantages of the comparison methods described above are the difficulties associated with the need for preliminary alignment, which requires significant computational and time costs. The previous statement also implies a limit on the size of sequences that can be evaluated. Therefore, such methods should be improved and replaced with more modern and effective ones. An option to overcome these difficulties is to use methods without preliminary alignment.

This is the approach used in [15], in which a method for comparing two DNA sequences without preliminary alignment is proposed. The method is to calculate a score using fuzzy membership values that are generated automatically based on the number of matches and discrepancies. Although the results reported in [15] are indicated as unique and correct, preliminary analysis indicates that such a comparison is a simple element-by-element comparison of matches between two sequences. That is, the method counts the number of identical elements for identical sequence indices, and the "score" is nothing more than a normalized index of matches.

Another option is a method that makes it possible to quickly compare sequence similarity without prior measurement by counting *k-mer* and calculating *k-mer distance*s. This is the approach proposed in [16]. However, the algorithm proposed in the work calculates all possible *k-mer* of the 4^k space, which causes a rather serious consumption of memory and hardware resources. This shortcoming needs further improvement.

An option is to use the MinHash algorithm to calculate sequence similarity, which is proposed in [17]. MinHash is a technique for quickly evaluating the similarity between two sets. For the first time, such a scheme was presented in [18]. Before the sequences can be compared, a "sketch" must first be created based on them, which gives a much-reduced representation of them. Sketches are used by the MinHash algorithm to quickly estimate the distance. To create a sketch, each *k*-mer in the sequence is hashed, which produces a pseudo-random identifier. After sorting the identifiers (hashes), a small subset from the top of the sorted list (min-hashes) can represent the entire sequence. The more similar the sequences are, the more common min-hashes they will have [17]. The limitation of this method is the use of hash functions with an output length of 32 and 64 bits, which, when applied to rather long *k*-mers, will not make it possible to uniformly map their entire space in the hash.

Systematizing the results of our review, one can state that the most critical shortcoming of all considered works related to obtaining random sequences based on DNA cryptography [7–9] is the non-compliance of the statistical properties of the obtained data with the requirements from [5]. Important shortcomings include either the complete lack of assessment of the properties of these data, or the use of outdated or narrowly focused methods. Also, the size of the obtained data is a limitation of the above studies. The disadvantage of the considered sequence comparison tools, in particular [16, 17], is the insufficient optimization of the use of resources and the impossibility of assessing the similarity of rather long sequences.

Therefore, the problem solved in the study is the extraction of PRSs and RSs with the required stochastic and statistical properties from the DNA noise source, experimental confirmation of these properties according to the requirements of current international standards, as well as evaluation and comparison of the obtained sequences for DNA of different subjects.

The essence of the problem is the fact that sequences, when DNA is represented in binary form, do not have the necessary statistical properties, but have the necessary stochastic (entropy) properties. Considering this, the required statistical properties can be obtained by applying encryption means using a standard block or stream cipher on session keys or long-term keys. In this study, for example, such an extractor has been implemented on the basis of DSTU 7624-2014 ("Kalyna") under the stream mode of operation (CTR).

Summarizing, to solve the task, the following is proposed:

– application of NIST-recommended conditioning components to enhance sequence properties;

– the use of modern methods of evaluating sequences, which involve full statistical and stochastic (entropy) testing;

– improvement of resource management in comparison algorithms;

– the use of more universal crypto primitives.

Thus, the above analysis reveals the expediency of conducting research in the area of implementing new algorithms for the generation and comparison of sequences based on DNA cryptography. The study and correct integrated application of algorithms for evaluating the properties of such sequences are also important.

3. The aim and objectives of the study

The purpose of our study is to implement algorithms for generating DNA-based PRS and RS with the required

statistical properties and entropy indicators, to provide suggestions for an algorithm for correct evaluation of these properties, to implement more effective comparison algorithms, as well as to directly evaluate and compare the obtained sequences with implemented algorithms. The use of DNA as a noise source could open up new opportunities for obtaining random data (sequences, numbers) for use, for example, in the process of calculating keys and system-wide parameters of the latest national and international quantum-resistant transformations. This, in turn, would create prerequisites for improving constructions that use randomness and improving their security. The implementation of evaluation and comparison algorithms, in turn, could make it possible to judge with high probability how uniform, unpredictable, and unique random data are and how safe they are for use in the required tasks. Given this, more accurate verification and standardization of received random data will be possible.

To achieve the goal, the following tasks were set:

– to choose a technique for obtaining binary data from DNA for use in calculating pseudo-random and random sequences and propose a procedure for their further evaluation;

– to implement the possibility of randomness extraction from DNA based on block encryption (using a national standard based on the crypto primitive recommended by NIST);

– to perform an analysis of the stochastic properties of RS based on DNA using appropriate methods;

– to analyze statistical properties of obtained on the basis of DNA PRS and RS using standardized methods (standards and recommendations);

– to improve and implement sequence similarity assessment algorithms (DNA, PRS, and RS), verify the possibility and expediency of their practical application.

4. The study materials and methods

The object of our study is the processes and algorithms for calculating, evaluating, and comparing pseudo-random and random sequences based on DNA.

The main hypothesis of the research assumes the possibility of using DNA as a noise source for further obtaining randomness, as well as the possibility of correct quantitative and qualitative assessment of this randomness using appropriate entropy (for example, Shannon entropy, min-entropy, and collision entropy), and statistical methods.

The research assumes the sufficiency of entropy in the noise source in the form of DNA for the possibility of applying randomness extraction-type transformations in order to further obtain PRS and RS with the required properties. An assumption is also adopted about the possibility of a sufficiently accurate representation of the sequence in the form of a set of substrings or their hashes.

AIS 20/AIS 31 [2], NIST 800-90A [19], and NIST 800-90B [4] were selected as the basic standards and recommendations for this study, which define requirements for the construction, evaluation, and improvement of generators of various types and species, as well as requirements for noise sources and their entropy.

In [2], the distribution of generators by classes of functionality is provided. The paper covers DRNGs (deterministic RNGs), PTRNGs (physical true RNGs), and NPTRNGs (non-physical true RNGs). Functionality classes formulate the general requirements that an RNG must meet. The requirements are technologically neutral, so they leave room for new designs. This will contribute to research and new developments in this field [2].

In addition to the functionality class requirements, there are other aspects and features that also affect the security of random number generation, such as choosing the right crypto primitive to build the generator. A detailed description and justification of the requirements for such crypto primitives is provided in [19].

Requirements for noise sources and their entropy, which are defined in [4], are also important for the construction of generators. An integral part of obtaining randomness is its correct quantitative and qualitative assessment. To this end, it is implied to use the estimation of the values of various types of entropy (min-entropy, Shannon entropy, collision entropy) and statistical testing of random sequences at the output of generators. A detailed description of assessment methods is reported in [20].

When evaluating the similarity of sequences, research and application of k-mer counting and k-mer distance calculation were chosen as the main direction [16, 17]. The main advantage of such methods over other comparison methods is the simplicity of implementation due to the absence of the need for preliminary alignment of sequences for correct evaluation.

Thus, the following methods were used to perform research:

– application of the basics of bioinformatics and DNA mathematics to represent DNA sequences in binary form;

– extraction of randomness using a secure block cipher in CTR mode to obtain PRS and RS;

– evaluation of statistical properties of samples and sequences at the output of the generator using a set of tests [5];

– estimation of stochastic (entropy) properties of both noise source in the form of DNA and RS at the output of the generator using min-entropy [4], Shannon entropy, and collision entropy;

– comparison of DNA sequences, PRS and RS based on them using the k-mer distance and MinHash distance calculation methods.

The following software tools were used during development and testing:

– universal, procedural, imperative general-purpose programming language C (United States of America);

– high-level general-purpose programming language C++ (United States of America);

– programming language for statistical calculations and data visualization R (New Zealand);

– symmetric block transformation algorithm DSTU 7624:2014 "Kalyna" (Ukraine) (analog: AES (United States of America));

– cryptographically protected Linux system generator (Finland) /dev/random [21];

– software implementation of a set of NIST tests: "NIST STS" (United States of America) and "IIT. Statistical testing of PRS" (Ukraine);

– software implementation of the entropy assessment tool according to [4]: SP800-90B_EntropyAssessment (United States of America);

– software implementations of Shannon entropy and collision entropy evaluation tests (Ukraine);

– the hash function DSTU 7564:2014 "Kupyna" (Ukraine) (analog: SHA (United States of America)).

5. Results of research into the possibility of using DNA cryptography in the field of randomness

5. 1. Technique for obtaining binary data from DNA for use in the calculation of pseudorandom and random sequences and the procedure for their further evaluation

The initial data for solving this task is DNA, which is proposed to be used as a NS for obtaining samples. Solving the task involves:

– obtaining various DNA samples from relevant databases and DNA banks;

– selection of the technique for their representation in the form of binary data for further use in the process of calculating PRS and RS;

– providing proposals regarding the order of their evaluation.

As a result of solving the task, the possibility of obtaining DNA samples in binary form is assumed, as well as the possibility of determining further actions with these samples based on the obtained evaluation values.

At its lowest level, DNA is made up of four types of nitrogenous bases: adenine (A), cytosine (C), guanine (G), and thymine (T). For the possibility of using DNA in solving the tasks, in particular, in the generation of PRS and RS, such a sequence must be represented in binary form. To this end, a technique was chosen that involves replacing these nitrogenous bases with the corresponding binary combinations. The representation of DNA in the form of a coding sequence of nitrogenous bases is shown in Fig. 1.

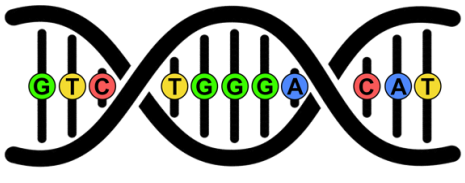


Fig. 1. Deoxyribonucleic acid (coding sequence) in the form of nitrogenous bases

To uniquely map the DNA alphabet into a binary alphabet, one must use combinations of length 2, since $2^2=4$, where 4 is the power of the DNA alphabet, 2 is the power of the binary alphabet, and the power is the length of the combinations to uniquely map the larger alphabet into the smaller one.

Thus, the representation follows the rules given in Table 1. Such binary combinations have been proposed because in this representation the nitrogenous bases are arranged alphabetically (A, C, G, T), and the values are in ascending order, respectively (0, 1, 2, 3). So, the earlier the combination occurs in the alphabet, the smaller the value it will correspond to.

DNA from the example in Fig. 1 (GTCTGGGACAT) after representation in binary form according to the rules from Table 1 will take the following form – 10110111101010000010011. The described rules make it possible to obtain binary sequences from DNA sequences of any length. It should be noted that the length of the binary sequence will be twice as long.

Table 1

Binary combinations to represent the DNA alphabet

Nitrogenous base in DNA	Binary combination to represent it
A	00
C	01
G	10
T	11

DNA sequences of various organisms can be obtained at the website of the National Library of Medicine (United States of America) in the gene bank [22]. There one can perform a search both by the organism as a whole and by a specific gene. At the same time, the DNA bank of Japan (Japan) [23] implements a more convenient search, which makes it possible to sort the results by organism, molecular type, DNA sequence length, date of receipt, etc. Each of the above services has advantages and disadvantages. DNA samples for calculating PRS and RS were obtained from [23] since it is easier to find DNA of the required length there. Samples for similarity calculation were taken from [22], as specific genes of organisms were chosen for this purpose.

Our paper reports the results of research only for human DNA sequences since the results for other organisms are similar, which additionally indicates the same nature of the origin of DNA, and therefore the possibility of considering DNA as a non-physical noise source. Also, as already noted above, DNA is unique and PRS or RS obtained on its basis can be used as unique data. So, having received a sample of one's own DNA, a person can use it to obtain, for example, unique secret keys, etc.

Human DNA, namely chromosome 14 from Homo sapiens CHM13, was used as the main DNA sequence in this study. This sequence is available in [22, 23] under the ID CP068264. The length of this DNA sequence is 101161492 nitrogenous bases. Fig. 2 shows a fragment of this sequence.

From the selected human DNA according to the rules from Table 1 we received binary data (a sequence in .dat format with binary data) for further use when solving the set tasks. When calculating PRS and RS, from the received binary sequence 3 fragments of 1 MB length will be selected from random areas in DNA.

To use sequences, it is necessary to be able to evaluate their properties. Depending on the obtained values, the sample can either be immediately used as random data, or it may require improvement using, for example, extraction [6], or it is rejected as completely non-random. This study proposes a properties estimation algorithm that is relevant for any binary data. The execution of the algorithm involves carrying out stochastic and statistical testing of sequences as follows:

```
252060  tctctaaagatttaggaaaatggccgggcacagtggctcatgcctgtaattccaacactt
252061  tgggagggtaaggcaggaggatcccttgagcccaggaatttaaggctgtattgagctgtg
252062  atcacaccactatactccagcctgggtgacagagtctctgtctcaaaaaataaaataata
252063  gtaataaataataaatttataaaatgccagccccctccatgaggcctcttttctccccac
252064  atatgatggtttttggcacagcagtgatggggagaaagagggcgagctgccagcgagag
252065  ctggactcttatggaatcccagcctgagcacgtgctatggagtctggtttctctccctga
252066  gctgggatcactgttagcctacaggatcagtgcagagtcctggcgtttcttgacctgtct
252067  gaagtgtgcagtctgccctctttatcagaatttcctggccagctacatgctgaggtctga
252068  ctaggaaaataaagcactcaggggttatggtaccccaccaagcacaaaggccactgaagt
252069  gcctggtcccattacctgctctttctttcctttctgcgcttgcttggcaaataaaagcca
252070  gcagtctgagaggccttcctgccggttactcctgcctttgtttgcaggccgtgcctggct
252071  gtacctggccctcaacgagaactccttggagagctacctgcggttgttccaggagaacct
```

Fig. 2. Fragment of Homo Sapiens DNA sequence

EvaluateSeq algorithm.
Input:
A binary sequence (in .dat, .bin, etc.) whose properties one wants to investigate.
Output:
The value of min-entropy estimates by appropriate methods;
The value of Shannon entropy estimates by appropriate methods;
The value of collision entropy estimates by appropriate methods;
Statistical test values from the NIST STS.
Or
Statistical test values from the NIST STS:
1. Determination of the type of sequence under investigation
2. Carrying out necessary assessments
if (pseudo-random sequence)
Evaluation of statistical properties according to the recommendations of NIST 800-22 [5]
else
Entropy estimation (min-entropy according to NIST 800-90B [4],
Shannon entropy according to [24] and collision entropy according to [25])
if (entropy values meet the requirements)
Evaluation of statistical properties according to the recommendations of NIST 800-22 [5]
else
Application of randomness extractor
end if
3. Derivation of the obtained values of the estimates.

For a clearer representation of the proposed algorithm for evaluating the properties of the sequences, Fig. 3 shows its block diagram.

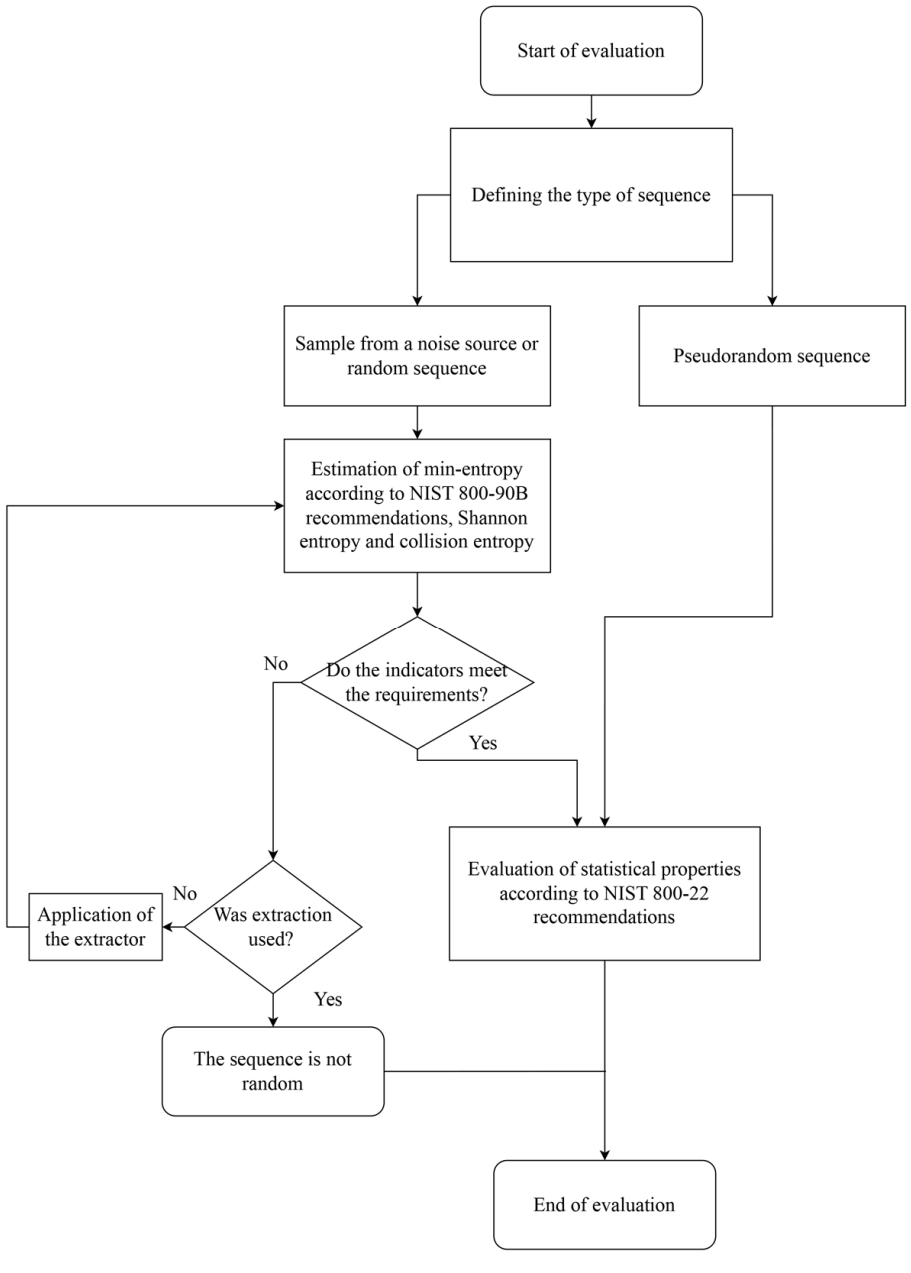


Fig. 3. Block diagram of the algorithm for evaluating sequence properties

Fig. 3 demonstrates that the verification of stochastic properties includes the evaluation of three types of entropy: min-entropy, which is described in detail in [4], Shannon entropy, and collision entropy. Estimating min-entropy involves the use of many methods. Each of the evaluation methods in [4] uses one of two approaches. The first approach is based on entropy statistics, and the second is based on predictors. In general, the following estimations are provided: most common value, collisions, Markov, compression, t-tuple, longest repeated substring, most common values in window prediction, lag prediction, MultiMMC, and LZ78Y. To estimate the Shannon entropy and collision entropy, correction methods are used to obtain the unshifted entropy obtained by the corresponding formulas for their estimation.

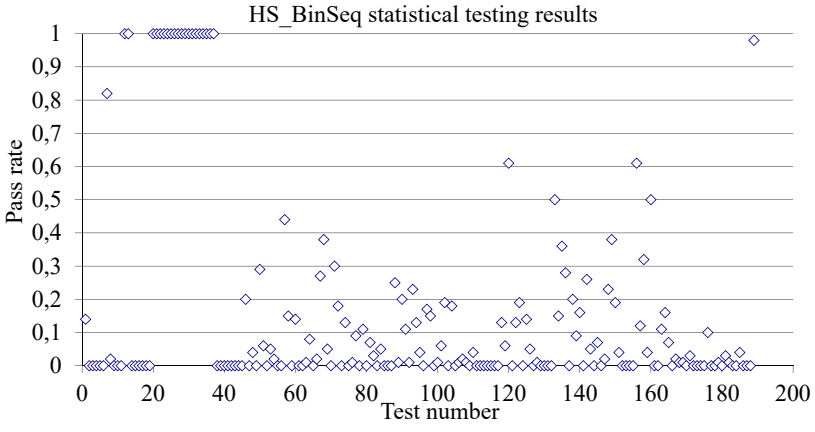


Fig. 4. A statistical portrait of sequence from Homo Sapiens DNA

The evaluation of statistical properties involves the use of a set of tests from [5]. The set contains 16 statistical tests, which are designed to test the hypothesis about the randomness of binary sequences of arbitrary length. The tests contained in the set can be found in detail in [5]. In fact, depending on the input parameters, 189 probability values are calculated, which can be considered as the result of the work of individual tests. All tests are aimed at detecting various defects of randomness. The set makes it possible to determine to what extent the sequence generated by the studied primitive is statistically safe.

Thus, the proposed evaluation algorithm makes it possible, depending on the type of the studied sequence, to obtain the values of its entropy and statistical indicators. Based on these assessments, it is possible to determine a further action plan for the sequence (acceptance, further improvement, or rejection).

The results of stochastic and statistical testing of the "raw" sequence from Homo Sapiens (human) DNA are given in Table 2, and in Fig. 4, respectively. A sequence fragment of 1 MB length (in binary representation) is used to check entropy indicators, and a fragment of 13 MB length (in binary representation) is used for statistical testing.

Table 2

Results of sequence from Homo Sapiens DNA testing using entropy methods

Estimating the min-entropy using [4]		Estimating Shannon entropy by methods from [24]	
MCVBitTest	0.992654	MaxLikehood	0.968398
CollisionTest	0.791030	MM	0.968413
MarkovTest	0.884869	UnveiIJ	0.968422
CompressionTest	0.328646	BUB	0.968413
MMCinWindowBitTest	0.273780	Bayes a=1.0	0.968413
LagPredictionBitTest	0.038148	Bayes a=1/k	0.968398
MultiMMCPredictionBitTest	0.179322	Bayes a=sqrt(n)/k	0.968460
LZ78YPredictionBitTest	0.202291	CS	0.968205
TupleTest	0.032402	SHR	0.968434
LRSTest	0.000022	SHU	0.968420
Estimating collision entropy using [25]			
Naive		0.936559	
Unbiased block=1		0.936590	
Unbiased block=10		0.870314	

Our results indicate that the statistical properties of such sequences do not meet the requirements – almost all statistical tests fail. This may be a sign of significant deviation of DNA sequences from uniformity, which is explained by their structure and features.

The results of stochastic testing indicate the presence of some entropy in the sequences, and therefore it is possible to try to perform randomness extraction, for example, by applying block symmetric transformations.

5. 2. Implementing the possibility of randomness extraction from DNA based on block encryption

The initial data for solving this task are samples obtained from DNA in binary representation. Solving the given task involves the application of extraction to these samples using block symmetric encryption. As a result of solving the problem, the PRS and RS with the necessary stochastic and statistical properties calculated by the implemented algorithms are assumed.

The generation of PRS and RS in our work is performed using the extractor implementation based on the DSTU 7624:2014 block symmetric encryption standard in CTR mode. The use of such a cryptographic transformation to calculate PRS and RS is substantiated in NIST 800-90A [19] and NIST 800-90B [4].

To generate sequences using the standard, the input data are the encryption key, the nonce, and the plaintext that will be the basis. To calculate PRS, the standard suggests setting the key and nonce as a hash of the internal state of the generator, and for calculating RS – the use of random strings of the appropriate length. Such strings can be obtained using, for example, trusted PTRNGs (Quantis QRNG (Switzerland) [26], Gryada-3 (Ukraine) [27]), or NPTRNG [22]. The structural diagram of the sequence calculation process using the extractor is shown in Fig. 5.

The ComputePRS algorithm makes it possible to calculate PRS of a given length based on binary fragments of DNA sequences (or any binary data, formats .dat, .bin, etc.) and a given key and nonce. The initial values of keys, nonce, and fragments of DNA sequences for calculating PRS are given in Table 3.

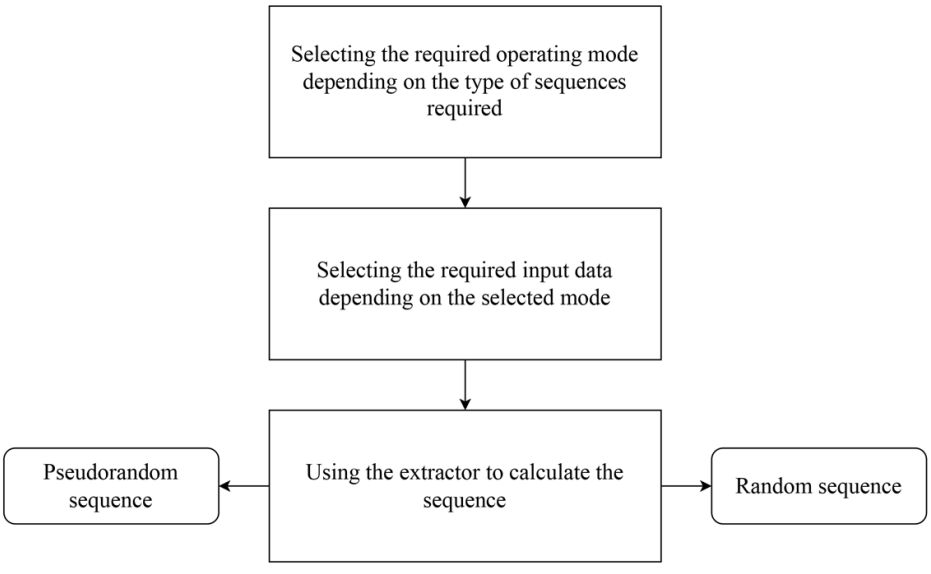


Fig. 5. The process of obtaining pseudorandom and random sequences

Table 3

Value of the input data of the PRS calculation algorithm

Key (256 bits)	Nonce (256 bits)	A fragment of DNA in binary form, taken as a basis (length 1MB)
0xFF, 0x06, 0x21, 0xA6, 0xA0, 0x28, ..., 0x02, 0x4D	0x73, 0xA4, 0x12, 0xEA, 0xD0, 0x45, ..., 0xFE, 0x3F	0xD5, 0x76, 0x48, 0xF7, 0x10, 0x02, 0x2E, 0xFD, ...
		0x80, 0xFA, 0x43, 0x10, 0xF2, 0x2F, 0xD1, 0x32, ...
		0x51, 0x4E, 0x7C, 0x9C, 0x3F, 0xFB, 0x3F, 0xF2, ...

The developed algorithm for obtaining pseudorandom sequences based on DNA is given below:

ComputePRS algorithm.
Input:
A file with a fragment of DNA sequence (in binary representation, in .dat, .bin, etc. format) that will be used as a basis – 1 MB in length;
The key (in HEX representation) is 256 bits long;
Nonce (in HEX representation) is 256 bits long.
Output:
A file with pseudo-random sequence (in binary format, .dat, .bin, etc.) of a chosen length (multiples of 1 MB) based on the provided DNA:
1. Setting the size of the key, nonce, and plain/encrypted text (in bytes) and initializing the block cipher
#define KEY_SIZE 32;
#define NONCE_SIZE 32;
#define PT_SIZE 32;
#define CT_SIZE 32;
ctx=KalynaInit;
2. Initialization of the buffer into which the DNA sequence will be read (a fragment of the required size, for example, 1MB)
rawSeq[1MB/8];
3. Reading the DNA sequence in binary form
rawSeq[i]=bytes[i – i+8];
4. Calculation of the sequence of the selected length
while (selected sequence length not reached)
Key and nonce initialization
if (first iteration of encryption)
Setting the key and nonce via input data
else
initKey=internal state of the generator;
nonce=internal state of the generator;
uint64_t initKey [KEY_SIZE/8]=Hash32(initKey | 0x01);
uint64_t nonce [NONCE_SIZE/8]=Hash32(nonce | 0x02);
end if
Encrypting the initial/already encrypted fragment
KalynaKeyExpand;
counter=0;
for (i < rawSeqLen; i+=PT_SIZE/8)
pt [PT_SIZE/8]={nonce[i] ... nonce[i+PT_SIZE/8-1]};
pt=pt^counter;
counter++;
KalynaEncipher(pt, ctx, ct);
ct[i] ... ct[i+CT_SIZE /8-1]=ct[i...i+...]^rawSeq[i...i+...];
rawSeq [i] ... rawSeq [i+CT_SIZE/8-1]=ct[i] ... ct[i+...];
end for
Write the encrypted fragment to the output file
end while.

Using this algorithm, PRSs based on human DNA were obtained, which will be used for further testing.

Next, the algorithm for obtaining RS is given. In order to comply with the requirements of DSTU 7624:2014 regarding generation, the key and nonce for each encryption iteration are obtained using the /dev/random generator [22] on the kernel of the operating system version 5.4, for which this generator according to [2] belongs to the NTG.1 class, i.e., is non-physically true generator. DNA fragments for obtaining RS are the same as for PRS (Table 3):

ComputeRS algorithm.
Input:
A file with a fragment of DNA sequence (in binary representation, in .dat, .bin, etc. format) that will be used as a basis – 1 MB in length;

Output:

A file with random sequence (in binary format, .dat, .bin, etc.) of a chosen length (multiples of 1MB) based on the provided DNA.

1. Setting the size of the key, nonce, and plain/encrypted text (in bytes) and initializing the block cipher

#define KEY_SIZE 32;
#define NONCE_SIZE 32;
#define PT_SIZE 32;
#define CT_SIZE 32;
ctx=KalynaInit;

2. Initialization of the buffer into which the DNA sequence will be read (a fragment of the required size, for example 1MB)

rawSeq[1MB/8];

3. Reading the DNA sequence in binary form

rawSeq[i]=bytes[i – i+8];

4. Calculation of the sequence of the selected length

while (selected sequence length not reached)
Key and nonce initialization
file=open("/dev/random", "r");
read(file, initKey, KEY_SIZE);
read(file, nonce, NONCE_SIZE);
Encrypting the initial/already encrypted fragment
KalynaKeyExpand;
counter=0;
for (i < rawSeqLen; i+=PT_SIZE/8)
pt[PT_SIZE/8]={nonce[i] ... nonce[i+PT_SIZE/8-1]};
pt=pt^counter;
counter++;
KalynaEncipher(pt, ctx, ct);
ct[i] ... ct[i+PT_SIZE/8-1]=ct[i...i+...]^rawSeq[i...i+...];
rawSeq [i] ... rawSeq [i+PT_SIZE/8-1]=ct[i] ... ct[i+...];
end for
Writing the encrypted part to a file
end while.

This algorithm is similar to the previous one but owing to the use of a non-physically true generator, it makes it possible to obtain random sequences.

This technique has worse performance (the degree of deterioration depends on the performance of the TRNG used) due to the need to generate new keys and nonces using a third-party generator at each iteration. However, this provides better protection against cryptanalysis. This is explained by the fact that when these values are compromised, the attacker will be able to forge and reproduce the data that was generated only on this iteration from the current state. At the same time, s/he will not be able to calculate the previous and next values of the key and nonce, which, for example, is theoretically possible when using a hash to calculate the next state, which means that s/he will not be able to compromise the data obtained on their basis.

With the use of such an algorithm, RS based on human DNA were obtained.

5. 3. Analyzing stochastic properties of the obtained random sequences using appropriate methods

The initial data for solving this task are RS obtained by applying extraction. Solving the given problem involves the use of the presented tests of entropy indicators (Shannon entropy, min-entropy, and collision entropy). The calculated values of the min-entropy, Shannon entropy, and collision entropy indicators of RS obtained on the basis of the DNA and decision-making about the success of the testing and the quality of their randomness are expected as the result of solving the problem.

Next, the results of the study of stochastic properties of RS, obtained using the ComputeRS algorithm, are reported. The proposed algorithm (Fig. 3) provides for the estimation of entropies: min [4], Shannon [24], and collision [25].

According to [4], stochastic research has two options: IID testing and non-IID testing, where IID means testing independent and identically distributed variables. Since in this case the sequences at the output of the extractor are independent and identically distributed, the estimation of min-entropy will be carried out by means of IID-testing. The value of Shannon entropy and collision entropy will be estimated as a minimum value based on the results of all possible test options to evaluate them.

The results of RS testing by entropy methods are given in Table 4. Testing was performed on data in binary file format (.dat) with 13 MB (13,631,488 bytes or approximately 10^8 bits) and 120 MB (125,829,120 bytes or approximately 10^9 bits) sequences, as specified in the Sequence column.

Table 4

Results of RS testing using the entropy methods

Sequence	Min-entropy	Shannon entropy	Collision entropy
13MB_RandSeq1	0.9959725	0.999983	0.999996
13MB_RandSeq2	0.9959991	0.998023	0.999997
13MB_RandSeq3	0.9962625	0.998302	0.999996
120MB_RandSeq1	0.9983775	0.994104	0.999998
120MB_RandSeq2	0.9986189	0.999989	0.999998
120MB_RandSeq3	0.9986769	0.999989	0.999998

The results indicate that the entropy indicators of the obtained RS fully meet the necessary requirements of unpredictability. It should also be mentioned that the sequences not only pass IID tests for min-entropy, but also all others, such as, for example, the chi-square test [28], which indicates a high quality of their randomness.

Since the obtained sequences successfully pass the stochastic testing, the next step of the evaluation algorithm is to evaluate the statistical properties using the NIST test suite [5]. Statistical testing was performed for both RS and PRS.

5. 4. Analyzing statistical properties of the obtained pseudo-random and random sequences using standardized methods

The initial data for solving this task are PRS and RS obtained by applying the extraction. Solving the given task involves the application of statistical tests from the NIST STS [5] to these sequences. As a result of solving the problem, the calculated values of the pass rates of the tests from the set, as well as judgments about the success of the testing based on these values, are assumed.

The results of statistical testing are given in Tables 5, 6. The tables contain the following data: length

and sequence identifier; the number of tests from [5] for which more than 96 % of the bitstreams passed the test successfully (>96 %); the number of tests for which more than 99 % of the bitstreams passed the test successfully (>99 %); the number of tests for which the value of p<0.001 (p<0.001); the status of passing the test (Status).

Testing was performed on data in binary file format (.dat) with 13 MB (13,631,488 bytes or approximately 10^8 bits) and 120 MB (125,829,120 bytes or approximately 10^9 bits) sequences, as specified in the Sequence column. The number of bit streams for each sequence is 100.

The tables above demonstrate that the results of testing some sequences for the 96 % criterion (13MB_PseudoRandSeq1, 13MB_RandSeq2, and 13MB_RandSeq3) indicate that not all tests from the set were passed. These results are due to the fact that the pass rate value for one of the set tests for these sequences is slightly less than the limit recommended by NIST for this number of bitstreams.

NIST in [5] puts forward several proposals for the interpretation of the obtained test results. Interpretation of empirical results involves different ways. Two approaches adopted by NIST include examining the proportion of sequences that pass a statistical test and the distribution of P-values to test for uniformity. In the case that none of these approaches work, additional experiments should be conducted on different samples of the generator to determine whether this phenomenon was a statistical anomaly or clear evidence of non-randomness [5].

Thus, the failure of some sequences of one test from the set can be considered a statistical error since the results for other sequences, in particular for larger sizes (which give a more accurate estimate), indicate a complete success of the testing. It should also be taken into account that NIST in [5] indicates the approximation of this limit.

Additionally, as indicated above in [5], attention should also be paid to the statistics of P-values, in particular for failed tests. For example, for the failed test for 13MB_PseudoRandSeq1, which was a Non-Overlapping Template Matching test, the pass rate is 0.9541 and the P-value is 0.7887, which is within the specified limits (greater than the chosen significance level of 0.001). This also applies to other sequences for which one test has failed. The pass value for one of the Non-Overlapping Template Matching Tests for 13MB_RandSeq2 and 13 MB_RandSeq3 is also 0.9541, with P-values of 0.2826 and 0.0830, respectively. This additionally indicates the presence of a statistical anomaly.

Also, an additional factor that may moderate the judgment of non-randomness based on the failure of the Non-overlapping Template Matching test itself is the number of results of this test in the overall statistic. The total result contains 148 values of this test, and therefore it can be assumed that the value of the pass rate in general can be interpreted as the average value for all these 148 tests. If the pass rate of one of the tests deviates slightly from the limit, it will still mean compliance according to the given criterion.

It is also important that when the sequence length is 10^9 bits, such statistical errors are not observed for the 96 % level. This may indicate that the length of 10^8 bits is insufficient to accurately perform such testing.

The statistics for the number of tests in which more than 96 % of the bitstreams passed the test are not different for sequences of different lengths, but for the 99 % pass rate, longer sequences show worse results. Despite this, the test pass rates for longer sequences have a more dense and uniform distribution, which can be observed in the statistical portraits in Fig. 6, 7.

The results of the statistical study indicate that the PRS and RS from DNA at the output of the extractor meet all the necessary requirements. Therefore, the PRSs obtained using the extractor successfully pass statistical testing, and the RSs pass both stochastic and statistical testing. Therefore, sequences obtained using the extractor have good properties of unpredictability and uniformity. This indicates the possibility of applying such sequences in the required tasks.

Table 5

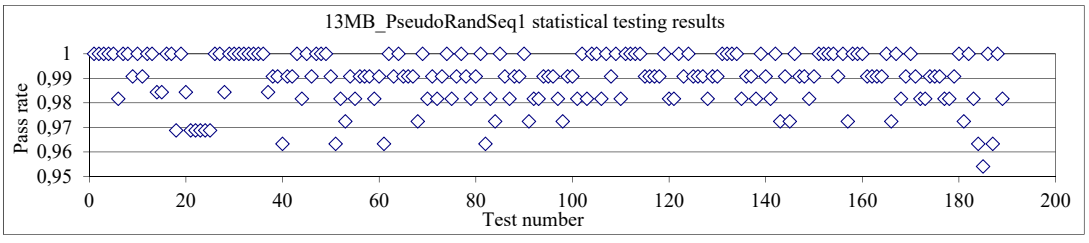
Results of statistical testing of PRS

Sequence	>96 %	>99 %	p<0.001	Status
13MB_PseudoRandSeq1	188 (99 %)	129 (68 %)	0	Success
13MB_PseudoRandSeq2	189 (100 %)	127 (67 %)	0	Success
13MB_PseudoRandSeq3	189 (100 %)	137 (72 %)	0	Success
120MB_PseudoRandSeq1	189 (100 %)	112 (59 %)	0	Success
120MB_PseudoRandSeq2	189 (100 %)	95 (50 %)	1	Success
120MB_PseudoRandSeq3	189 (100 %)	98 (52 %)	1	Success

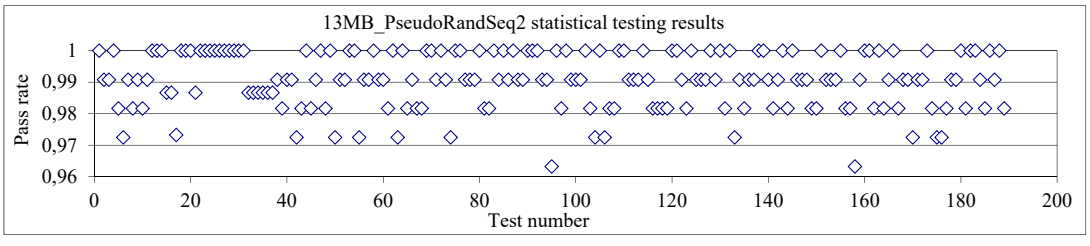
Table 6

Results of statistical testing of RS

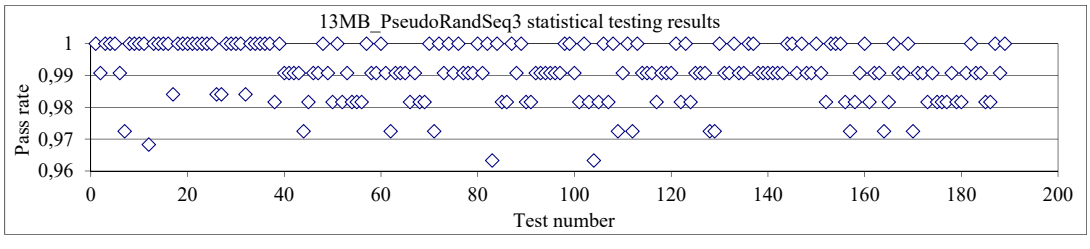
Sequence	>96 %	>99 %	p<0.001	Status
13MB_RandSeq1	189 (100 %)	132 (70 %)	0	Success
13MB_RandSeq2	188 (99 %)	118 (62 %)	0	Success
13MB_RandSeq3	188 (99 %)	124 (66 %)	0	Success
120MB_RandSeq1	189 (100 %)	94 (50 %)	1	Success
120MB_RandSeq2	189 (100 %)	113 (60 %)	0	Success
120MB_RandSeq3	189 (100 %)	104 (55 %)	1	Success



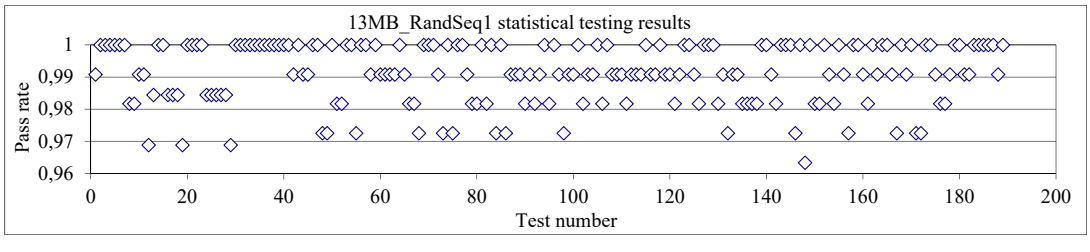
a



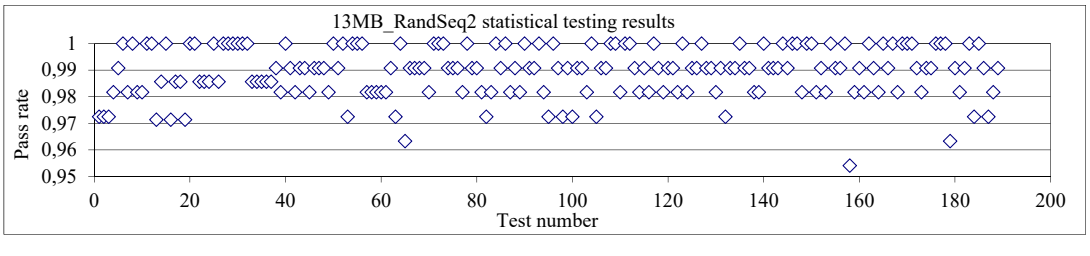
b



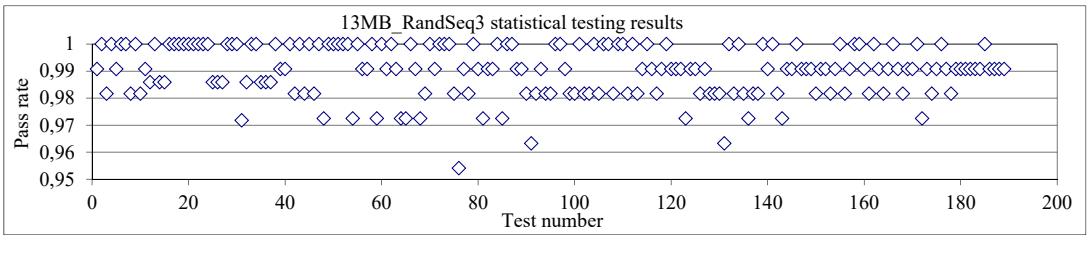
c



d

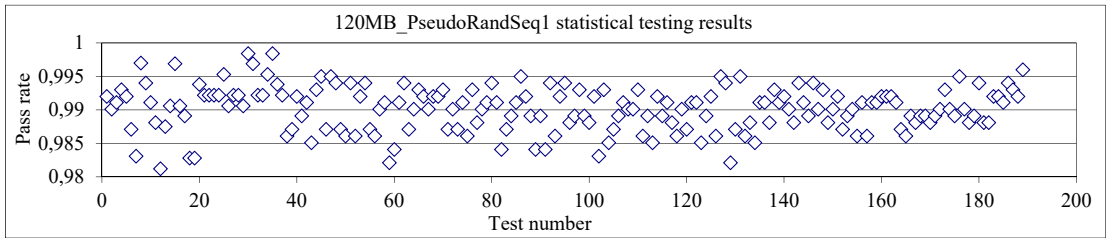


e

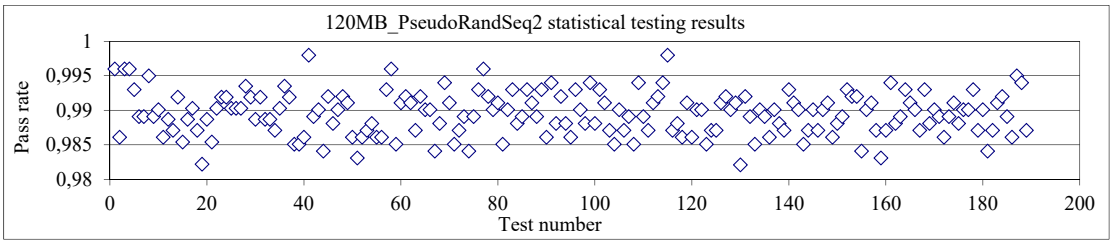


f

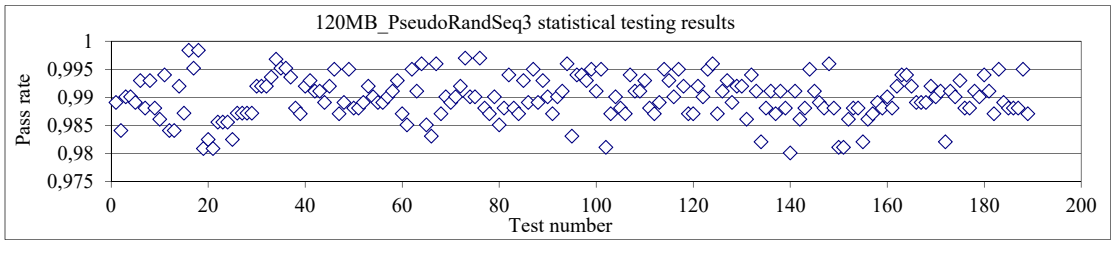
Fig. 6. Statistical portraits of pseudorandom and random sequences of length 13 MB: *a* – 13 MB_PseudoRandSeq1, *b* – 13 MB_PseudoRandSeq2, *c* – 13 MB_PseudoRandSeq3, *d* – 13 MB_RandSeq1, *e* – 13 MB_RandSeq2, *f* – 13 MB_RandSeq3



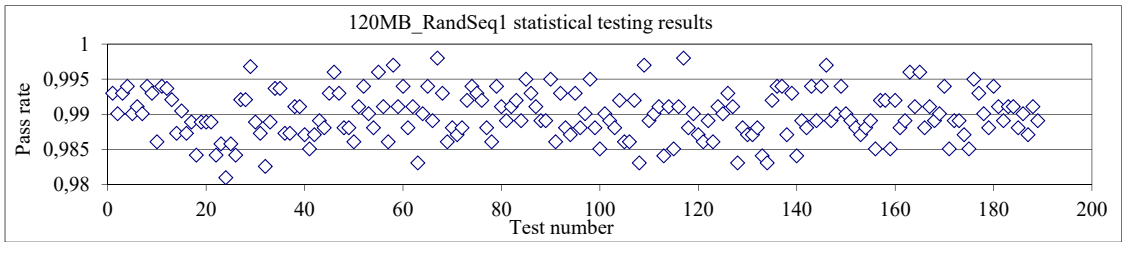
a



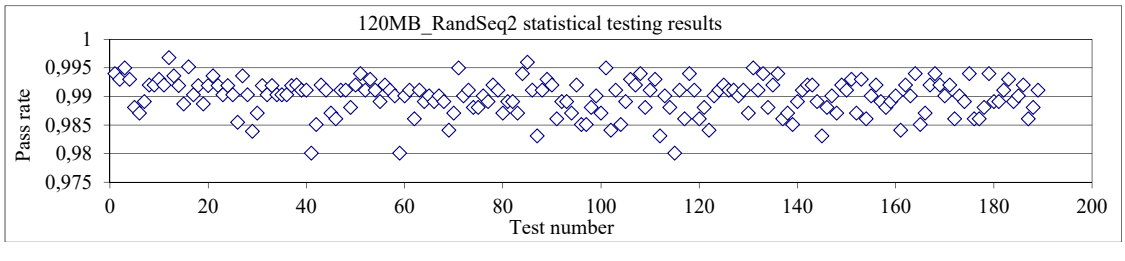
b



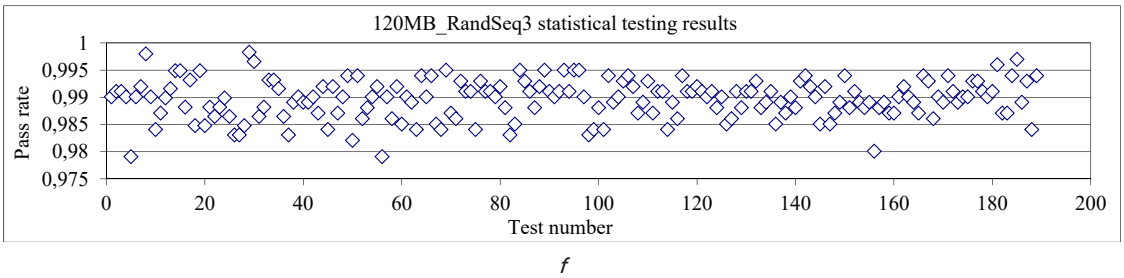
c



d



e



f

Fig. 7. Statistical portraits of pseudo-random and random sequences of length 120 MB: *a* — 120MB_PseudoRandSeq1; *b* — 120 MB_PseudoRandSeq2; *c* — 120 MB_PseudoRandSeq3; *d* — 120 MB_RandSeq1; *e* — 120 MB_RandSeq2; *f* — 120 MB_RandSeq3

5. 5. Improvement and implementation of sequence similarity assessment algorithms, verification of the possibility and expediency of their practical application

The initial data for solving this task are existing algorithms for evaluating the similarity of sequences, which have certain shortcomings in efficiency. Solving the given task involves eliminating the shortcomings by using more optimized structures and data types that give an advantage in the use of memory resources, and universal crypto primitives that can work in different modes. As a result of solving the problem, algorithms improved in terms of efficiency are provided, which are capable of saving resources or have extended functionality.

A simple and effective way to assess sequence similarity without alignment is the method based on *k*-mer counting. The main application of this method was in the field of DNA, however, our paper attempts to use it when comparing binary PRS and RS. In bioinformatics, *k*-mers are substrings of a biological sequence of length k [29]. Relative to bit, byte, or other strings, *k*-mer will be substrings of bits, bytes, or characters.

As an example of counting *k*-mers in the DNA sequence, Table 7 will give all possible *k*-mers of length from 1 to 6, for a short DNA fragment of the sequence GTCAGC.

Table 7

Values of possible *k*-mers of different lengths for a DNA sequence fragment

Sequence: GTCAGC	
Length of *k*-mer	*k*-mers
1	G, T, C, A, G, C
2	GT, TC, CA, AG, GC
3	GTC, TCA, CAG, AGC
4	GTCA, TCAG, CAGC
5	GTCAG, TCAGC
6	GTCAGC

Also, for a more visual demonstration, Fig. 8 provides a graphical representation of the division of the sequence proposed as an example into *k*-mers of length 3.

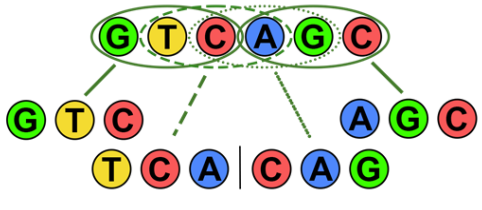


Fig. 8. Splitting a DNA sequence fragment into *k*-mers of length 3

Splitting into *k*-mers makes it possible to analyze a set of fragments of a certain size, rather than the entire sequence as a whole. The effectiveness of such a method is further substantiated by the speed and simplicity of operations with sets, as well as the availability of many algorithms and methods for working with them.

To correctly choose the length of *k*-mers, it is important to rely on the length of the sequences and their alphabet. In [17] it is proposed to determine the required size from the following formula:

$$p=\frac{1}{\frac{\left(\bar{\Sigma}\right)^{k}}{g}+1}<0.01, \quad (1)$$

where g corresponds to the size of the genome or sequence, and Σ is the alphabet of the sequence.

However, this formula is more suitable when applied to the binary alphabet or when using the MinHash algorithm. When comparing DNA using *k*-mer distances, it is possible to choose the length of *k*-mers so that their total space is larger than the size of the studied genome:

$$\left(\bar{\Sigma}\right)^{k}>g. \quad (2)$$

Evaluation of the similarity of two sequences by the proposed algorithm involves the following steps:

KmerDistCount algorithm.

Input:

Two sequences (DNA, PRS, RS in text file format .fasta, .fna, etc. for DNA/ in binary file format .bin, .dat, etc. for PRS and RS) for which a similarity score should be performed.

Output:

Calculated *k*-mer distance, by which similarity is determined:

1. Decomposing the compared sequences into *k*-mers (the length is calculated using formula (1) or (2)).

2. Representation in binary form (according to the rules in Table 1) and conversion into 64-bit numbers.

3. Creation of their union to obtain a set of unique *k*-mers.

4. Initialization of the key-value structure for sequences.

5. Setting values from the union as keys.

6. Traversing the sequences, adding +1 to the value with the key that matches the *k*-mer found in the sequence.

7. Calculation of *k*-mer distance.

The first improvement over the existing algorithm presented in [16] is the conversion of *k*-mers into 64-bit numbers (Step 2). This makes it possible to significantly simplify the comparison process since it is much easier to compare integers than strings. This is because, for example, comparing integers requires only one operation, while comparing strings requires a character-by-character similarity check, which is a long and expensive process. Also, owing to the 64-bit string representation, significant memory savings are achieved, since storing 64-bit numbers requires significantly less resources than their corresponding 32-character strings. For example, storing a single *k*-mer of length 32 as an integer would require 8 bytes of memory, while storing it as a string would already require 32 bytes. It should be noted that the implementation of Step 2 is possible under the condition that $k<33$ when applied to DNA sequences and $k<65$ when applied to binary ones, because if the values are exceeded, the bit sequences will overflow the 64-bit numbers. In cases where it is necessary to use larger sizes of k, it is possible to use simple strings instead of 64-bit numbers (as it is done in [16]).

The second improvement relative to [16] is the formation of a key-value structure using the union of unique *k*-mers that are included either in one of the sequences or in both (Steps 3 and 4). Unlike in [16], in which such a structure is created for all *k*-mers of the space of the chosen

dimensionality, the proposed approach makes it possible not to store redundant *k*-mers, that is, those that are not part of any of the sequences. This allows for a significant reduction in memory usage, since there is no need to store a huge space of *k*-mers whose number of occurrences will be equal to 0 and which will not affect the similarity calculation. For example, to compare sequences with k=31, the algorithm from [16] will create an array of 4.6^{18} elements, regardless of the length of the compared sequences, which is not possible on any modern computer. At the same time, the memory usage in the improved algorithm depends only on the length of the compared sequences. Thus, for example, when comparing DNA sequences with a length of 10,000 elements and a selected *k*-mer size of 31, the maximum possible number of unique *k*-mers will be 9,970 (sequence length minus *k*-mer size plus 1), and therefore a structure of 9,970 elements will be created. At that time, the algorithm from [16] will in any case create a structure for the full dimensionality of the space, i.e., 4^k.

Thus, the proposed improvements solve the main problem of the existing algorithm.

The calculation of *k*-mer of the distance in Step 7 takes place using the formula used in [16]. Such a formula was proposed in [30]:

$$dist(s_1,s_2)=\frac{\log(0.1+F)-\log(1.1)}{\log(0.1)}, \quad (3)$$

$$F=\sum_{Distinct.length}\frac{\min(p(s_1),p(s_2))}{\min(len(s_1),len(s_2))-k+1}, \quad (4)$$

where F is the fractional common *k*-mer count, $p(s)$ is the number of occurrences of a particular *k*-mer from the unique space to the corresponding sequence, and *len(s)* is the length of the compared sequences.

The research also considers the algorithm for comparing sequences based on MinHash. The main improvement regarding the important theoretical shortcoming of the size of the k-mers that can be used for comparison is the use of a hash function with a longer output. The national hashing standard DSTU 7564:2014 is proposed as such a tool. Existing algorithm [17] uses hash sizes of 32/64 bits for *k*-mers of length up to 32/64 for binary and 16/32 for DNA alphabet. In contrast to [17], the use of an extended size of hashes will make it possible to use significantly larger sizes of *k*-mers (up to 512 for the binary alphabet and up to 256 for the DNA alphabet), preventing possible collisions.

Such an improvement could in theory be important since the development of computer systems in the future will make it necessary to compare much larger sets of data. At the same time, for example, when dividing the sequence into substrings of more than 64 bits, the hash function, which will have a 64-bit output, will not be able to uniformly map all their possible space into the hash space. This will cause collisions, that is, different substrings will be mapped to the same hashes, which in turn will significantly reduce the accuracy of the comparison.

The proposed improvement solves this problem and, according to formula (1), makes it possible to compare data of practically infinite sizes.

The algorithm involves the following procedure:

MinHashDistCount algorithm.

Input:

Two sequences (DNA, PRS, RS in text file format .fasta, .fna, etc. for DNA/ in binary file format .bin, .dat, etc. for PRS and RS) for which a similarity score should be performed.

Output:

Calculated MinHash distance, by which similarity is determined:

1. Decomposition of the compared sequences into *k*-mers (the length is calculated using formula (1)).

For k<33 when applied to DNA and k<65 – for binary sequences:

2. Representation in binary form (according to the rules in Table 1) and conversion into 64-bit numbers.

For larger sizes of *k*-mers:

2. Representation in binary form (according to the rules in Table 1).

3. Hashing of the received values.

4. Sorting hashes in order of the increased value.

5. Selection of the necessary number of smallest hashes (sketch) to represent sequences.

6. Calculation of MinHash distance.

The more similar the sequences are, the more min-hashes they will have in common.

For hashing at Step 3 of the algorithm in this study, the "Kupyna" hash function is used. Since 8-byte hashes are enough to represent 64-bit numbers with appropriate sizes of *k*-mers, the 32-byte output of "Kupyna" will be truncated to 8 bytes. "Kupyna" is a cryptographic hash function and truncating the data to a certain size does not deteriorate their characteristics [31, 32]. Although when comparing short-length sequences, the use of larger hashes and subsequent truncation incurs performance costs, owing to this the universality of the presented algorithm is achieved.

The similarity of sets of hashes is determined using Jaccard score. The Jaccard index for sets A and B is determined from the following formula [17]:

$$jaccard(A_s,B_s)=\frac{|A_s\cap B_s|}{s}, \quad (5)$$

where A_s, B_s are sets of hashes such that the size of the intersection $|A_s \cup B_s|$ corresponds to the size of the sketch – s. The selected size affects the accuracy of the sequence display, but it depends on the number of resources used and the time of the comparison. The MinHash distance estimation error is defined as $\sqrt{\frac{1}{s}}$ [17].

MinHash distance is calculated as follows [17]:

$$mHashDist = \frac{-\log(2.0\times jaccard)/(1.0+jaccard)}{k}, \quad (6)$$

where k is the length of *k*-mers.

The presented algorithm makes it possible to quickly assess the similarity of two sequences with an accurate approximation, which is determined by the size of the sketch. At the same time, it requires significantly less resources, in particular memory, than methods with pre-alignment. Also, the algorithm gives an advantage in speed compared to the *k*-mer distance estimation method since its simplified representation in the form of a small set (sketch) of minimal hashes is used to represent the sequence.

The accuracy assessment of the presented comparison algorithms is given below. They are suitable both for evaluating DNA sequences, which in this study act as samples, and binary sequences at the output of the extractor.

For a visual demonstration of the KmerDistCount algorithm, DNA sequences from the same genes (human and chimpanzee) are compared. The INSR (insulin receptor) and VDR (vitamin D receptor) genes are used in the comparison. The length of the DNA sequences for VDR is approximately 63,000 elements, and for INSR from 181,000 to 183,000 elements. To evaluate the similarity of sequences of this size according to formula (2), it is necessary to use k-mer of length 9. For practical evaluation, the following lengths of k-mers were chosen: k=9, according to the calculation, k=5 (the default length in [16]), and k=7, 11 to more thoroughly test and track changes in accuracy with changes in k-mer size. Similarity is defined as one minus the k-mer distance.

The results of the calculation of k-mer distance by the KmerDistCount algorithm are given in Table 8.

Our values of the k-mer distance indicate that the implemented algorithm makes it possible to correctly assess DNA similarity – the assessment values are almost identical to those obtained using various evaluation tools with preliminary alignment [11–13]. The algorithm shows the most accurate estimate precisely when approaching the length of k-mers calculated for the corresponding length of the sequence. Thus, it is important to correctly calculate the length of k-mers according to the corresponding formula (2) since the accuracy of the estimate will depend on it.

To evaluate the similarity of DNA-based binary sequences, 2 sequences of different lengths were calculated: 2 MB, 6 MB, and 13 MB. The choice of the length of the sequences is due to the necessary length of k-mers for their evaluation: 31, 33, and 35, respectively. The results of the calculation of k-mer distance between RSs are given in Table 9. Lengths 29 and 37 were examined to check the change in accuracy of the estimate.

The calculated values of the k-mer distance indicate that for the length of the k-mers, which best corresponds to the length of the sequences according to the corresponding formula, the similarity of RS is estimated at 1.4–3.3 % (k-mer distance ≈0.97–0.986). This shows that the output sequences of the developed algorithms for calculating PRS and RS are almost completely unique.

Table 10 gives the results of calculating the MinHash distance for DNA sequences of various organisms using the MinHashDistCount algorithm.

Table 8

Results of k-mer distance calculation for human and chimpanzee VDR and INSR DNA genes

Gene VDR					
S1	S2	k-mer distance			
		k=5	k=7	k=9	k=11
Human DNA 1	Human DNA 2	0.0010403	0.0017395	0.0022834	0.0027967
Human DNA	Chimpanzee DNA	0.0053960	0.0215622	0.0363721	0.0462452
Gene INSR					
S1	S2	k-mer distance			
		k=5	k=7	k=9	k=11
Human DNA 1	Human DNA 2	0.0012025	0.0026962	0.0039040	0.0049289
Human DNA	Chimpanzee DNA	0.0038318	0.0181211	0.0413278	0.0573898

Table 9

Result of calculating k-mer distance for random sequences

S1	S2	k-mer distance				
		k=29	k=31	k=33	k=35	k=37
2MB_RandSeq1	2MB_RandSeq2	0.885233	0.967787	0.991931	0.997763	0.999482
6MB_RandSeq1	6MB_RandSeq2	0.731474	0.910465	0.975394	0.993671	0.998408
13MB_RandSeq1	13MB_RandSeq2	0.571947	0.828794	0.948960	0.986212	0.996457

Table 10

Result of calculating the MinHash distance for human and chimpanzee VDR and INSR DNA genes

Gene VDR						
S1	S2	MinHash distance				
		k=7	k=9	k=11	k=13	k=31
Human DNA 1	Human DNA 2	0.000719685	0.000447128	0.00055041	0.000780952	0.000630904
Human DNA	Chimpanzee DNA	0.00511872	0.00989287	0.0119934	0.0116824	0.0114459
Gene INSR						
S1	S2	MinHash distance				
		k=7	k=9	k=11	k=13	k=31
Human DNA 1	Human DNA 2	0.000143072	0.00101369	0.00101679	0.00134195	0.00133643
Human DNA	Chimpanzee DNA	0.00309769	0.0120202	0.0158604	0.0155881	0.0155836

Our MinHash distance values indicate that when k>=13, the accuracy of the estimate increases and (with a deviation within a few %) corresponds to the results obtained both by the algorithm for calculating the k-mer distance presented in the paper and by other means of comparison. The deviation in the accuracy of the estimate is explained by the use of a simplified representation of the sequence as a small set of min-hashes. In particular, the results correspond to those obtained for these sequences using the tool [17]. The main advantage of such an algorithm, despite the slightly lower accuracy (due to the use of a sketch of hashes to represent the sequence), is speed and more efficient use of computer resources. This is explained by the need to store only a small set of minimum hashes for each of the sequences.

The same sequences as in Table 9 were used to assess the similarity of RSs. The results of the MinHash distance calculation are given in Table 11.

The calculated MinHash distance values show that when using the required length of k-mers, the similarity of RSs is estimated to be approximately 82–85 %, although the Jaccard index is almost zero (0.00099–0.005). Such indicators may indicate insufficient sensitivity of the algorithm when

applied to the binary alphabet. At the same time, increasing the size of *k*-mers by one step gives an estimate of the distance between RSs of 100 %, which shows the complete difference between their sketches.

Table 11

Result of calculating the MinHash distance for random sequences

S1	S2	MinHash distance (pre-calculated Jaccard index)				
		k=29	*k*=31	*k*=33	*k*=35	*k*=37
2MB_RandSeq1	2MB_RandSeq2	0.123774 (0.014)	0.148715 (0.005)	1.00 (0.00)	1.00 (0.00)	1.00 (0.00)
6MB_RandSeq1	6MB_RandSeq2	0.086830 (0.042)	0.120698 (0.012)	0.150465 (0.0035)	1.00 (0.00)	1.00 (0.00)
13MB_RandSeq1	13MB_RandSeq2	0.062814 (0.088)	0.115789 (0.014)	0.118848 (0.010)	0.177876 (0.00099)	1.00 (0.00)

6. Discussion of results of research into the possibility of using DNA cryptography in the field of randomness

Presenting DNA as a source of randomness is possible because DNA is considered unique to each existing organism and is not repeated twice. Figure 1 shows that the DNA sequence alphabet consists of four possible values. This makes it possible to choose a way to simply replace these values with combinations according to the rules from Table 1 to get binary data.

To use such data in required applications, it is important to be able to correctly and comprehensively evaluate their properties. The research proposes a properties estimation algorithm applied to binary data. Our algorithm (Fig. 3) involves the use of modern proven methods, owing to which it is possible to clearly determine to what extent the statistical and stochastic properties of the sequences meet the requirements and whether they are sufficiently safe for use in the required tasks.

Evaluation of the DNA sample using the algorithm showed that its statistical properties do not meet the requirements (Fig. 4). This indicates a significant deviation of the DNA sequences from uniformity, which is explained by their structure and features. DNA carries information about the structure of RNA and proteins, which are usually repeated or clustered, and therefore cause the presence of patterns in the sequences. The results of stochastic testing (Table 2) indicate the presence of a certain amount of entropy in the sequences. This makes it possible to use such sequences to extract randomness and obtain PRS and RS based on them.

To solve the problems in existing works, our research proposes the implementation of an algorithm that makes it possible to obtain PRS and RS based on a small amount of input data in the form of DNA. The algorithm is implemented on the basis of the DSTU 7624:2014 block symmetric encryption standard in CTR mode. A similar crypto primitive is recommended for use as an conditioning component in NIST 800-90A. The process of obtaining and the results are determined precisely by the features of the cryptographic design of such a transformation, which makes it possible to obtain long (pseudo) random sequences based on short input data.

The scheme presented in Fig. 6 describes the process of obtaining sequences based on DNA. In contrast to [7–9], in which "raw" DNA sequences with properties that do not meet modern requirements are obtained, the developed algorithms make it possible to obtain pseudo-random and random sequences based on DNA that meet the necessary requirements through randomness extraction. Also, the advantage of the developed algorithms over existing ones is the possibility of generating random data of almost infinite length based on short input data, which cannot be achieved using [7–9]. A feature of the formation of RS, which additionally increases the security of both the algorithm itself and the received data, is the use of random session keys and nonce obtained using NPTRNG /dev/random. This provides improved protection against compromise, which is explained by the fact that at each iteration these values are generated in a completely random and independent manner and do not contain any information about previous values.

The results of RS testing by entropy methods (Table 4), as well as statistical testing of PRS and RS (Tables 5, 6) show that the obtained sequences, in contrast to works [7–9], are uniform, do not contain statistical flaws, and have high levels of unpredictability. Therefore, the implemented algorithms make it possible to eliminate the problem of non-compliance of the statistical properties of the data with the requirements and to prevent a significant weakening or compromise of cryptographic applications. This indicates the possibility of applying such sequences in the required tasks.

Our comparison algorithms make it possible to accurately approximate the similarity of DNA sequences. The obtained estimates (Tables 8–11) correspond to the estimates of tools with preliminary alignment [11–13], as well as without preliminary alignment [16, 17]. For the most accurate assessment, the size of *k*-mers should be calculated according to the size of the sequence and the alphabet according to formulas (1), (2).

The features of the proposed algorithm based on *k*-mers, owing to which it provides an advantage over the similar one in [16], are the use of 64-bit representations of *k*-mers and taking into account when counting only *k*-mers that are directly included in the sequences. Using 64-bit representations significantly simplifies the comparison process, and also provides significant memory savings, since storing 64-bit numbers requires significantly less resources. Unlike [16], in which all possible *k*-mers of the space of the chosen dimension are counted (which causes quite serious memory costs), the developed algorithm counts only *k*-mers present in the sequence. This makes it possible to compare sequences on large *k*-mer sizes. For example, to compare sequences at *k*=31, the algorithm from [16] will create an array of $4,6^{18}$ elements, which is not possible on any of the modern computers. At the same time, the memory usage in the proposed algorithm depends only on the length of the compared sequences.

A feature of the proposed algorithm based on MinHash, which provides an advantage in the size of the compared sequences, is the use of a hash function with a longer output [23]. In theory, this will make it possible to use significantly larger *k*-mer sizes, preventing possible collisions. Also, such an algorithm provides an advantage in speed and an additional advantage in the use of resources compared to the *k*-mer distance estimation method, since its simplified representation in the form of a small set (sketch) of min-hashes is used to represent the sequence.

The limitations of this study include the following:

– the applied methods of stochastic and statistical research are relevant only for the binary alphabet (and the

alphabet of the power of two), so other methods should be used to evaluate the sequences of an arbitrary alphabet, for example, from [33];

– the implemented algorithms, in particular for obtaining RS, are difficult to assess for the possibility of collisions or the period of repetition of sequences;

– the algorithm based on MinHash has insufficient evaluation sensitivity, in particular when applied to the binary alphabet.

The disadvantages of the study include the relative difficulty of obtaining unique samples of human DNA since the same samples are present in most available databases, which makes the concept of randomness impossible. This can also be said about the need to obtain DNA as a whole, which is a rather complex process that occurs separately.

A further development of our research could be a more thorough evaluation of the developed DNA-based RNG, such as proving the characteristics of forward and backward secrecy and enhanced forward and backward secrecy. Also, a possible area of further research is the improvement of comparison algorithms: an additional increase in speed, an increase in the accuracy of the assessment, etc.

7. Conclusions

1. DNA samples consist of an alphabet with 4 possible values: A, C, G, and T. The chosen technique for obtaining binary data from them involves replacing these values with the corresponding binary combinations. The use of such data requires a preliminary careful assessment of their statistical and stochastic properties. Our study proposes an evaluation algorithm that involves the use of modern proven methods of stochastic and statistical testing. Statistical evaluation of the DNA sample showed a significant deviation from uniformity, which is explained by the structure and features of DNA. The results of stochastic testing show the presence of a certain amount of entropy, which indicates the possibility of using such samples to obtain PRS and RS.

2. Extraction of randomness from DNA for the generation of PRS and RS in the study was performed using the conditioning component implemented on the basis of the block symmetric encryption standard DSTU 7624:2014. A special feature of the implemented generation algorithms is the possibility of generating pseudo-random and random data of arbitrary length with the required properties, in contrast to existing works in the area of obtaining randomness on the basis of DNA. The process of obtaining PRS and RS and the results are explained by using the NIST-recommended design (extractor), which makes it possible to calculate highly reliable data with appropriate indicators based on a short unique input.

3. The entropy indicators of RS obtained by the implemented algorithms fully meet the necessary requirements. Sequences not only pass IID tests for minimum entropy but also all others, such as, for example, the chi-square test. The peculiarity of our study is obtaining the results of estimation of not only min-entropy, as recommended by NIST, but also estimation of Shannon entropy and collision entropy. When reviewing existing works, such a comprehensive approach to stochastic estimation was not encountered, which may have critical consequences when adopting sequences for use in real problems.

4. The obtained PRS and RS successfully pass statistical testing according to the recommendations of NIST 800-22. In contrast to the considered works, tests from the set were performed in full, on data sets that meet the requirements for correct testing. Our results indicate that the sequences calculated using the extractor, in contrast to the "raw" DNA, have the desired properties of randomness and do not contain statistically significant errors or flaws. In general, the passing of all the necessary checks by sequences makes it possible to judge the possibility of their application in the necessary crypto primitives or algorithms.

5. The reported improved comparison algorithms based on k-mer and MinHash distances allow for accurate similarity assessment. A feature of the algorithm implemented on the basis of k-mers is a significant saving of hardware resources, in particular memory. Unlike existing ones, the algorithm presented in the study uses more economical data types and makes it possible to correctly assess similarity while working with much smaller arrays of information. A feature of the algorithm based on MinHash is the possibility for significantly increasing the size of the compared sequences. In contrast to existing algorithms, the algorithm implemented in our study uses a hash function with a longer output, which makes it possible to compare larger arrays of data on larger dimensions of k-mers. The presented algorithms calculate the similarity of DNA sequences with an accurate approximation, however, the evaluation of the similarity of binary sequences, especially using the MinHash distance, requires further research and development. Checking the calculated DNA-based PRS and RS using k-mer distances shows that with correctly selected k values, their similarity is minimal, and therefore the sequences are unique.

Conflicts of interest

The authors declare that they have no conflicts of interest in relation to the current study, including financial, personal, authorship, or any other, that could affect the study, as well as the results reported in this paper.

Funding

The study was conducted without financial support.

Data availability

The data will be provided upon reasonable request.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the current work.

Acknowledgments

The authors express their gratitude for the support provided during the preparation of this work, as well as for useful comments and suggestions from the research group AT "IIT".

References

1. Goldreich, O. (2001). Foundations of Cryptography. Cambridge University Press. https://doi.org/10.1017/cbo9780511546891
2. Matthias, P., Werner, S. (2023). A Proposal for Functionality Classes for Random Number Generators. Version 2.36 – Current intermediate document for the AIS 20/31 workshop. Available at: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Certification/Interpretations/AIS_31_Functionality_classes_for_random_number_generators_e_2023.pdf?__blob=publicationFile&v=2
3. Gorbenko, I., Derevianko, Y., Gorbenko, D. (2024). DNK – a source of noise and non-physical random sequences. The main principles of practical research. II International scientific and practical conference «Cyberwarfare: intelligence, defense and offensive security», 30–31. Available at: https://cyberwarfare.viti.edu.ua/assets/files/Cyberwarfare_2024.pdf
4. Turan, M. S., Barker, E., Kelsey, J., McKay, K., Baish, M., Boyle, M. (2018). NIST SP 800-90B. Recommendation for the Entropy Sources Used for Random Bit Generation. NIST. https://doi.org/10.6028/NIST.SP.800-90B
5. Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S. et al. (2010). NIST Special Publication 800-22 Revision 1a. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. Available at: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf
6. Extractors and Pseudorandom Generators (2001). Journal of the ACM (JACM), 48 (4), 860–879. https://doi.org/10.1145/502090.502099
7. Barman, P., Djilali, B., Benatmane, S., Nuh, A. (2024). A New Hybrid Cryptosystem Involving DNA, Rabin, One Time Pad and Fiestel. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.4771411
8. Zhang, Y., Liu, X., Sun, M. (2017). DNA based random key generation and management for OTP encryption. Biosystems, 159, 51–63. https://doi.org/10.1016/j.biosystems.2017.07.002
9. Siddaramappa, V., Ramesh, K. B. (2020). True Random Number Generation Based on DNA molecule Genetic Information (DNA-TRNG). Cryptology ePrint Archive, Paper 2020/745. Available at: https://eprint.iacr.org/2020/745.pdf
10. Li, Z., Peng, C., Tan, W., Li, L. (2020). A Novel Chaos-Based Image Encryption Scheme by Using Randomly DNA Encode and Plaintext Related Permutation. Applied Sciences, 10 (21), 7469. https://doi.org/10.3390/app10217469
11. Sievers, F., Higgins, D. G. (2017). Clustal Omega for making accurate alignments of many protein sequences. Protein Science, 27 (1), 135–145. https://doi.org/10.1002/pro.3290
12. Tu, S.-L., Staheli, J. P., McClay, C., McLeod, K., Rose, T. M., Upton, C. (2018). Base-By-Base Version 3: New Comparative Tools for Large Virus Genomes. Viruses, 10 (11), 637. https://doi.org/10.3390/v10110637
13. Beretta, S. (2019). Algorithms for Strings and Sequences: Pairwise Alignment. Encyclopedia of Bioinformatics and Computational Biology, 22–29. https://doi.org/10.1016/b978-0-12-809633-8.20317-8
14. Needleman, S. B., Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology, 48 (3), 443–453. https://doi.org/10.1016/0022-2836(70)90057-4
15. Shanker, K. P. S., Austin, J., Sherly, E. (2010). An Algorithm for alignment-free sequence comparison using Logical Match. 2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE), 536–538. https://doi.org/10.1109/iccae.2010.5452072
16. Wilkinson, S. (2022). Fast K-Mer Counting and Clustering for Biological Sequence Analysis. Available at: https://cran.r-project.org/web/packages/kmer/kmer.pdf
17. Ondov, B. D., Treangen, T. J., Melsted, P., Mallonee, A. B., Bergman, N. H., Koren, S., Phillippy, A. M. (2016). Mash: fast genome and metagenome distance estimation using MinHash. Genome Biology, 17 (1). https://doi.org/10.1186/s13059-016-0997-x
18. Broder, A. Z. (1997). On the resemblance and containment of documents. Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171). https://doi.org/10.1109/sequen.1997.666900
19. Barker, E., Kelsey, J. (2015). NIST SP 800-90A Rev. 1. Recommendation for Random Number Generation Using Deterministic Random Bit Generators. NIST. http://dx.doi.org/10.6028/NIST.SP.800-90Ar1
20. Holubnychiy, D. Yu., Kandiy, S. O., Yesina, M. V., Gorbenko, D. Yu. (2024). Methods and means for analysing, evaluating and comparing properties of random sequences and random numbers. Radiotekhnika, 1 (216), 30–45. https://doi.org/10.30837/rt.2024.1.216.02
21. Müller, S. (2024). Linux /dev/random – A New Approach. Available at: https://www.chronox.de/lrng/releases/v53/lrng-v53.pdf
22. Index of /genbank. Available at: https://ftp.ncbi.nlm.nih.gov/genbank/
23. DNA Data Bank of Japan. Available at: https://www.ddbj.nig.ac.jp/index-e.html
24. Contreras Rodríguez, L., Madarro-Capó, E. J., Legón-Pérez, C. M., Rojas, O., Sosa-Gómez, G. (2021). Selecting an Effective Entropy Estimator for Short Sequences of Bits and Bytes with Maximum Entropy. Entropy, 23 (5), 561. https://doi.org/10.3390/e23050561
25. Skorski, M. (2016). Improved Estimation of Collision Entropy in High and Low-Entropy Regimes and Applications to Anomaly Detection. Cryptology ePrint Archive, Paper 2016/1035. Available at: https://eprint.iacr.org/2016/1035
26. Quantis QRNG USB. Available at: https://www.idquantique.com/random-number-generation/products/quantis-random-number-generator/
27. Hardware RNG «Gryada-3». AT «IIT». Available at: https://iit.com.ua/index.php?page=itemdetails&p=3&gtype=1&type=1&id=96

28. Chi-Square Goodness-of-Fit Test. NIST Engineering Statistics Handbook. Available at: https://www.itl.nist.gov/div898/handbook/eda/section3/eda35f.htm
29. Genome Assembly Workshop 2020. UCDavis Bioinformatics Core. Available at: https://ucdavis-bioinformatics-training.github.io/2020-Genome_Assembly_Workshop/kmers/kmers
30. Edgar, R. C. (2004). Local homology recognition and distance measures in linear time using compressed amino acid alphabets. Nucleic Acids Research, 32 (1), 380–385. https://doi.org/10.1093/nar/gkh180
31. Kelsey, J. (2005). SHA-160: A Truncation Mode for SHA256 (and most other hashes). NIST. Available at: https://csrc.nist.gov/csrc/media/events/first-cryptographic-hash-workshop/documents/kelsey_truncation.pdf
32. Dang, Q. (2009). NIST Special Publication 800-107. Recommendation for Applications Using Approved Hash Algorithms. NIST. Available at: https://csrc.nist.rip/library/NIST%20SP%20800-107%20Recommendation%20for%20Apps%20Using%20Approved%20Hash%20Algorithms,%202009-02.pdf
33. Gorbenko, I. D., Alekseychuk, A. N., Kachko, Ye. G., Derevianko, Ya. A. (2024). Research into methods and algorithms for generating (pseudo) random sequences over an arbitrary alphabet. Radiotekhnika, 216, 7–29. https://doi.org/10.30837/rt.2024.1.216.01