The practice of block symmetric encryption for a secure Internet connection

Tetiana Korobeinikova[1,*,†], Ihor Zhuravel[1,†], Lesya Mychuda[1,†] and Axel Sikora[2,†]

[1] Lviv Polytechnic National University, 12 Stepana Bandery str., 79000 Lviv, Ukraine
[2] University of Applied Sciences Offenburg, 24 Badstrasse, 77652 Offenburg, Germany

Abstract

The paper discusses the process of block parallelization in the Advanced Encryption Standard (AES) cipher, focusing on the Counter (CTR) mode. It details the benefits of this process, including increased data processing performance and effective resource utilization; emphasizes the independent encryption of each data block in CTR mode, which allows for effective parallelization, especially when handling large data volumes. This work outlines the steps involved in the AES operation scheme in CTR mode, from splitting data into blocks to generating the final ciphertext. It further explains the concept of a unique "counter" or "initialization vector" for each block, which, combined with the key, generates a unique encryption key, enabling parallel processing. The idea implementation delves into the programming of the block parallelization algorithm using services on the Java Spring Boot platform. It describes the roles of the purposed Client Service and Server Service in encrypting and transmitting messages and files and decrypting received messages. This work presents an experiment that tests the hypothesis that blocks parallelization in AES cipher using CTR mode increases performance during the processing of large data volumes. The experiment involves different data volumes and compares the processing speeds of the AES algorithm with and without parallelization. The results confirm the hypothesis, showing that block parallelization in AES for large data volumes can double the data processing speed compared to the non-parallel approach. The paper concludes that block parallelization might be effective not only for the AES algorithm but also for any block symmetric algorithm. It also suggests that parallelization allows for more efficient use of multi-core systems and reduces the execution time to complete the encryption operation.

Keywords

block parallelization, AES cipher, CTR mode, data processing performance, encryption optimization

1. Introduction

In a world of increasing digital connectivity and Internet interaction, the issue of connection security is becoming one of the most crucial problems [1–9]. Ensuring the confidentiality, integrity, and availability of information is becoming a priority [9–11]. This work is focused on reviewing and improving block symmetric encryption methods aimed at solving the problems of connection security on the Internet. New technologies and encryption algorithms are important tools for maintaining data confidentiality and ensuring resilience to modern cyber threats [12–17].

Thus, there is a need to analyze, improve, and expand the methodological base of methods and tools that use block symmetric ciphers.

The **goal** is to improve the methods of block symmetric encryption through the process of block parallelization to solve the problem of secure connections on the Internet.

The **object** is the processes associated with the use of block encryption algorithms, in the context of their application on the Internet and the HTTPS protocol.

The **subject** is methods and means of protecting connections on the Internet, including the analysis of encryption efficiency.

2. Related works

It is known that the increasing number of digital connections during Internet interaction becomes the issue of connection security. Cristina Del-Real et al. [4] say that the design of software systems plays a crucial role in mitigating cybersecurity incidents; Sood et al. [5] and Jang et al. [9] are paying attention to DoS attacks through HTTP and IP.

Gentile et al. In [1] showed the algorithms to ensure suitable data transmission and encryption ratios and used Transport Layer Security (TLS) tunnels for local sensor data and secure socket layer tunnels to transmit TLS-encrypted data to a cloud-based central broker. Also, Kampourakis et al. in [3] offer a framework that can accurately detect all anomalous enterprise network activities.

Nowadays connections to the Internet are connections to the Cloud. Cloud storage is popular among syudy and businesses because of cost reduction, performance

CSDP-2024: Cyber Security and Data Protection, June 30, 2024, Lviv, Ukraine
*Corresponding author.
†These authors contributed equally.
tetiana.i.korobeinikova@lpnu.ua (T. Korobeinikova); ihor.m.zhuravel@lpnu.ua (I. Zhuravel); lesia.z.mychuda@lpnu.ua (L. Mychuda); axel.sikora@hs-offenburg.de (A. Sikora)
0000-0003-2487-8742 (T. Korobeinikova); 0000-0003-1114-0124 (I. Zhuravel); 0000-0001-8266-1782 (L. Mychuda); 0000-0003-0878-2919 (A. Sikora)

improvement, productivity enhancement, and security. However, Alqahtani et al. in [10] are focused on security risks since data is stored with third-party providers, and internet access can limit visibility and control. Compared to traditional on-premise computing, data security and protection are critical concerns in cloud computing. Ensuring data security in the cloud involves various methods, with cryptography being the most crucial (Sasikumar and Nagarajan in [11]).

So, cryptography provides several security features, including authentication, confidentiality, integrity, and availability.

Today, it is important to analyze, improve, and expand the methodological base of methods and tools that use block symmetric ciphers.

Boura et al. in [14] offer tools and algorithms to search for related-key distinguishers and attacks of a differential nature against the AES. Yevseiev et al. [15] offer the Niederreiter-modified crypto-code structure with additional initialization vectors that require an increase in the speed of cryptographic transformation of the system as a whole. While Opirskyy et al. [16] are working on standardizing post-quantum cryptography.

So, there is a critical need to work on improving the methods of block symmetric encryption through the process of block parallelization to solve the problem of secure connections on the Internet.

3. Initial information that establishes research

Stream encryption algorithms. A stream cipher is a type of symmetric encryption that encrypts each digit of the plaintext separately using a key stream. Stream ciphers are faster than block ciphers and require less hardware but can be vulnerable to attacks [17–19].

There are two types of stream ciphers: synchronous and self-synchronizing. Synchronous stream ciphers generate a sequence of pseudorandom digits independently of the plaintext and ciphertext messages. Self-synchronizing stream ciphers use the last N digits of the ciphertext to generate a key stream [20].

Block encryption algorithms. A block cipher is an algorithm that encrypts fixed-length data blocks to protect information. Electronic Code Book mode encrypts blocks independently, but patterns can be detected. To increase security, modes that introduce randomization through an Initialization Vector (IV) are used. Various modes of block cipher operation have been developed and defined in national and international standards [21–22]. The basic idea is to introduce randomization of the plaintext data using an IV to achieve probabilistic encryption.

The rationale for using block symmetric encryption for secure Internet connections. Encryption algorithms are known for their high computational requirements, especially for wireless devices with limited resources [23].

Encryption guarantees data confidentiality and protection against interception and plays an important role in authentication, data integrity, and access control.

Encrypting even a small amount of data, such as 13.6 kilobytes, using a 32-bit Blowfish key can consume about 75% of resources. Modern security standards recommend using keys of at least 80 bits.

Encryption and data transmission are important in wireless networks. There is potential for encrypting test packets with lightweight encryption algorithms if the security of the device is not compromised. There are numerous encryption algorithms for wired networks, divided into two categories: symmetric key encryption and asymmetric key encryption. Symmetric key encryption requires the distribution of a key between the parties before data is transmitted.

Asymmetric key encryption solves the problem of key distribution but is computationally intensive. In wireless devices, symmetric key encryption, such as RC4, is predominant, as it is fast and efficient. However, the discovered vulnerabilities in RC4 led to the introduction of a new security standard for WLANs—IEEE 802.11i, based on AES. AES is known for its speed, flexibility, and robust security features.

Block ciphers, such as AES, are used in WLANs to encrypt data, ensuring its confidentiality. Security protocols such as WPA2 and WPA3 use AES-CCMP to ensure data integrity. Block ciphers are also used to authenticate devices and users, and to manage encryption keys. They are the core components of WLAN security.

Block ciphers are used in WLANs to ensure data confidentiality and integrity, as well as device and user authentication. They are standardized, efficient, and scalable, making them a reliable choice for securing wireless communications. Overall, they enhance WLAN security by meeting security standards and requirements.

Rationale for choosing the AES to solve the problem of secure Internet connection. DES, while a popular encryption standard in the past, has fallen out of favor due to its limited key length and vulnerability to modern attacks. AES is known for its high security, efficiency, and wide support, making it the best choice for secure Internet connections. The Twofish algorithm, although it has its advantages, is not as widespread as AES and is not a standard for general use.

The rationale for choosing AES is based on its standardization, wide support, high level of security, and efficiency. Its status as a global standard recommended by key security institutions such as NIST emphasizes its reliability. Its ability to work efficiently in real time makes it an ideal choice for secure Internet connection tasks where speed and security of data transmission are important.

This choice provides strong encryption, meets modern requirements, and helps to create secure connections in a virtual environment.

AES can be slow when processing large amounts of data for several reasons. First, the AES block size is fixed (typically 128 bits), and a large amount of data requires many blocks, which can affect the overall speed. In addition, some AES modes of operation may include padding operations and other additional operations that can also increase the time consumption.

A large amount of data can lead to additional computational costs and increased execution time, in cases where enhanced security or authentication modes are used.

To improve the algorithm, a block parallelization mechanism was chosen.

The decision to optimize the AES algorithm using block parallelization for secure Internet connections can be justified:

- Large volumes of data: large amounts of data are commonly processed on the Internet, especially when transferring files or in other data-intensive scenarios. Block parallelization allows the efficient distribution of encryption tasks among different computing resources, ensuring fast processing.
- Increased performance: performance is a key factor in determining the quality of an Internet connection. Parallelization allows you to use parallel resources to simultaneously process multiple blocks of data, increasing the speed of encryption and decryption operations.

Data security: AES is a secure algorithm, but improving performance should not compromise security. Block parallelization allows for optimal performance without compromising encryption security.

4. The block parallelization process

Parallelization is a key strategy for encryption optimization aimed at increasing data processing performance in this work. It will show the main aspects of the block parallelization process, its benefits, and possible challenges, and consider how the implementation of block parallelization helps to optimize the AES cipher for high-performance information processing tasks.

4.1. Detailed description of the process

When parallelizing blocks and using AES encryption, CTR (Counter) [24] mode is the most effective strategy, especially in situations where parallel processing and high performance are important.

Optimized parallelization allows to maintenance of high performance in real-time processing areas where responsiveness to data is key. At the same time, efficient resource utilization and security ensure that large amounts of information are effectively handled, providing an optimal combination of performance and data privacy [25].

The choice of AES ciphers in CTR mode for block parallelization is based on their combination of security, efficiency, and standardization, namely, keys of different lengths (including 256 bits) and implementation speed.

In CTR mode, each block of data is encrypted independently, which allows to effectively parallelize the encryption process. This is especially important when processing large amounts of data, where parallel use of resources can significantly improve performance.

The combination of AES and CTR modes provides a robust encryption mechanism that can work effectively in a parallelized environment, ensuring a high level of security and data processing efficiency.

The main advantages of using CTR mode when parallelizing blocks:

- **Block independence** (CTR mode uses encryption with a key and unpredictable counter values for each block. This allows each block of data to be encrypted independently, which is ideal for parallel processing).
- **Ease of implementation and computation** (CTR mode is noted for its ease of implementation and minimized computation, as each block is encrypted independently of the others. This makes it effective for use in environments where computing resources are limited).
- **Parallel processing capability** (since each block is encrypted independently, CTR mode lends itself easily to parallel computing. Different blocks can be processed by different computing units or even on different devices, contributing to speed and efficiency).
- **Data Structure Preservation** (CTR mode allows you to preserve the structure of the original data during encryption. Each block is replaced by an encrypted block of the same length, which avoids data expansion or compression during encryption).

Note that while CTR mode has many advantages for parallel processing, it is important to manage the counter and key properly to avoid vulnerabilities. When using CTR mode, you should avoid reusing counter values for the same key.

The AES operation scheme in the CTR mode is shown in Fig. 1.

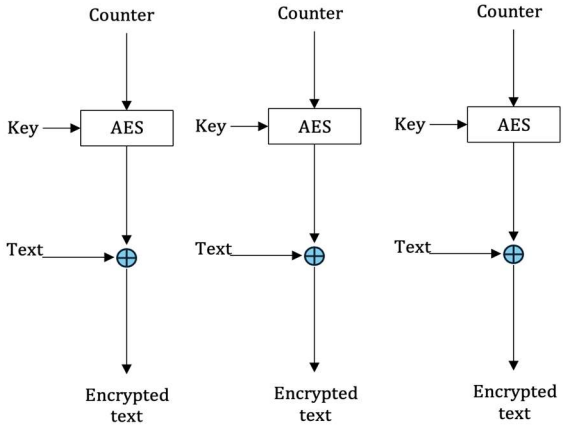


Figure 1: Diagram of the AES algorithm in CTR mode

The AES operation scheme in the CTR mode contains the following steps:

1. Splitting Data into Blocks: the input data is divided into blocks of fixed size. Each block has a length due to the AES block size (128 bits or 16 bytes).

2. Generate Counters: a unique counter is created for each block. It generates a unique key for each block.

3. Encrypting Blocks with AES: the resulting counters are used as input for the AES block used in the encryption mode. The generated result is XOR'd with the corresponding block of input data, which ensures block encryption.

4. Counter increment: after the block encryption is completed, the counter is incremented to generate a new value for the next block.

5. Repeat the Process for Each Block: All input blocks are processed similarly, generating unique keys for each.

6. Generate Ciphertext: the encrypted blocks are combined into the final ciphertext.

4.2. Block parallelization process diagram

The main idea of the AES block parallelization algorithm in CTR mode is to create independent keys for each data block, which allows to effectively parallelize the encryption process. Each block is processed independently, which improves the speed of processing large amounts of information. The AES block parallelization scheme in CTR mode is shown in Fig. 2.

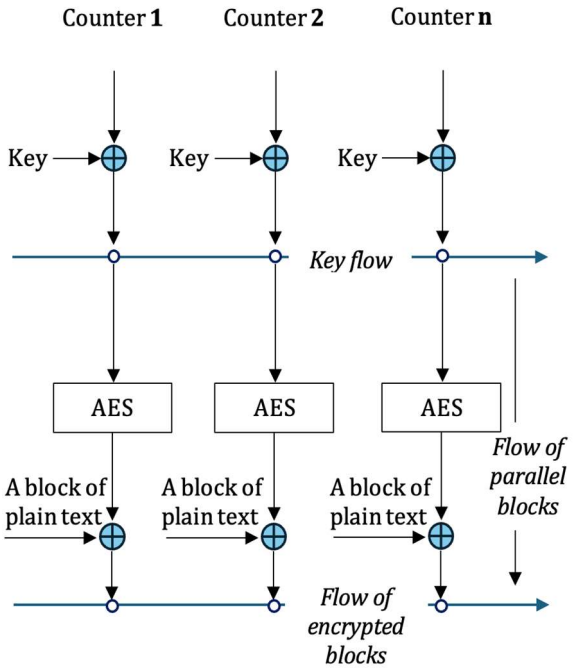


Figure 2: Block parallelization scheme

In this mode, a unique "counter" or "initialization vector" is created for each block, which is combined with the key to generate a unique encryption key. Thus, each block uses a different key, and they can be processed in parallel.

The AES block parallelization algorithm in CTR mode (Fig. 3) involves partitioning into blocks, generating unique counters for each block, encrypting each block independently of the others, and combining the encrypted blocks to produce the final ciphertext.

The key stream is an important part of the encryption process, especially when considering parallel computing to encrypt multiple blocks of data simultaneously. The key stream is used to create unique keys for each block of data. This is achieved by using a counter and key values. The counter value is determined by the stream number that points to a specific data block. The key is formed using the result of the Diffie-Hellman algorithm [26], which ensures the security of key exchange.

- **Parallel computation**: all data blocks compute their keys in a parallel way, which significantly speeds up the encryption process. Each block uses its key stream to generate a unique key. This increases the efficiency of encryption because the blocks can be processed simultaneously without unnecessary delays.

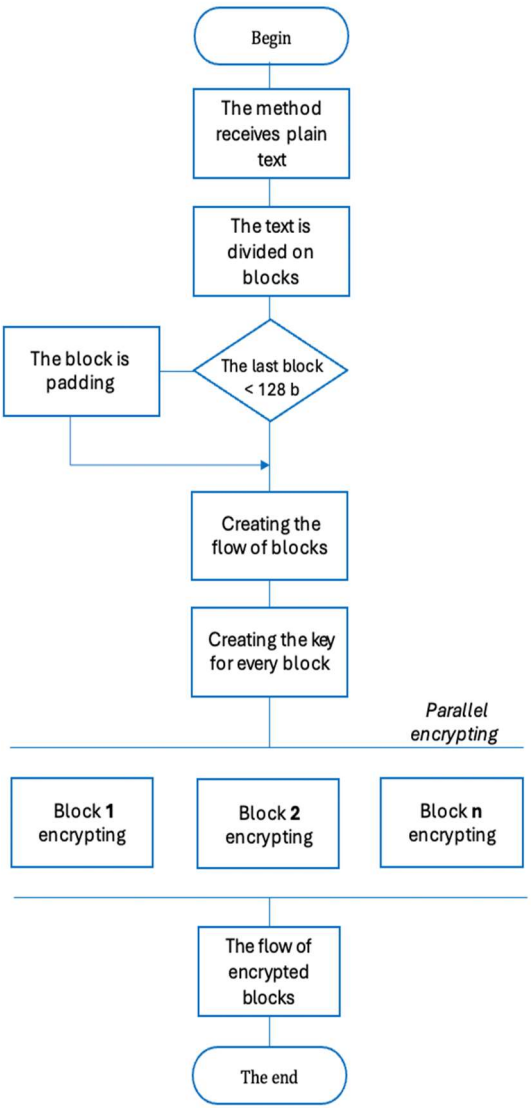


Figure 3: Block diagram of the AES encryption algorithm with block parallelization

- **Storing of encrypted blocks**: when a block completes its work, the encrypted block is saved to a separate thread that functions as a container for collected encrypted blocks. This encrypted block collection stream can be used for further processing or transmission.
- **Creating of the ciphertext**: whet the computation of all blocks is complete, the encrypted blocks are formed into ciphertext. This text can be used for storage, transmission, or other purposes, ensuring the data confidentiality and integrity.
- **Using the Diffie-Hellman algorithm**: the DH algorithm guarantees the security of key exchange by defining a shared secret key between the parties, which ensures the confidentiality of information. The result of this algorithm is used as the basis for creating unique keys for each block of data in the stream.

In general, this process ensures a high level of security and encryption efficiency for parallel data block computation.

5. Programming the block parallelization algorithm

This section describes the process of developing and programming the AES block parallelization algorithm using developed services on the Java Spring Boot platform. These services will interact to securely exchange and process encrypted messages and files.

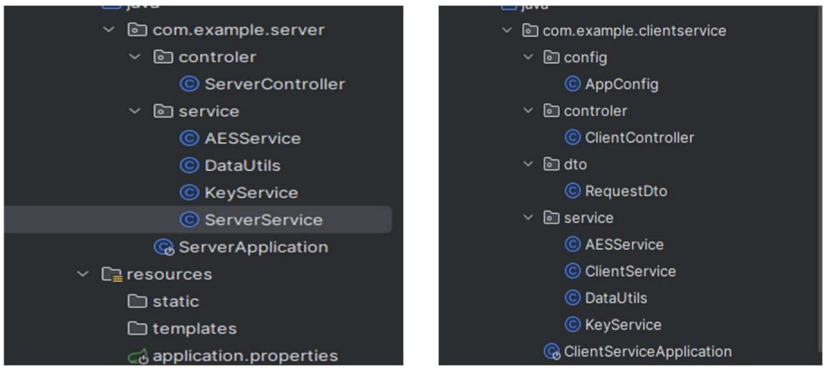
Client Service is responsible for messages encrypting and messages transmitting to the server for further processing. It uses AES and keys to encrypt messages. The encrypted messages are sent to the server for decryption and saving to a file.

Server Service is responsible for receiving encrypted messages and decrypting those messages. The decrypted messages are saved to a file. Server Service is responsible for encrypting files and sending encrypted files to the client.

To implement this functionality, Java Spring Boot, a framework for developing web applications and microservices, is used. It ensures the configuration and efficiency of service deployment.

The proposed approach ensures a secure and efficient mechanism for exchanging encrypted messages and files between the client and the server.

So, two services, the client and the server, interact with a secure exchange of secret keys over an unreliable channel using the Diffie-Hellman algorithm. The structure of the server is shown in Fig. 4a and the structure of the client is shown in Fig. 4b.



a) b)

Figure 4: The structures of developed services

The Server Service diagram of classes is shown in Fig. 5. The developed Server Service consists of such classes:

- **ServerController**: handles HTTP requests; receives and transmits data between the client and other services; initiates calls to other services for data processing.
- **AESService**: implements the AES encryption and decryption algorithm. It is used by the ServerController to ensure the confidentiality of the exchanged data.
- **DataUtils**: contains general methods for data operations.
- **KeyService**: handles keys (generation, storage, and processing) necessary for data encryption and decryption. The ServerController and other services can interact with the KeyService to ensure the security of the exchanged data.
- **ServerService**: receives and processes data from the ServerController; uses the AESService for encryption and decryption, DataUtils for data processing, and KeyService for key management.

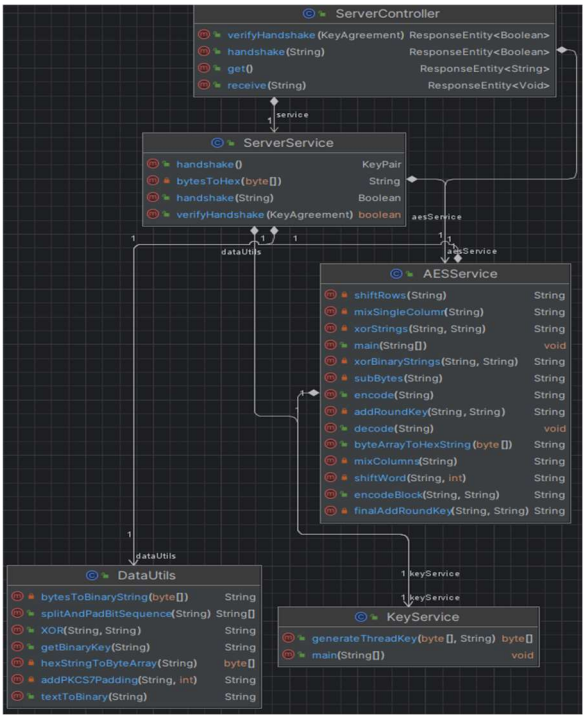


Figure 5: The Server Service diagram of classes

The developed Client Service consists of (the diagram of classes is shown in Fig. 6).

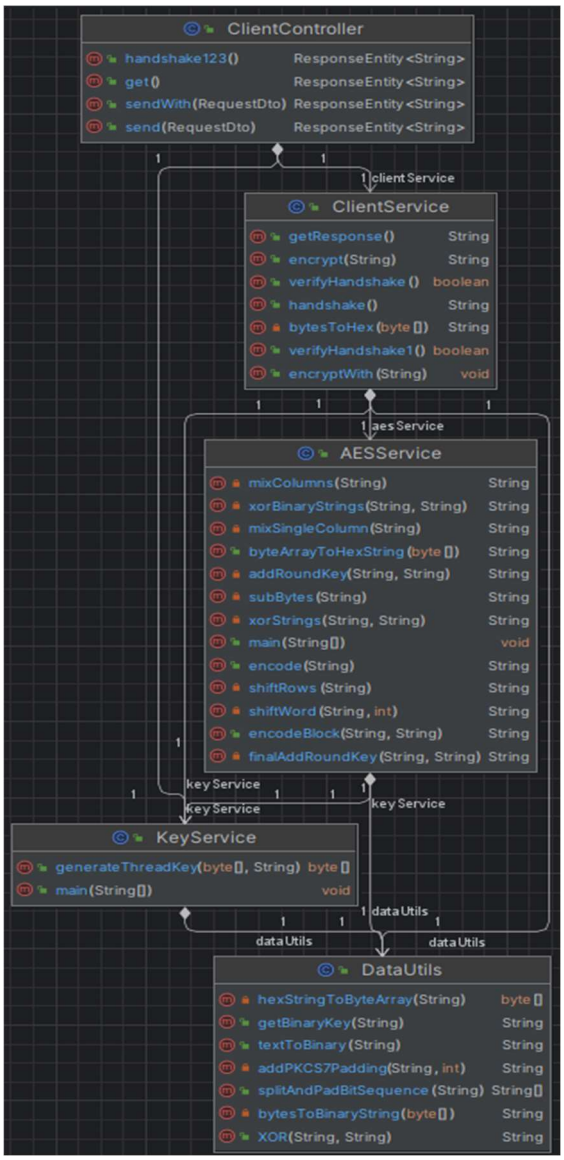


Figure 6: The Client Service diagram of classes

- ClientController: handles HTTP requests from the server; receives and transmits data between the server and other services; initiates calls to other services for data processing.

Implementation of the proposed process of block parallelization in the AES cipher in CTR mode based on two developed services significantly optimizes and improves the operation of the encryption system. The increase in performance is the result of the efficient use of computing resources due to the process of parallelizing the processing of data blocks. Splitting the tasks between two services allows better management of encryption and decryption processes, contributing to the system's scalability needs.

6. Conclusions

6.1. Detailed description of the process

It is assumed that the implementation of block parallelization in AES cipher using CTR mode increases performance during processing a large volume of data. The main idea is: that dividing into parallel blocks allows more efficient use of resources and reduces encryption and decryption time. The hypothesis assumes that because of parallelization implementation: (1) processing speed increases; (2) system scalability grows. The experiment contains a comparative analysis of speed and productivity between systems with and without block parallelization in the AES cipher during the processing of a real data volume.

6.2. Input data and its analysis

For the implementation of an improved AES algorithm, the text data was used. Each data block was defined by a size that corresponds to the AES standard, 128 bits. The considered data volume was determined by the number of blocks that were chosen for encryption. The chosen data volume value corresponded to specific experimental conditions to study the efficiency of parallelization in the algorithm. Input data for the AES algorithm with improved block parallelization came from user input. This considered the variety of algorithm usage scenarios and subjected it to realistic conditions. Algorithm parameters, including block size and data volume, were chosen based on the efficiency check of the improvement.

6.3. Working with data

Different data volumes were taken and checked by the AES algorithm without parallelization and with it to determine the efficiency of the improvement.

To start, check the connection between the two services and generate a key for the algorithm (Fig. 7).

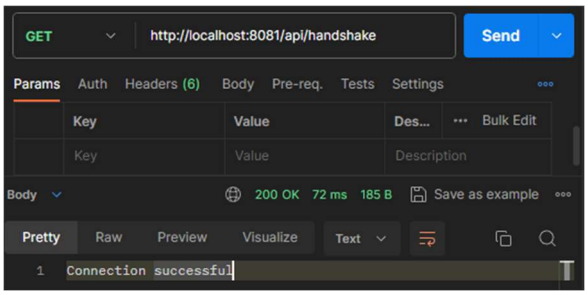
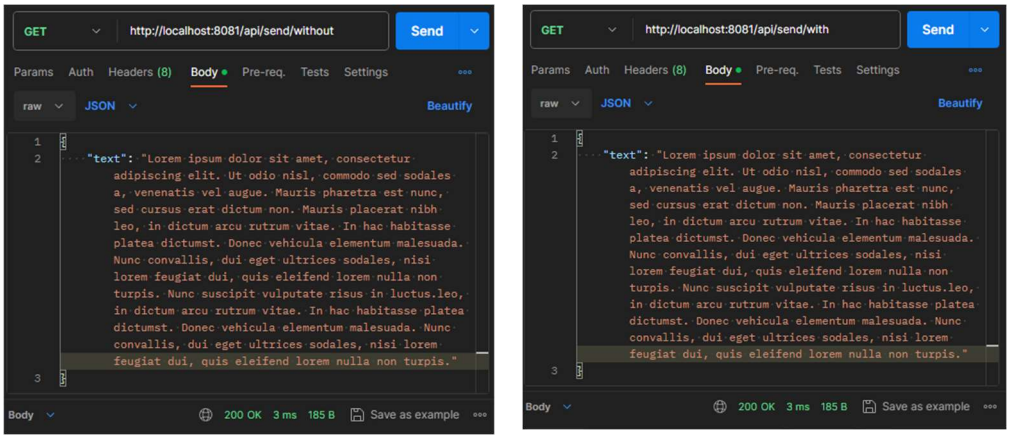


Figure 7: Connecting two services

For the request_1 in the **first test**, small data volumes (less 500 blocks) and standard AES algorithm in CTR mode without parallelization were used (Fig. 8a). The result is 3 milliseconds to encrypt the request_1. Then a request_1 is created with the same data sets, but the algorithm with parallelization is used. (Fig. 8b).

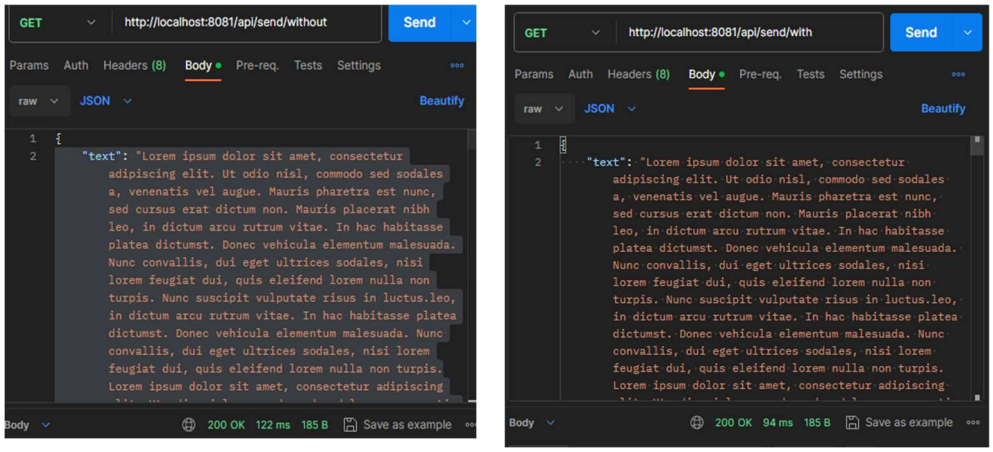


a) b)

Figure 8: Test 1. Requests execution result with small data volumes

The parallelization does not play a significant role for small data (it was processed in 3 milliseconds).

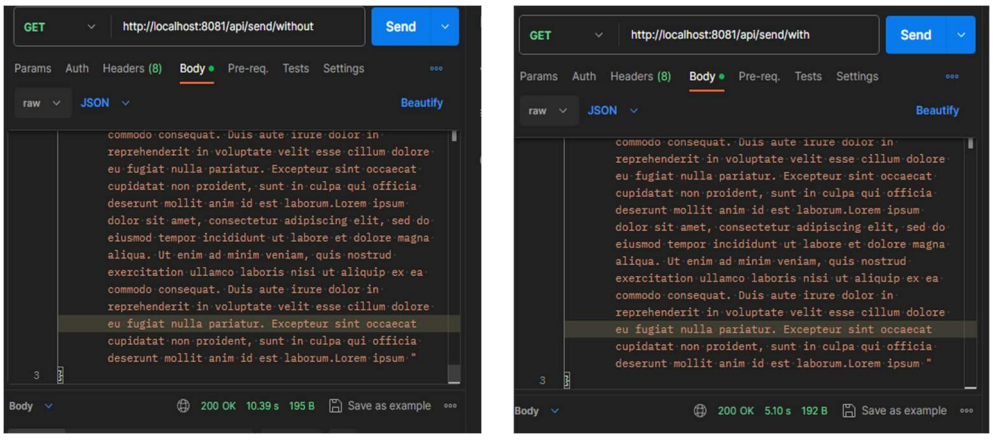
For the request_2 in the **second test**, bigger data volumes were used (10000 blocks) and the standard AES algorithm in CTR mode without (Fig.9a) and with (Fig. 9b) parallelization was used. The request_2 was executed in 94 milliseconds, which is 38 milliseconds less than the previous result.



a) b)

Figure 9: Test 2. Requests execution result with bigger data volumes

For the request_3 in the **third test**, big data volumes were used (1000000 blocks) and the standard AES algorithm in CTR mode without and with parallelization was used. The request_3 was processed for 10.39 seconds (Fig.10a) and 6.05 seconds (Fig. 10b), which is twice as fast as the previous result.



a) b)

Figure 10: Test 3. Requests execution result with big data volumes

6.4. Implementation results

In this section, the results of implementing the AES block parallelization process are shown. The data processing speeds with (Table 1) and without (Table 2) parallelization process are compared.

Table 1
Processing time of the AES algorithm (without parallelization)

Number of blocks	Processing Time of the Request (s)					Average Time (s)
500	0.003	0.004	0.004	0.003	0.003	0.0034
10000	0.122	0.117	0.124	0.122	0.119	0.121
1000000	10.90	10.50	10.54	10.42	10.32	10.384

Table 2
Processing Time of the Improved AES Algorithm (with parallelization)

Number of blocks	Processing Time of the Request (s)					Average Time (s)
500	0.003	0.004	0.003	0.003	0.004	0.0034
10000	0.094	0.089	0.087	0.093	0.090	0.0906
1000000	5.10	4.89	5.03	4.95	4.10	4.98

The speed of encryption and decryption decreases with the data increase.

6.5. Hypothesis Confirmation

The hypothesis about the implementation of block parallelization in the AES cipher in CTR mode will lead to a significant improvement in speed and encryption efficiency in the context of processing large volumes of data. An experiment was conducted that involved testing two algorithms (with and without parallelization) for operation speed and efficiency.

For this experiment, different volumes of data were used, which could be used in the real world. The experiment showed that the hypothesis was confirmed, so block parallelization might be effective not only for the AES algorithm but for any block symmetric algorithm. The experiment showed that block parallelization in AES for big data volumes increases data processing speed by 2 times compared to the non-parallel approach. Parallelization allows to use of multi-core systems more efficiently and accelerates the block processing by 50%. Without parallelization, each block processing is sequential and it takes a longer execution time (10.38 seconds) with processing over 100,000 blocks. Parallelization provides optimal resource usage, reducing the execution time by 2 times to complete the encryption operation compared to the regular algorithm.

7. Conclusions

In this work, a detailed exploration of the block parallelization process when using the Advanced Encryption Standard (AES) cipher is presented. The focus is on the Counter (CTR) mode, which is identified as the most effective strategy for parallelizing blocks, particularly in contexts where parallel processing and high performance are crucial. The benefits of block parallelization, including block independence, ease of implementation and computation, parallel processing capability, and data structure preservation are shown. It also emphasizes the importance of proper management of the counter and key to avoid vulnerabilities.

The process of block parallelization is described in detail, from splitting data into blocks, generating counters, encrypting blocks with AES, incrementing the counter, repeating the process for each block, to generating the ciphertext. The paper highlights that each block is processed independently, which enhances the speed of processing large volumes of information. The development and programming of the AES block parallelization algorithm were performed by developing services on the Java Spring Boot platform. These services interact to securely exchange and process encrypted messages and files. The structure and functions of the developed Server Service and Client Service are explained.

The paper presents an experiment that tests the hypothesis that the implementation of block parallelization in the AES cipher in CTR mode increases performance during the processing of large volumes of data. The experiment involves testing two algorithms (with and without parallelization) for operation speed and efficiency using different volumes of data. The results of the experiment confirm the hypothesis. Block parallelization in AES for big data volumes increases data processing speed by two times compared to the non-parallel approach. Parallelization allows for more efficient use of multi-core systems and accelerates the block processing by 50%. Without parallelization, each block processing is sequential and takes a longer execution time.

The work demonstrates that block parallelization is not only effective for the AES algorithm but for any block symmetric algorithm. It provides optimal resource usage, reducing the execution time by two times to complete the encryption operation compared to the regular algorithm. This process ensures a high level of security and encryption efficiency for parallel data block computation. The proposed approach ensures a secure and efficient mechanism for exchanging encrypted messages and files between the client and the server. The increase in performance is the result of the efficient use of computing resources due to the process of parallelizing the processing of data blocks. Splitting the tasks between two services allows better management of encryption and decryption processes, contributing to the system's scalability needs.

References

[1] A. Gentile, et al., A Performance Analysis of Security Protocols for Distributed Measurement Systems Based on Internet of Things with Constrained Hardware and Open Source Infrastructures, Sensors 24 (2024) 2781. doi: 10.3390/s24092781.

[2] M. Baee, L. Simpson, W. Armstrong, Anomaly Detection in the Key-Management Interoperability Protocol Using Metadata, in IEEE Open Journal of the Computer Society 5 (2024) 156–169. doi: 10.1109/OJCS.2024.3386715.

[3] V. Kampourakis, G. Makrakis, C. Kolias, From Seek-and-Destroy to Split-and-Destroy: Connection Partitioning as an Effective Tool against Low-Rate DoS Attacks, Future Internet 16 (2024) 137. doi: 10.3390/fi16040137.

[4] C. Del-Real, E. De Busser, B. van den Berg, Shielding Software Systems: A Comparison of Security by Design and Privacy by Design Based on a Systematic Literature Review, Computer Law & Security Review 52 (2024) 105933. doi: 10.1016/j.clsr.2023.105933.

[5] S. Sood, N. Hubballi, SlowTrack: Detecting Slow Rate Denial of Service Attacks Against HTTP with Behavioral Parameters, J. Supercomput. 80 (2024) 1788–1817. doi: 10.1007/s11227-023-05453-3.

[6] O. Harasymchuk et al., Generator of Pseudorandom Bit Sequence with Increased Cryptographic Security, Metallurgical and Mining Industry: Sci. Tech. J. 6(5) (2014) 24–28.

[7] O. Yudin, et al., Efficiency Assessment of the Steganographic Coding Method with Indirect Integration of Critical Information (2019) 2019 IEEE International Conference on Advanced Trends in Information Theory, ATIT 2019 - Proceedings, pp. 36-40.

[8] V. Maksymovych, M. Shabatura, O. Harasymchuk, M. Karpinski, et al., Development of Additive Fibonacci Generators with Improved Characteristics for Cybersecurity Needs. Appl. Sci. (Basel) 2022, 12, 1519, doi:10.3390/app12031519.

[9] M. Jang, K. Lee, An Advanced Approach for Detecting Behavior-Based Intranet Attacks by Machine Learning, in IEEE Access 12 (2024) 52480–52495. doi: 10.1109/ACCESS.2024.3387016.

[10] F. Alqahtani, M. Almutairi, F. Sheldon, Cloud Security Using Fine-Grained Efficient Information Flow Tracking, Future Internet 16 (2024) 110. doi: 10.3390/fi16040110.

[11] K. Sasikumar, S. Nagarajan, Comprehensive Review and Analysis of Cryptography Techniques in Cloud Computing, in IEEE Access 12 (2024) 52325–52351. doi: 10.1109/ACCESS.20 24.3385449.

[12] T. Luong, N. Cuong, B. Vo, AES Security Improvement by Utilizing New Key-Dependent XOR Tables, in IEEE Access 12 (2024) 53158–53177. doi: 10.1109/ACCESS.2024.3387268.

[13] K. Rakhimberdiev, A. Bozorov, M. Berdimurodov, Round Key Generation Algorithm Used in Symmetric Block Encryption Algorithms to Ensure the Security of Economic Systems, 7th International Conference on Future Networks and Distributed Systems (ICFNDS '23) (2024) 548–554. doi: 10.1145/3644713.3644794.

[14] C. Boura, P. Derbez, M. Funk, Related-Key Differential Analysis of the AES, IACR Transactions on Symmetric Cryptology 2023(4) (2023) 215–243. doi: 10.46586/tosc.v2023.i4.215-243.

[15] S. Yevseiev, et al., Development of Niederreiter Hybrid Crypto-code Structure on Flawed Codes, Eastern-European J. Enterp. Technol. 1(9(97) (2019) 27–38. doi: 10.15587/1729-4061.2019.156620.

[16] A. Horpenyuk, I. Opirskyy, P. Vorobets, Analysis of Problems and Prospects of Implementation of Post-Quantum Cryptographic Algorithms, in: CEUR Workshop Proceedings, vol. 3504 (2023) 39–49.

[17] V. Luzhetsky, et al., Adaptive Compression Methods of Data Based on Fibonacci Linear Forms, Proc. SPIE 10445, Photonics Applications in Astronomy, Communications, Industry, and High Energy Physics Experiments (2017). doi: 10.1117/12.2280944.

[18] F.-H. Hsiao, S.-W. Chang, Integrating RC6 Stream Cipher to a Chaotic Synchronization System, in IEEE Access 12 (2024) 26308–26323. doi: 10.1109/ACCESS.2024.3364702.

[19] N. Zidarič, et al., The Welch-gong Stream Cipher—Evolutionary Path, Cryptogr. Commun. 16 (2024) 129–165. doi: 10.1007/s12095-023-00656-0.

[20] Y. He, et al., Improved Cube Attacks on Some Authenticated Encryption Ciphers and Stream Ciphers in the Internet of Things, IEEE Access 8 (2020) 20920–20930. doi: 10.1109/ACCESS.2020.2967 070.

[21] O. Kara, Lower Data Attacks on Advanced Encryption Standard, Turkish J. Electrical Eng. Comput. Sci. 32(2(8)) (2024). doi: 10.55730/1300-0632.4072.

[22] J. Qiu, Ciphertext Database Audit Technology Under Searchable Encryption Algorithm and Blockchain Technology, J. Global Inf. Manag. 30(11) (2022) 1–17. doi: 10.4018/JGIM.315014.

[23] A. Sahun, et al., Devising a Method for Improving Crypto Resistance of the Symmetric Block Cryptosystem RC5 Using Nonlinear Shift Functions, Eastern-European J. Enterp. Technol. 5(9(113)) (2021) 17–29. doi: 10.15587/1729-4061.2021.240344.

[24] J. Liang, H. Shahrzad, R. Miikkulainen, Asynchronous Evolution of Deep Neural Network Architectures, Appl. Soft Comput. 152 (2024) 111209. doi: 10.1016/j.asoc.2023.111209.

[25] V. Luzhetsky, et al., Adaptive Compression Methods of Data Based on Fibonacci Linear Forms, Proc. SPIE 10445, Photonics Applications in Astronomy, Communications, Industry, and High Energy Physics Experiments (2017). doi: 10.1117/12.2280944.

[26] A. May, C. Schneider, Dlog is Practically as Hard (or Easy) as DH – Solving Dlogs via DH Oracles on EC Standards, IACR Transactions on Cryptographic Hardware and Embedded Systems 2023(4) (2023) 146–166. doi: 10.46586/tches.v2023.i4.146-166.