UDC 003.26.004.056.55(07)
DOI: 10.32626/2308-5916.2024-25.78-88

Volodymyr Palahin, Dr.Sc.,
Olena Palahina, PhD,
Oleksandr Ivchenko, PhD,
Anatolii Bairak
Cherkasy State Technological University, Cherkasy

DEVELOPMENT OF THE COMBINED OPERATIONS METHOD TO IMPROVE THE EFFICIENCY OF BLOCK ENCRYPTION

Real-time information protection requires the implementation of special methods aimed at providing reliable and fast encryption algorithms to protect personal and corporate information from unauthorized access. With the growth of data volumes and the speed of their processing, the importance of effective encryption methods increases significantly. One of the common, reliable and well-known encryption algorithms is AES (Advanced Encryption Standard), also known as Rijndael, which is a symmetric block cipher. AES has high efficiency and cryptoresistance and is suitable for processing large volumes of data. The reliability and speed of encryption and decryption using the AES algorithm depends on the size of the key and the data.

This paper proposes to improve algorithm of symmetric block encryption AES to provide faster data processing. The possibility of combining mathematical operations that have a similar principle of processing elements is shown. This approach made it possible to reduce the processing time for data encryption and decryption compared to known implementations. A comparative analysis of the practical implementation of the standard and optimized AES cryptoalgorithms has been carried out. The general principles of the proposed method are to transform all two-dimensional arrays into one-dimensional arrays, add auxiliary tables for ShiftRows and MixColumns operations, and combine operations with similar element processing principles. The simulation results showed that the modified implementation of the AES algorithm demonstrates a reduction in processing time of up to 50% when encrypting and up to 75% when decrypting data compared to known results.

Key words: *cryptographic data processing, symmetric block cipher, optimization of data processing.*

1. Introduction. Ensuring data safety and security are relevant in any sphere of human activity, which is directly related to the implementation of modern and advanced symmetric and asymmetric algorithms and methods of cryptographic protection of information. One of the well-known and safe

encryption algorithms, known as AES (Advanced Encryption Standard), which is endowed with the properties of ease of practical implementation and high reliability, is widely used for block symmetric data encryption [1]. Features of symmetric encryption is the processing of large amounts of data, use of the same key for encryption and decryption operations, high reliability of data protection [2]. However, when processing large amounts of data, certain time delays are observed, which can be critical for some tasks. The speed of data encryption and decryption using the AES algorithm depends on the size of the key. Depending on the length of the key, the number of encryption rounds and the time of round key generation also change. Even if 128-bit keys are used, there will be a noticeable time delay with large amounts of data. Therefore, there are already many effective hardware and hardware-software implementations for AES optimization [3-9]. It should be noted that the optimization of the AES algorithm can be extended to the operation of the symmetric block algorithm «Kalyna», which has become the national encryption standard of Ukraine and ensures high dispersibility of data and their mixing [10, 11]. It is also necessary to note the need to improve the efficiency of symmetric block crypto-algorithms and develop new methods to reduce processing time, which is associated with the constant growth of data flows that are processed, including in real time [12]. The analysis of the literature shows that the acceleration of data encryption and decryption operations is mainly performed when additional hardware resources are used. There are not enough software implementations to increase the speed of data processing by AES algorithms [13]. Thus, there is a need to improve the AES algorithm without significant changes in transformations that may affect the cryptographic strength of the cipher.

It is proposed to improve the AES symmetric encryption algorithm to provide faster data processing. All modifications of the AES transformations do not reduce the high cryptographic strength of the cipher, and their principles are described in detail in the paper. This practical implementation of the optimized algorithm shows the possibility of combining a number of mathematical operations that have a similar principle of processing elements, which allowed to reduce the processing time.

The main objective of the paper is to reduce the time of cryptographic processing of data in the symmetric block cryptographic algorithm AES by optimizing mathematical operations and improving the software implementation without reducing the cryptographic strength of the algorithm.

2. Method of optimization of procedural transformations. AES encryption and decryption are based on four different transformations. They are performed repeatedly in a certain sequence. The mathematical foundations of the construction of the algorithm are described in [1, 14]. The processing time of round-robin data encryption and decryption processes depends on the length of the encryption key.

The article discusses ways to increase the speed of cryptographic data processing.

One of the ways to increase the data processing speed is to improve the principles of working with the State matrix. The State matrix has a fixed size – 128 bits and 4 rows and 4 columns. The matrix is filled in by columns. After round transformations, the result of encryption will need to be read from the columns of the original array of encrypted bytes. This requires additional calculations and loop initialization. It is proposed to fill the data block in the form of a one-dimensional array, but also by columns Fig. 1.

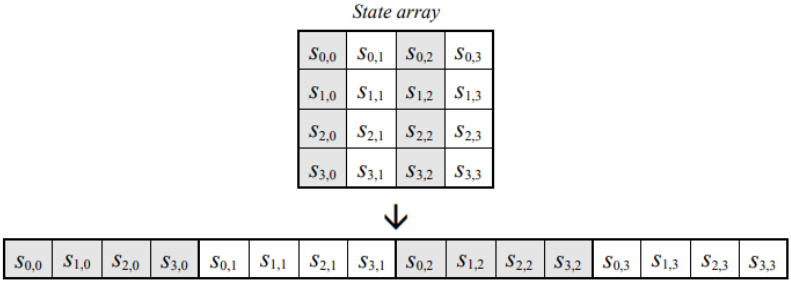


Fig. 1. *Filling the modified State array*

Note that the MixColumns and AddRoundKey operations work with columns. Therefore, it will not be difficult to modify these transformations to work with the changed state matrix. And it will not be necessary to convert the matrix of the result into a string of data.

In addition to the modifications of individual round transformations, it is possible to speed up the execution of encryption by combining them. For example, if we combine two operations that have similar loops in the program code, then it will go through half as many loop iterations, which will significantly reduce data processing time. Therefore, for the practical implementation of an optimized algorithm, it is necessary to combine operations that have a similar principle of processing elements.

MixColumns is a complex procedure for mixing columns, where each column of the State matrix is multiplied by a fixed polynomial. In matrix form, it looks like this (1)

$$\begin{bmatrix} s'_0 \\ s'_1 \\ s'_2 \\ s'_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}. \quad (1)$$

InvMixColumns is a complex procedure of mixing columns where each State column is multiplied by another specified polynomial (2)

$$\begin{bmatrix} s'_0 \\ s'_1 \\ s'_2 \\ s'_3 \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}. \quad (2)$$

To process one column, you need to perform 8 multiplication operations for encryption and 16 operations for decryption. Then only for one round is 24/64 operations of multiplication by defined constants. Instead of implementing such multiplication operations, you can use pre-calculated tables to find the required value. To do this, you need to define 6 such tables – two for the encryption process and four for the decryption process.

Let's write in the data line the results of multiplying all possible numbers up to 1 byte (256 bits) by {02}. The same operations must be calculated for other constants – {03}, {09}, {0b}, {0d}, {0e} (Table 1). In the software implementation, it is necessary to initiate 6 tables of 256 bits each [15].

Table 1

An example of creating auxiliary tables for mixcolumns

	The number to be multiplied										
	00	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**…**
Result of multiplication by {02}	00	02	04	06	08	0a	0c	0e	0e	12	…
Result of multiplication by {03}	00	03	06	05	0c	0f	0a	09	18	1b	…
multiplication by {09}{09}, {0b}, {0d}, {0e}											

Next, you need to find the required number in the table. The value of the State array element will be the index which to search for the element in the specified tables for MixColumns. This transformation should go through 4 iterations. The AddRoundKey transformation must also go through 4 iterations. Therefore, you can combine them into one function. When encrypting, the MixColumns operation is performed first, then AddRoundKey, and the reverse is true when decrypting. In the last round, you can't mix the columns to ensure the decryption of the text. A good solution is to leave the AddRoundKey function, which will be applied only in the last round of encryption.

It is also possible to speed up encryption execution by combining separate cyclic transformations. In the practical implementation of the optimized algorithm, it is suggested to combine operations that have a similar element processing principle: MixColumns, AddRoundKey. This implementation makes it possible to get rid of 36-48 (depending on the number of rounds) iterations of passing cycles.

SubBytes is a byte swap operation. It works independently of each byte of the data matrix using the S-box substitution table [15]. For example, if the

element of the matrix *s* = {53}, then the substitution value will be determined by the intersection of the row with the index «5» and the column with the index «3». The value of the element with is converted to {ed}.

In the S-box table, you can easily split the element value into two parts and get the row and column indexes of the lookup table. The software implementation requires mathematical transformations to separately determine the first and second digits of the element. To solve this problem, you can write the S-box table as a one-dimensional array of 256 elements. Then there is no need to calculate row and column indices.

ShiftRows is an operation for ordering the elements of the State matrix. It performs a cyclic shift of each row. The length of the cyclic shift in each row is different. The zero row remains unchanged, in the first row each element is shifted one position to the left, the elements of the second and third rows are shifted two and three positions.

It is necessary to find a convenient and fast way to implement a cyclic shift for operation ShiftRows. A table of indexes can be created for each element of the block [16]. This table will indicate where each element should move. For example, if the zero row does not change, then the 0th, 5th, 9th, and 13th elements should remain at the 0th, 5th, 9th, and 13th positions, respectively (Table 2):

Table 2

Changing elements in the shiftrows operation

New index value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Old index value	0	5	10	15	4	9	14	3	8	13	2	7	12	1	6	11

InvShiftRows operation that performs a cyclic shift to the right by 1 element for the first row of State, by 2 for the second and by 3 for the third. The null line is not changed. For the inverse transformation, you also need to create an index table (Table 3) [16]:

Table 3

Changing elements in the invshiftrows operation

New index value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Old index value	0	13	10	7	4	1	14	11	8	5	2	15	12	9	6	3

It is also proposed to merge the modified ShiftRows and SubBytes operations into a single function. This will reduce the number of cycles in both operations. The software implementation will get rid of 160-224 (depending on the number of rounds) iterations of the cycle. That is, thanks to this combination, instead of 32 iterations per round, we will already use 16.

In Fig. 2. a simplified scheme of the optimized AES crypto-algorithm is presented, where the decryption process will have a similar principle.

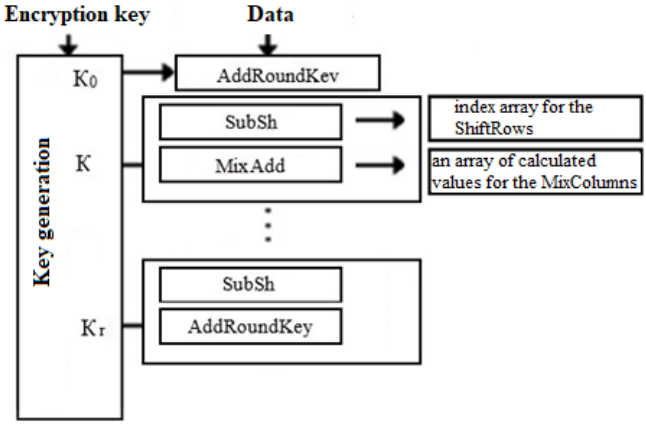


Fig. 2. *Filling the modified State array*

3. Results of implementation of optimization of procedural transformations. To determine the degree of optimization of the AES crypto-algorithm, it is necessary to compare the time required for encryption and decryption for two software versions of the algorithm – traditional and improved. For the software implementation of the algorithm, an open-source programming language, Python, was chosen. Performance evaluation based on program execution time was performed in Windows 10.

To obtain comparative characteristics, studies were carried out for AES keys 128 bits, 192 bits. The basis is text files of different sizes – 1 KB, 10 KB, 50 KB, 100 KB and 200 KB. This will allow us to estimate the dependence of the processing time on the length of the message.

The simulation results of the modified encryption and decryption algorithms are shown in the tables 4, 5.

Table 4

Results of algorithm optimization: encryption

Data size, (Kb)	Encryption					
	AES-128			AES-192		
	traditional, (c)	improved, (c)	saving time, (%)	traditional, (c)	improved , (c)	saving time, (%)
1	0,02	0,01	50	0,02	0,01	52
10	0,23	0,12	48	0,27	0,13	52
50	1,43	0,86	40	1,61	0,95	41
100	3,57	2,42	33	3,93	2,59	34
200	9,74	7,52	23	10,48	7,68	27

Table 5

Results of algorithm optimization: decryption

Data size, (Kb)	Encryption					
	AES-128			AES-192		
	traditional, (c)	improved, (c)	saving time, (%)	traditional, (c)	improved , (c)	saving time, (%)
1	0,05	0,01	76	0,06	0,01	75
10	0,52	0,13	75	0,62	0,15	76
50	2,87	0,93	68	3,37	1,06	69
100	6,45	2,51	61	7,43	2,73	63
200	15,53	7,74	50	17,35	7,89	55

Analysis of the results shows that decryption in a traditional crypto algorithm takes twice as long as encryption. This is due to the greater number and complexity of the multiplication operations of the InvMixColumns transformation. Improving this function significantly improves performance. In general, all minor modifications in the aggregate give a significant result of optimizing both the encryption process and the decryption process.

Graphs comparing the time required for encryption and for decryption of implemented crypto-algorithms (for the average key length – 192 bits) are given (Fig. 3, Fig. 4).

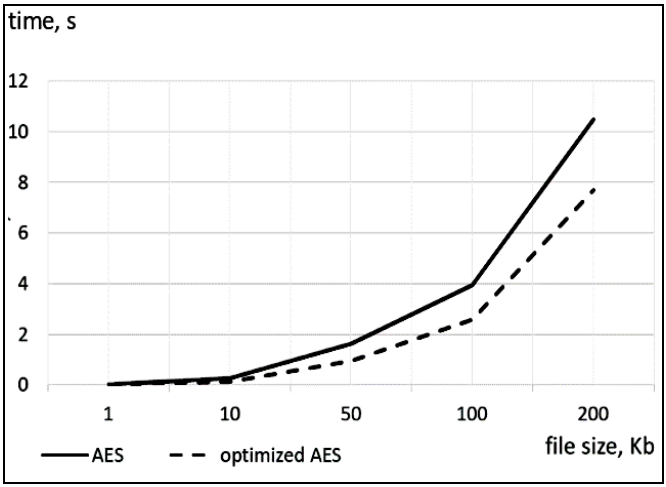


Fig. 3. *Time required for encryption of implemented crypto-algorithms*

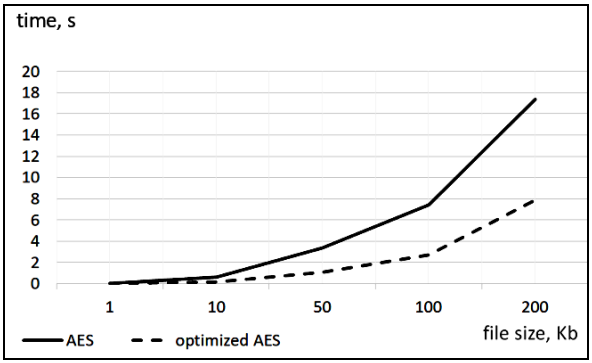


Fig. 4. *Time required for decryption of implemented crypto-algorithms*

Studies have shown that the implementation of data processing optimization procedures can significantly reduce encryption and decryption time. The reduction in processing time depends on the size of the key and the amount of data being processed. The modified of AES demonstrates a significant percentage of time optimization – up to 50% when encrypting and up to 75% when decrypting data.

An application for performing data encryption and decryption operations based on an advanced method has been developed (Fig. 5). The application consists of choosing the encryption or decryption procedure, as well as the key size and its generation. Features of the software implementation include a convenient graphical interface, support for Cyrillic characters, generation of a unique key and its recording in a file, the ability to select a text file for further encryption, selection of the AES encryption standard (128, 192, 256), use of the advanced AES algorithm.

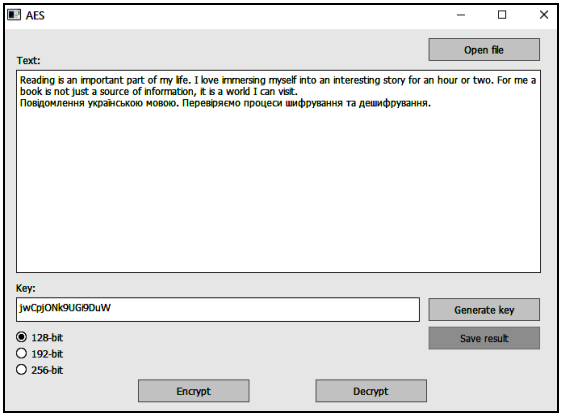
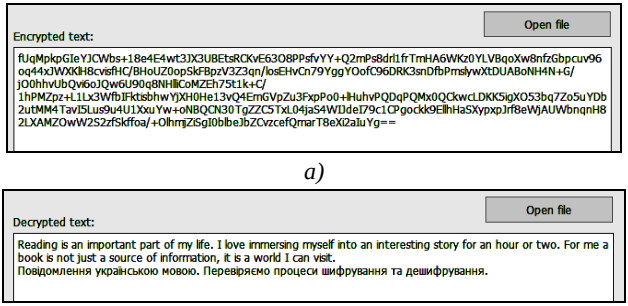


Fig. 5. *Application for encrypting and decrypting data with an advanced AES algorithm*



a)

b)

Fig. 6. *Fragment of the application windows for encrypted (a) and decrypted data (b)*

Thus, a new method for increasing the efficiency of processing block ciphers has been proposed. This approach can be used in other block algorithms to reduce data processing time.

Conclusions. The paper proposes a new method for optimizing the AES algorithm. The new method is based on combining a mathematical data processing operations that have a similar element processing principle. This made it possible to reduce the time of data processing compared to the known implementation. At the same time, the cryptographic stability of the cipher was preserved. The features of the proposed method are the transformation of all two-dimensional arrays into one-dimensional arrays, the addition of auxiliary tables for ShiftRows and MixColumns operations. A study of the effectiveness of the proposed method was conducted. A modified implementation of the AES algorithm demonstrates a reduction in processing time of up to 50% when encrypting and up to 75% when decrypting data compared to known results.

References:

1. FIPS PUB 197. Specification for the Advanced Encryption Standard (AES). 2001.
2. Daoud L., Hussein F., Rafla N. Optimization of Advanced Encryption Standard (AES) Using Vivado High Level Synthesis (HLS). *EPiC Series in Computing.* 2019. Vol. 58. P. 36-44.
3. Soltani A., Sharifian S. An ultra-high throughput and fully pipelined implementation of AES algorithm on FPGA. *Microprocessors and Microsystems.* 2015. Vol. 39. № 7. P. 480-493.
4. Karthigaikumar P., Anitha Christy N., Siva Mangai N. M. PSP CO2: an efficient hardware architecture for AES algorithm for high throughput. *Wireless Personal Communications.* 2015. Vol. 85. №1. P. 305-323.
5. Taufik M., Amin D. E., Saifuddin M. A.. Hardware implementation and optimization of advanced encryption standard (AES) algorithm based on CCSDS. *AIP Conference Proceedings.* 2020. Vol. 2226. P. 060004
6. Liu Y., Xu X., Su H. AES Algorithm Optimization and FPGA Implementation. *IOP Conf. Series: Earth and Environmental Science*. 2019. Vol. 267. №4. P. 042070.

7. Pavithra S., Vaishnavi M., Vinothini M., Umadevi V. Optimization of aes algorithm using hardware and software. *International Journal of Scientific & Engineering Research.* 2015. Vol. 6. № 4. P. 965-971
8. Li M., Gan J., Cheng S., Hu X., Yu Y., Li J. Design of Lightweight AES Algorithm Based on Masked Lookup Table. *IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT).* 2019. P. 437-441.
9. James M., Kumar D. S. An Optimized Parallel Mixcolumn and Subbytes design in Lightweight Advanced Encryption Standard. *International Journal of Computational Engineering Research (IJCER).* 2016. Vol. 6. № 3. P. 25-28.
10. Gorbenko I. D., Totsky O. S., Kazmina S. V. Promising block cipher «Kalina» – the main provisions of the specification. *Applied Radio Electronics.* 2007. T. 6. № 2. P. 195-208.
11. Kalinin D. A., Kozina G. L. The speed of the Kalina and AES ciphers. *Radioelectronics, informatics, management.* 2013. № 1. P. 62-65.
12. Khatri Y., Chhabra R., Gupta N., Khanna A., Gupta D. Secure modified aes algorithm for static and mobile networks. *International Conference on Innovative Computing and Communications*. 2020. P. 389-399.
13. Arman S., Rehnuma T., Rahman M. Design and Implementation of a Modified AES Cryptography with Fast Key Generation Technique. *2020 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE).* 2020. P. 191-195.
14. Stallings W. Cryptography and network security: principles and practice. 2017. Vol. 6. Pearson Education.
15. Rijndael MixColumns. URL: https://en.wikipedia.org/wiki/Rijndael_MixColumns
16. Riyaldhi R., Kurniawan A. Improvement of advanced encryption standard algorithm with shift row and S. box modification mapping in mix column. *Procedia computer science.* 2017. Vol. 116. P. 401-407.

РОЗРОБКА МЕТОДУ КОМБІНОВАНИХ ОПЕРАЦІЙ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ БЛОКОВОГО ШИФРУВАННЯ

Захист інформації в режимі реального часу потребує реалізації спеціальних методів, які спрямовані на забезпечення надійних та швидких алгоритмів шифрування для захисту персональної та корпоративної інформації від несанкціонованого доступу. Зі зростанням обсягів даних і швидкістю їх обробки, важливість та актуальність розробки ефективних методів шифрування суттєво зростає. Одним із поширених, надійних та відомих алгоритмів шифрування є AES (Advanced Encryption Standard), також відомий як Rijndael, який є симетричним блоковим шифром. AES має високу ефективність та криптостійкість і підходить для опрацювання великих обсягів даних. Надійність та швидкість шифрування та дешифрування за допомогою алгоритму AES залежить від розміру ключа та даних.

В роботі пропонується вдосконалення алгоритму симетричного блокового шифрування AES для забезпечення швидшої обробки даних. Показано можливість комбінування математичних операцій, що мають схожий принцип обробки елементів. Такий підхід дозволив скоротити час обробки для шифрування та дешифрування даних порівняно з відомими реалізаціями. Проведено порівняльний аналіз практичної реаліза-

ції стандартного та оптимізованого криптоалгоритмів AES. Програмна реалізація алгоритму AES, яка базувалась на офіційній специфікації стандарту шифрування, була модифікована для зменшення часу обробки даних з умовою збереження криптографічної стійкості шифру. Загальні принципи запропонованого методу полягають у перетворенні усіх двовимірних масивів на одномірні, додаванні допоміжних таблиць для операцій ShiftRows та MixColumns, об'єднанні операцій зі схожими принципами опрацювання елементів. Результати моделювання показали, що модифікована реалізація алгоритму AES демонструє скорочення часу обробки до 50% при шифруванні та до 75% при дешифруванні даних у порівнянні з відомими результатами.

Ключові слова: *криптографічна обробка даних, симетричний блоковий шифр, оптимізація обробки даних.*

Отримано: 03.09.2024

UDC 004.9; 004.8
DOI: 10.32626/2308-5916.2024-25.88-96

Myhajlo Sydoruk*,
Solomiia Liaskovska*,,** PhD

*Lviv Polytechnic National University, Lviv,
**Kingston University, London, United Kingdom

AN ENSEMBLE METHOD FOR THE FRAUD DETECTION IN TRANSACTIONS

In today's world, bank fraud has become one of the significant threats to the financial stability and security of clients of financial institutions. The development of technologies, in particular in the field of machine learning, opens up wide opportunities for building effective systems for detecting and preventing fraud in the banking sector [1, 2].

Detecting fraudulent transactions is an important task that requires thoughtful and technological solutions. One of these methods is the use of machine learning approaches and methods.

This paper proposes the use of an ensemble method that combines several machine learning models at once. This approach will reduce the probability of false positives and increase classification accuracy. In addition, for the optimal operation of the model, preprocessing of the data will be carried out, in particular, their normalization, balancing of classes, as well as the selection of features. During the research, it is important not only to achieve high accuracy, but also to reduce as much as possible the number of fraudulent transactions that will be mistakenly classified as normal [3]. This is related to the business requirements of the banking sector, as each such transaction causes losses to the system's reputation, as well as direct financial losses.