Study of properties, applications and software implementation of the digital signature algorithm based on elliptic curves

Serhii Buchyk[1,*,†], Anastasiia Shabanova[1,†], Serhii Toliupa[1,†], Oleksandr Buchyk[1,†], Maksym Delembovskyi[2,†]

[1] Taras Shevchenko National University of Kyiv, 60 Volodymyrska St., Kyiv, 01033, Ukraine
[2] Kyiv National University of Civil Engineering and Architecture, Povitrianykh Syl ave. 31, Kyiv, 03037, Ukraine

Abstract

The development of a digital signature algorithm based on elliptic curves, which is based on the basic models of asymmetric cryptography, confirms the increased security of the authentication procedure, which in turn is the primary task of ensuring the correct functioning of the system and the introduction of updated software solutions, the construction of which is the result of a detailed analysis of state standardisation. When determining the mathematical nature of elliptic curves, it can be established that they are complex algebraic objects that create obstacles to hacking due to the impossibility of solving the discrete logarithm problem to find a base point when the order of the elliptic curve is large. Consideration of the mathematical structure of elliptic curves and their practical use in software solutions based on the modernisation and optimisation of the use of the DSTU 4145-2002 standard is of great theoretical and practical importance, which is to create an opportunity for further evaluation of the effectiveness of cryptographic methods and improvement of the random sequence generator using both physical and functional sources, which is more productive and efficient in the context of setting the initial state of random sequences.

Keywords

Elliptic curves, digital signature, key generation, hash function, performance, signature verification

1. Introduction

1.1. Standardisation of digital signature algorithms based on elliptic curves

Since elliptic curve digital signatures (ECDSA) are the most advanced and secure technology in modern cryptography, the development of their standardisation is a key factor in ensuring interoperability, security and efficiency of use in different environments and technologies.

In this paper, we focus on the analysis of DSTU 4145-2002 [1], but for a more detailed review, we present several other global standards along with their detailed description to illustrate the results of their use.

ANSI X9.62, adopted in 1999 by the American National Standards Institute (ANSI), is a key standard for ensuring the security of digital signatures in financial institutions, which discloses the use of the ECDSA algorithm and specifies the curves recommended by NIST and DSS, such as P-192, P-224, P-256, P-384 and P-521.

The main provisions of the standard, like all the others, include detailed key generation procedures, algorithms for creating and verifying digital signatures that guarantee a high level of cryptographic security, but the main difference is that ANSI X9.62 focuses on the use of high-quality random numbers, which makes it impossible to forge digital signatures.

Information Technology and Implementation (IT&I-2024), November 20-21, 2024, Kyiv, Ukraine
* Corresponding author.
† These authors contributed equally.
buchyk@knu.ua (S. Buchyk); nastiash.2003@gmail.com (A. Shabanova); alex8sbu@knu.ua (O. Buchyk); tolupa@i.ua (S. Toliupa); delembovskyi.mm@knuba.edu.ua (M. Delembovskyi)
0000-0003-0892-3494 (S. Buchyk); 0009-0008-4962-569X (A. Shabanova); 0000-0001-7102-2176 (O. Buchyk); 0000-0002-1919-9174 (S. Toliupa); 0000-0002-6543-0701 (M. Delembovskyi)

Also, for a general understanding, it is worth considering the FIPS 186-4 standard, which was established by the US National Institute of Standards and Technology (in 2013, another equally common name is the Digital Signature Standard (DSS). Unlike its predecessor, it supports not only ECDSA but also DSA and RSA, providing a wider range of cryptographic methods for various applications. FIPS 186-4 details procedures for key generation, eavesdropping protection, and random number recommendations, increasing overall system security. Its use covers government and commercial applications where a high level of security and interoperability with other cryptographic standards is required.

RFC 6090, developed by the Internet Engineering Task Force (IETF) in 2011, is part of a series of recommendations for the use of elliptic curves in cryptography for Internet applications, also supports traditional ECDSA algorithms and focuses on ensuring interoperability and reliability of implementations for secure communications and authentication on the Internet.

Brainpool Brainpool Curves, recommended by the Brainpool Group, introduce alternative elliptic curves to increase diversity and security in cryptography. Brainpool Curves are optimised for a variety of applications, including financial systems, secure communications and cryptocurrency platforms, offering an alternative to the standard NIST curves - providing additional options for developers who wish to use unique curve parameters to enhance the security of their cryptographic solutions.

Description of the algorithm for generating and verifying a digital signature based on elliptic curves in accordance with DSTU 4145:2002

1.2.1 Basics of elliptic curves

Elliptic curves [2] are defined by the equation:

$$y^2 = x^3 + ax + b \quad (1)$$

, where a and b are the coefficients that define a particular curve, provided that 4a^3+27b^2≠0, which ensures the absence of special points. Cryptography uses elliptic curves over finite fields GF(p) or GF(2^m). The field GF(p) contains p elements, where p is a prime number, and all operations are performed modulo p. The field GF(2^m) consists of 2^m elements and is an extension of the field GF(2).

1.2.2 Group of points on an elliptic curve

The points on an elliptic curve, together with a neutral element, form an abelian group whose basic operations include point addition and point doubling - the basis for elliptic curve-based cryptographic algorithms such as key generation and the creation and verification of digital signatures.

For two points $P = (x_1, y_1)$ і $Q = (x_2, y_2)$ on the curve, the sum of $P + Q$ is equal to the point $R = (x_3, y_3)$, whose coordinates are calculated using the following formulas:

$$x_3 = \lambda^2 - x_1 - x_2 \quad (2)$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \quad (3)$$

where $\lambda = \frac{y_2 - y_1}{x_2 - x_1} mod\, p$ – slope of the line that passes through the points P and Q.

For the point $P = (x_1, y_1)$, double point $2P = (x_2, y_2)$ is calculated by the formulas:

$$x_2 = \lambda^2 - 2x_1 \quad (4)$$

$$y_2 = \lambda(x_1 - x_2) - y_1 \quad (5)$$

where $\lambda = \frac{3x_1 + a}{2y_1} mod\, p$ – slope of the line that passes through the points P and Q, and at that time a – set point.

1.2.3 Algorithm for generating and verifying a digital signature

After meeting the initial conditions of the algorithm [3], the formation of a digital signature takes place in the following steps:

1. First of all, the correctness of the general parameters of the digital signature is checked in accordance with clauses 8.1-8.3 of the standard, and in case of incorrect parameters, the calculation of the digital signature is terminated.

2. The private key is checked in accordance with clause 10.2, and if the private key has any errors, the digital signature calculation is also terminated.

3. Check whether the LD number (data length) is a multiple of 16 and is greater than twice the length of the base point order. If the conditions are not met, the digital signature calculation is terminated.

The following steps are required for the ECDSA generation itself:

1. First of all, a random integer e is calculated, imposing the condition $0 < \mathrm{e} < n$.

2. Calculate the point $R = \mathrm{e}P$, where P - base point of the elliptic curve. If the coordinate $x_R = 0$, return to the step 4.

3. Calculate the value of the hash function H(T) for notification T.

4. Calculate digital signature parameters s і r:

$$s = \mathrm{e}^{-1}(H(T) + \mathrm{d}r)\, mod\, n \tag{6}$$

where $r = x_R$, but if r is zero, return to the step 4.

5. Digital signature (r, s) add to the message [4].

The digital signature verification algorithm is described as follows, subject to the above conditions, and also checks whether the identifier of the hash function used to calculate the hash value of the message is valid and complies with the established standard:

The parameter of the modular inverse of the signature part is calculated $w = s^{-1}\, mod\, n$, where s - signature part, and n - order of the base point.

After the first step, the scaling of the hash value (7) is calculated and the parameter (8) r using the parameter w:

$$u_1 = H(T) \cdot w\, mod\, n \tag{7}$$

$$n_2 = r \cdot w\, mod\, n \tag{8}$$

The last step is, based on the previous steps, to calculate a point on the elliptic curve by using a linear combination of two points on the elliptic curve to obtain a new poin V:

$$V = u_1 \cdot P + u_2 \cdot Q \tag{9}$$

where u_1, u_2 - parameters calculated in the previous step;

P - base point of the elliptic curve, and Q - public key of the signatory.

The last step is to verify that the calculated point corresponds to the signature parameters: first, the x-coordinate of the point V, denoted as x_v, is obtained. Then the condition for confirming the equivalence of the signature parameter r in the modular space n is performed using the following formula:

$$r = x_v\, mod\, n \tag{10}$$

If this condition is met, the signature is considered valid, and vice versa.

1.3. Stability analysis of digital signature algorithms using elliptic curves

The algorithmic strength of asymmetric encryption is calculated as the reliability defined in DSTU 4145-2002, which is approximately equal to 2^{2m}, where m is the degree of expansion of the main field, then the complexity of identifying the parameter during the enumeration of cryptographic calculations processed by the presented algorithm is defined as 2^t, in this case, the parameter t restrictions are imposed $t > 0$.

The main reason for the high resistance of ECDSA is the complexity of solving the ECDLP algorithm. The task of cracking is to find a certain number k (unknown scalar) for the given points P (known point on the curve) so that the following equality holds $Q = kP$. Modern algorithms, such as two-way search and Babcock-Rivert's convergence method, are inefficient for solving ECDLPs due to the property of curves to have a sufficiently large order. On the other hand, cryptosecurity also depends on the correct choice of curve parameters, since not only must the original order of the group be a large prime number, but the generator point must also belong to a group with a large order. For example, curves with a small ratio coefficient may be less resistant to attacks using division methods, so for a sufficient level of protection, it is recommended to use curves that have already passed the vulnerability check and are recommended by the above standards.

To describe the possible attacks that can be carried out when using ECDLP, the following subparagraphs were identified:

An exhaustive search (Brute Force Attack):

Since the space of possible values is very large (growing exponentially with the key size), this attack is extremely inefficient in practice, even for moderate key sizes such as 256-bit numbers.

Baby-Step Giant-Step Attack

The baby giant steps method optimises the search process by splitting the search space into smaller subsets. First, all possible values are calculated $b_i = \text{i}P$ for small steps i(baby-step) and are stored in a table. Then the values are calculated $g_j = Q - j(mP)$ for big steps j (giant steps), where m - some order value n. Finding a match between values in a table allows you to find k is much faster than a full lookup. The main disadvantage of this method is the need for a large amount of memory to store tables.

Attack using the Pollard-Roe method (Pollard's Rho Algorithm)

The Pollard-Roe algorithm applies the ideas of random walk to find the discrete logarithm. It uses a function that defines random paths in the space of possible values to find two points that coincide (collisions). Once a collision is found, you can output the value k. The algorithm efficiently uses the ρ- image, that visualises the collision detection process and works out an average of $O(n)$ steps, but requires significantly less memory compared to the baby giant steps method. Parallel collision detection is an extension of the Pollard-Roe method, which uses parallel computing to speed up the process: several processors or computers work on the same task, distributing the search among them, which can significantly reduce the time required to find the discrete logarithm, although it requires coordination between processors and certain data exchange costs.

2. Practical implementation of digital signature algorithms based on elliptic curves

2.1 Selection and justification of software tools for the implementation of digital signature algorithms

The implementation was carried out in the C# programming language in the Microsoft Visual Studio 2022 integrated development environment. For proper operation of the application, the operating system Windows 10 or a newer version that supports the x64 architecture was required. However, when it comes to compiling the application, you can use any compiler that is compatible with the standards of the C# programming language. The advantages of using C# are the provision of convenient tools for working with the bit level, which is important for the cryptographic operations

under consideration, which involved hashing and manipulating bits when working with the key space, and the ability to use the .NET platform, which provides alternative implementation options for Mono and .NET Core, which allow the code to be used on various platforms, including Linux and macO.

The System.Security.Cryptography library is part of the standard .NET Framework or .NET Core class library and contains classes and other tools for working with cryptographic operations in C# programs. Another key security element in C# is the type system, which allows you to define and control the types of data that can be used in a program, which helps to avoid many types of software errors, such as buffer overflows or memory leaks. The advantage of choosing C# was also the global using Xunit construct, which greatly simplifies the creation of test code, since you do not need to add imports, but simply use test instructions from the Xunit library, and the compiler automatically recognises these classes and methods due to global imports.

2.2 Technical description of the software implementation of the digital signature creation and verification algorithm

The objective is to analyse a generalised assessment of the possibility of integrating an algorithm for creating and verifying a digital signature that complies with the DSTU 4145:2002 standard into systems with limited resources, with minimal impact on time efficiency. The algorithm is based on an elliptic curve with the degree of the main field 179 in accordance with DSTU 4145:2002. To achieve optimality, the polynomial basis was chosen due to its simplified structure in software implementation, which includes the use of simple polynomials and widespread use due to standardisation requirements. Taking into account the need to replace the basis, the choice can be made on the basis of the inverse transition matrix based on mathematical principles.

The software representation did not include parameters for checking the primitivity of the selected polynomial, the simplicity of the base point order, or the use of the Menezes-Okamoto-Wenston condition. During the experimental study, the functional purpose was to perform basic operations on the set of points of elliptic curves, as well as to calculate the general parameters of the main field, the base point, and the identifier of the hashing function.

The programme does not require any additional data or parameters to be entered by the user to run, as they are built-in with the necessary algorithms for generation. However, if it is necessary to change parameters, such as the elliptic curve or other configuration parameters, this can be done directly in the program code before compiling it. This approach ensures that the programme can be run without the need to enter external data or parameters, which allows you to effectively manage its functionality within a given project.

The output data provided by the programme after its launch includes the following:

Measurement of the time required to generate a base point on an elliptical curve. This value is important for assessing the performance of the algorithm and its resource consumption;

Private key in hexadecimal format. This key is confidential and is used to create digital signatures;

Public key in the form of two hexadecimal numbers. This data is used to verify digital signatures;

A digital signature in hexadecimal format. This signature is used to confirm the authenticity of the message;

After performing digital signature verification, the program provides a result indicating whether the authenticity of the signed message has been successfully confirmed. This result is a sign of the message's authenticity;

The full result of the programme will be able to prove the following aspects:

The result is different digital signatures of the same message due to a random generation factor based on a generator on the time interval function;

When verifying a signature for the same message using a public key derived from a private key that is also randomly generated, the program will correctly confirm or reject the signature depending on whether it matches the message.

3. Experimental study of digital signature efficiency based on elliptic curves

3.1 Technical description of the software implementation of the algorithm stress test

Since side-channel attacks use physical properties of cryptographic algorithm execution to extract additional information, for example, analysing energy consumption or electromagnetic radiation during operations can reveal secret keys, protection against such attacks requires the implementation of special measures, such as masking computations, randomising operations, and shielding hardware. This topic was the impetus for the implementation of the experimental part on time measurement, since time-based attacks can use the difference in execution time for different key values to extract key information.

The purpose of this part of the experimental study is to conduct stress testing of the digital signature generation and verification algorithm using a large number of random messages, aimed at identifying the statistical relationship between the expected and actual usable performance level in different conditions: from low to high load and a variety of input data. This approach makes it possible to estimate the probability of effective operation and justify the level of possible delay in processing input data in systems with low throughput.

The structure of the software implementation includes the UnitTest1 class in the Test namespace, which contains the Test1 method. The constructor of the UnitTest1() class creates an array of RandomStrings of size 10. Each element of this array is a random string with a length of 1000 characters, which represents random data.

The method uses a parallel Parallel.For loop that iterates over the elements of the RandomStrings array. For each iteration of the loop, instances of the Ecc and DigitalSignature classes are created, which are used to generate and verify the digital signature. Each time, a new random private key d is generated, with which the corresponding public key q is generated. It is worth noting that the study also considers the option of using a non-parallel method, i.e. instead of Parallel.For, a regular for loop is used, the processing of elements of the RandomStrings array will occur sequentially, regardless of whether it can be processed. The use of parallel methods in testing can speed up the process, especially when the system needs to process large amounts of data, although performing verification simultaneously and in parallel requires additional system resources, so it is worth choosing an approach taking into account the implicit specific conditions and requirements for processing incoming messages.

The output is the results of the digital signature validation for each of the random messages.

3.2 Analysis of the influence of an array of messages on the stability and speed of the algorithm during parallel and sequential iteration

The first part of the study covers a comparative analysis of the results of parallel and sequential message processing in order to determine the optimal way to perform the digital signature algorithm according to DSTU 4145:2002 in accordance with the increased requirements and amount of processing resources.

Parallel message processing uses the capabilities of multi-core processors and distributed systems to calculate digital signatures for multiple messages simultaneously.

Sequential message processing performs digital signatures sequentially for each message, without parallel execution.

The task was to develop a test set of input data of the same length, but with different volumes and types, to compare the process, followed by serial experiments for 10 generation cycles to assess the algorithm's resistance to tampering attacks and load effects. The following statistical analysis is based on the results of comparing the obtained programme developments, which are listed in Table 3.1.

Table 3.1
Experimental observation of the dependence of time on the amount of input data of the same length and the iteration method

	Input data processing method			
	Parallel method		The sequential method	
Series №	Time, min	Number of start-up cycles to test	Time, min	Number of start-up cycles to test
1	1,4	10	4,4	10
2	3,2	25	11,8	25
3	6,7	50	23,3	50
4	9,9	75	31,7	75
5	13,2	100	43,5	100
Estimation of the harmonic average		3,69		12,2

Since the speed of an algorithm can be viewed as a specific performance (i.e., the number of tasks performed per unit of time), the harmonic mean provides a correct estimate of the average execution time of iterations, which will be quite influential for parallel computing, where performance can fluctuate significantly between iterations. In this case, given that the digital signature algorithm execution time test depends on the amount of data, the execution time can vary significantly depending on the amount of data, the harmonic mean was chosen, which takes into account the execution time and is calculated as a mutual arithmetic mean of the mutual, the final result is presented in Table 3.1 by the following formula:

$$\overline{t_h} = \frac{n}{\sum_{i=1}^{n} \frac{1}{t_i}} \tag{10}$$

where $\overline{t_h}$ is the average harmonic value of the experiment time;
t_i – time for one series of the start-up cycle;
n – number of series.

The harmonic mean for the parallel method is approximately 3.69 minutes, while for the sequential method it is about 12.2 minutes. Given that a higher value of the harmonic mean indicates a shorter execution time, the parallel method of processing the input data was more efficient than the sequential method (almost 3.3 times). Therefore, the harmonic mean was chosen to estimate the execution time, as it better accounts for large values and is much more sensitive than other Pythagorean averages to variations in large emissions or 'worse' values.

According to Table 3.1, the optimal processing method, taking into account all the aspects studied in terms of performance, is parallel if speed and large input data are preferred, an example of one development cycle is shown in Figures 1 - 2, although at the same time, excessive parallelisation can be avoided in the case of smaller data.

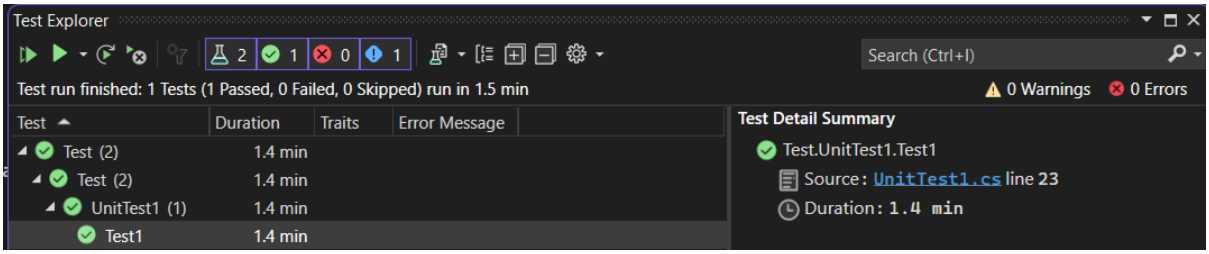

Figure 1. Running a stress test using the parallel method

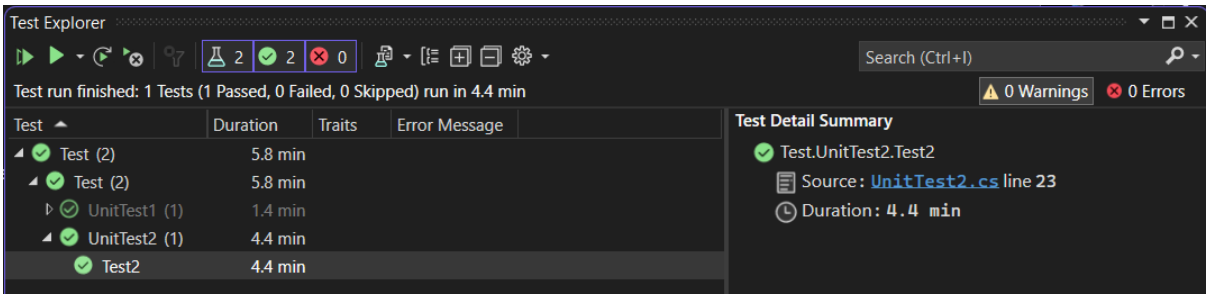

Figure 2. Performing a stress test using the sequential method

3.3 Estimating the processor load when using the algorithm in parallel and sequential iterations

The second part of the experimental study aims to evaluate the CPU load during the execution of the algorithm using parallel and sequential data processing methods by measuring the amount of RAM used.

As part of the study, we used not only standard Visual Studio tools, but also additional tools such as ReSharper, an extension for the .NET platform.

The main features of dynamic analysis [6] in ReSharper include:

Detect and automatically fix code performance issues, such as under-optimised or inefficient parts of the application that can slow down the application;

- In-depth research and provision of process tools for analysing and monitoring the memory usage of a software product, which allows to identify potential memory retention issues and detect memory leaks that can lead to unstable operation of the application or its crash.

To run dynamic analysis in ReSharper, we used the built-in tools of this plug-in after installing it as an additional tool in Visual Studio. To display it as an extension, it was enough to open the solution of the project in which the study was carried out and select the option 'Run Code Analysis', hereinafter 'DPA' from the ReSharper menu or in the context menu of the corresponding stress test. After running the algorithm iterations both in parallel and sequentially, the amount of memory used in each case was recorded.

A successful implementation is the result of automatic analysis of the project code after running in the 'Debug' mode, some of which are presented in Appendix H. The results of measuring the amount of RAM used for both data processing methods are shown in Table 3.2.

When choosing a mean estimation method, it is important to understand what properties of each method make it most suitable for a particular type of data. The geometric mean is better suited for data that is multiplicative or orthonormally distributed. The presented part of the study is related to additive values (memory and CPU utilisation percentages), for which the geometric mean will not give a correct picture of the average level of resource utilisation. Likewise, the harmonic mean is not appropriate because it analyses absolute values of resource utilisation, not relative values. Using the arithmetic mean is the most appropriate method to estimate the average amount of RAM and CPU usage in our study, as these are additive and the arithmetic mean provides a simple and accurate estimate. The arithmetic mean formula is presented as follows:

$$\overline{M} = \frac{\sum_{i=1}^{n} m}{n} \tag{11}$$

where $\overline{M}$ – the arithmetic mean of the experiment memory;

m – amount of RAM per series;

n – number of series.

According to Table 3.2, the parallel method of data processing has a slightly higher amount of RAM usage compared to the sequential method, as parallel computing requires additional resources for thread management and data synchronisation. It can also be concluded that the parallel method also creates a greater load on the processor compared to the sequential method - parallel computing uses more processor cores to perform several tasks simultaneously.

Table 3.2
Experimental observation of the dependence of RAM on the amount of input data of the same length and the iteration method

	Input data processing method			
	Parallel method		The sequential method	
Series №	Time, min	Number of start-up cycles to test	Time, min	Number of start-up cycles to test
1	268,3	10	261,5	10
2	272.32	25	265.42	25
3	279.03	50	272.96	50
4	285.67	75	280.52	75
5	292.45	100	285.14	100
Estimation of the harmonic average		279.95		273.11

4. Conclusions

The key functionality of the studied digital signature algorithm based on elliptic curves is its ability to provide a high level of cryptographic security, which is achieved through the use of complex mathematical structures such as abelian groups and discrete logarithm. It has been theoretically substantiated and practically confirmed that this approach, based on the use of DSTU 4145:2002 standards, significantly increases the security of authentication procedures and ensures integration into high-load information systems.

The effectiveness of the algorithm is based on its ability to provide reliable cryptographic encryption with minimal performance loss, which was confirmed during experimental studies of stability and performance. The method of parallel data processing used in the stress tests demonstrates a significant reduction in execution time, which is critical for the application of the algorithm in high-load systems. The experimental results also showed that the correct choice of elliptic curve parameters significantly affects the overall resistance to cryptanalytical attacks, especially to such attacks as the Pollard-Roe method and baby giant steps attacks.

The use of mathematical methods to calculate the key parameters of the algorithm allows it to be flexibly adapted to various environments, increasing efficiency in the context of limited resources. An important advantage is the implementation of the algorithm on the basis of the modern C# platform using the System.Security.Cryptography library, which provides optimal conditions for working with cryptographic operations. This increases the level of security and compatibility with other cryptographic standards, which allows it to be implemented in various areas, including financial systems and commercial applications.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] DSTU 4145:2002. Informatsiini tekhnolohii. Kryptohrafichnyi zakhyst informatsii. Tsyfrovyi pidpys, shcho gruntuietsia na eliptychnykh kryvykh. Formuvannia ta pereviriannia. Chynnyi vid 2003-07-01. Vyd. ofits. Kyiv: Male pidpryiemstvo «Dyna», 2002. 36 s.

[2] Lawrence Washington Elliptic curves, Number theory and Cryptography. – CRC Press, 2000. – 430 c.

[3] Zavhorodnii, V. V., H. A. Zavhorodnia, Yu. S. Berezinskyi, i I. P. Berezinska. «REALIZATsIIa KRYPTOSTIIKOHO ALHORYTMU IZ PROSTOIu PROTsEDUROIu ShYFRUVANNIa TA DEShYFRUVANNIa NA OSNOVI ELIPTYChNYKh KRYVYKh». Tavriiskyi naukovyi visnyk. Seriia: Tekhnichni nauky, vyp. 3, Zhovten 2023, s. 13-20, doi:10.32782/tnv-tech.2023.3.2.

[4] Ievseiev, S. P. Laboratornyi praktykum z osnov kryptohrafichnoho zakhystu [Elektronnyi resurs] : navch. posib. / S. P. Yevseiev, O. V. Milov, O. H. Korol ; Kharkivskyi natsionalnyi ekonomichnyi universytet im. S. Kuznetsia. - Elektron. tekstovi dan. (12,3 MB). - Kharkiv : KhNEU im. S. Kuznetsia, 2020. - 221 s. : il. - Zahol. z tytul. ekranu. - Bibliohr.: s. 211-213(date of access 12 May 2024) .

[5] Pro Polozhennia pro poriadok zdiisnennia kryptohrafichnoho zakhystu informatsii v Ukraini : Ukaz Prezydenta Ukrainy vid 22.05.98 r. № 505/98. – Rezhym dostupu: http://zakon.rada.gov.ua.

[6] Dynamic Program Analysis (DPA). ReSharper. URL: https://www.jetbrains.com/help/resharper/Dynamic_Program_Analysis.html (date of access 12 May 2024).