Sakarya University Journal of Science

ISSN : 2147-835X
Publisher : Sakarya University

Vol. 28, No. 4, 756-773, 2024
DOI: https://doi.org/10.16984/saufenbilder.1440949

Research Article

Enhancing Industrial Robot Arms Data Security with a Hybrid Encryption Approach

Mustafa Emre Erbil[1], Merdan Özkahraman[1*], Hilmi Cenk Bayrakçı[1]

[1] Isparta University of Applied Sciences, Faculty of Technology, Department of Mechatronics Engineering, Isparta, Türkiye, mail@mustafaemreerbil.com, merdanozkahraman@isparta.edu.tr, cenkbayrakci@isparta.edu.tr
[*]Corresponding Author

ARTICLE INFO

Keywords:
Industrial robot arms
Hybrid encryption
Cryptography
Data security

Article History:
Received: 21.02.2024
Accepted: 14.06.2024
Online Available: 01.08.2024

ABSTRACT

In the context of the widespread application of robotics technology across numerous industrial sectors, the security of data communication in industrial robot arms emerges as a paramount concern. These robotic arms are instrumental in enhancing productivity and safety in a variety of fields, including but not limited to transportation, agriculture, construction, and mining, by automating tasks and reducing human exposure to hazardous conditions. This paper proposes a novel hybrid encryption strategy to fortify the data security of these industrial robot arms, particularly focusing on preventing data breaches during both wired and wireless communications. The suggested encryption framework combines the strengths of Elliptic Curve Cryptography (ECC) for its efficient asymmetric encryption capabilities, ChaCha20 for its rapid and low-energy symmetric encryption, and Poly1305 for ensuring data integrity through its message authentication code (MAC) algorithm. By leveraging these technologies, the paper outlines the development and application of a secure communication protocol, implemented using Python, that guarantees the confidentiality and integrity of data shared among robot arms and between these arms and their control systems. Additionally, the research conducts a comparative analysis between the ECC-based method and the RSA encryption standard, highlighting the efficiency and effectiveness of the proposed hybrid approach through various tests on different data types and sizes. The findings illustrate a marked improvement in safeguarding against potential data leaks, thereby significantly contributing to the enhancement of industrial robot arms' data security. This study not only addresses the pressing need for robust data protection mechanisms in the face of evolving cyber threats but also sets a benchmark for future research in the field of industrial robotics security.

1. Introduction

Under the leadership of technological innovation, the role of robots, especially industrial robot arms, in modern society has been steadily increasing, particularly in areas such as manufacturing and assembly. One of the key reasons for the preference of robots is their ability to move quickly and precisely [1]. Industrial robot arms are designed to automate processes, save labor costs, reduce risky working conditions, and provide support in situations where human labor is inadequate. These robot arms find widespread applications, ranging from factory automation to surgical operations, from the agricultural sector to military applications, and they have been highly successful in increasing the efficiency of business processes while reducing costs. However, the wireless or wired communication capabilities of industrial robot arms pose serious challenges in terms of data security. Security vulnerabilities in communication can potentially lead to unauthorized access to sensitive industrial data, necessitating significant measures to be taken to ensure the communication security of industrial robot arms.

This is an open access paper distributed under the terms and conditions of the Creative Commons Attribution-NonCommercial 4.0 International License.

The data security of industrial robot arms has become critical with the increasing automation and system complexity. However, most previous studies have focused on the security of automation systems [2] and robot operating systems [3], leaving a gap in the area of data security for robot arms. With the aim of addressing this gap, the objective of this study is to propose a hybrid encryption method based on ECC (Elliptic Curve Cryptography), ChaCha20, and Poly1305 to ensure data security during the communication of industrial robot arms.

This hybrid method combines symmetric encryption with a message authentication code algorithm and combines this combination with asymmetric encryption to provide an optimal solution in terms of both security and communication speed. ECC is known for its strong security features while offering advantages in terms of processor intensity and computation time. On the other hand, symmetric encryption methods provide fast communication but may face challenges such as key management. In this study, the ChaCha20-Poly1305 combination created with the security advantages of ECC and its fast, data-integrity-ensuring communication capacity will be used to securely encrypt communication between industrial robot arms and control systems.

Furthermore, a comparison will be made between ECC and the RSA (Rivest-Shamir-Adleman) asymmetric encryption method, elucidating the advantages of ECC. The ECC-ChaCha20-Poly1305 hybrid encryption method developed using the cryptography, Crypto.Cipher, and Crypto.Random libraries within the Python programming language aims to prevent potential data leaks during communication and to protect the confidentiality and integrity of transmitted data. The results obtained through this methodology will represent a significant step in ensuring data security for industrial robot arms.

Industrial robot arms have become an indispensable part of modern production processes. These robot arms, complex devices created by the combination of mechanical and control systems, can automate various tasks and enhance the efficiency of the production process. They can replace human labor in complex, repetitive, or hazardous tasks [4]. Equipped with advanced control systems and precision sensors, industrial robot arms can be programmed to learn, execute, and optimize the movements and processes required to complete specific tasks [5]. Used in various industrial sectors such as automotive, electronics, food and beverage, pharmaceuticals, these robot arms complete operations quickly and consistently, increasing overall production efficiency. Additionally, the use of robot arms in hazardous or challenging work conditions enhances workplace safety, reducing work accidents and injuries [6].

The application areas of industrial robot arms are extensive and diverse. Tasks such as material handling, assembly, welding, paint spraying, packaging, and quality control can be automated with the assistance of robot arms. The use of this technology accelerates the workflow, increases efficiency and safety, reduces production costs, and enhances product quality [7]. Robot arms enable businesses to produce faster and more effectively while safeguarding worker health. Robot arms equipped with various sensors and artificial intelligence technology can perceive their surroundings and move independently, adapt to various applications, and successfully complete complex tasks [8].

With advancing technology, the capabilities of industrial robot arms are increasing, making it easier to complete more complex and precise operations with their assistance. This facilitates the automation of the production process and reduces labor costs. The use of robot arms enhances production process flexibility, enabling businesses to respond more quickly and effectively to market demands [9]. Furthermore, robot arms efficiently complete tasks, leading to energy savings and waste reduction, contributing to environmental sustainability [10].

2. State of the Art

Encryption is a method employed for ensuring the secure protection of information and data security. This method transforms information into a format that can only be accessed or understood by authorized individuals. This transformation process is carried out using one or more private keys, resulting in the conversion of plain text, which is the understandable form of information, into unreadable encrypted text [11]. Encryption technology holds vital importance, particularly in the realms of information security and data privacy. Encryption methods are generally categorized into two main categories: asymmetric encryption and symmetric encryption. Both of these methods play a critical role in safeguarding data.

Wireless and wired communication refer to communication conducted over networks that are commonly used today, encompassing mobile devices, sensor networks, Internet of Things (IoT) devices, and broadband connections, among others. Both of these communication types face a range of security challenges. Wireless communication, in particular, becomes more vulnerable to attacks due to the transmission of data in a physically unprotected environment, emphasizing the significance of data security and privacy [12].

Encryption methods are widely employed in both communication systems to ensure data security. Encryption protects data against unauthorized access by transforming it from a comprehensible state into an encrypted form. Encryption methods used in wireless and wired communication systems are designed to meet security requirements and ensure data integrity, confidentiality, and identity authentication. Hybrid encryption is a security strategy commonly preferred in both systems. In this method, asymmetric encryption, symmetric encryption, and message authentication code algorithms, such as ECC, ChaCha20, and Poly1305, are used together to combine the advantages of each algorithm, resulting in a stronger security solution [13].

Asymmetric encryption (ECC) is used for operations like key exchange and identity verification, symmetric encryption (ChaCha20) is used for data encryption and decryption, and the message authentication code algorithm (Poly1305) is used to enhance symmetric encryption [14]. In this way, the hybrid encryption approach provides a secure key exchange while offering a fast and efficient solution for data encryption. Additionally, this method examines and elucidates the differences between ECC and RSA asymmetric encryption methods, which have been frequently compared in recent times, to explain why ECC encryption method is the preferred choice.

The use of hybrid encryption in both wireless and wired communication systems offers significant advantages in terms of data security. Asymmetric encryption algorithms reduce the risk of unauthorized access during data transfer by providing a secure key exchange. Simultaneously, symmetric encryption algorithms offer rapid data encryption and decryption, facilitating high-performance data transmission. Therefore, the hybrid encryption approach strikes a balance between security and performance in wireless and wired communication systems [15].

2.1.Asymmetric encryption

Asymmetric encryption, also known as public-key cryptography, is a method that enables secure data transfer using two keys. These two keys form a pair, one being public and the other private. The public key is shared openly with everyone, while the private key is known only to the recipient of the message. Data is encrypted using the public key and can only be decrypted using the matching private key. This ensures secure data transmission but is computationally more expensive. Asymmetric encryption is widely used today in many security protocols and applications, particularly due to its ability to provide secure data transmission [16]. Asymmetric encryption relies on the complex structure of mathematics, and its security is based on the difficulty of solving specific mathematical problems. It is a combination of mathematical concepts that form the foundation of encryption, allowing its use in various applications [17].

Asymmetric encryption is based on the following mathematical principles:

- One-way Functions; These are mathematical functions that can be easily calculated in one direction but are difficult to reverse. For example, multiplying two large prime numbers is easy, but factoring the product back into its prime factors is challenging.

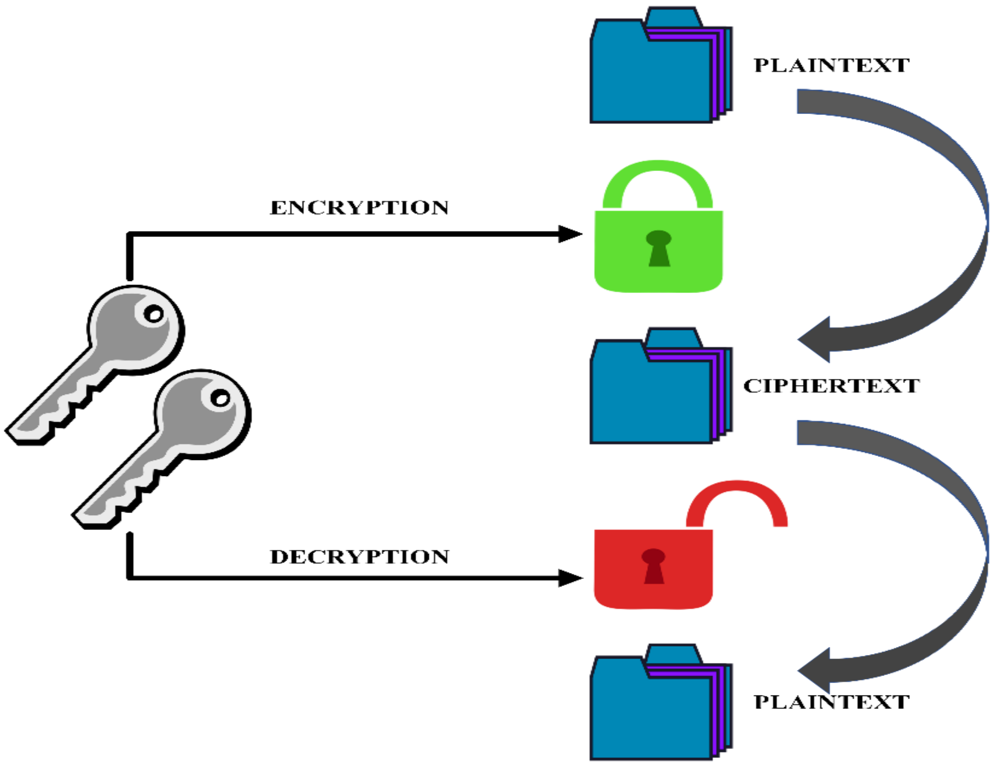

Figure 1. Asymmetric encryption scheme.

- Modular Arithmetic; This involves processing numbers within a limited system defined by a modulus value. Modular arithmetic is commonly used in encryption operations because some operations are difficult to reverse.
- Distributed Systems; Asymmetric encryption allows both keys to operate independently. The public key is accessible to everyone, while the private key is known only to authorized users.

These principles enable the application of asymmetric encryption in areas such as data security, digital signatures, and secure electronic transactions [18]. Figure 1 illustrates the asymmetric encryption scheme.

2.1.1. RSA (Rivest-Shamir-Adleman)

RSA, an asymmetric encryption algorithm proposed by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977, employs two keys: a private key and a public key. The public key is used for data encryption, while the private key is utilized for data decryption. RSA finds common application in digital signatures and key exchange protocols [19]. RSA encryption is grounded in the principle of one-way functions in asymmetric cryptography. Its security hinges on the complexity of factoring large prime numbers. The RSA encryption and decryption process consists of key generation, encryption, and decryption steps [20]. In the key generation phase, two large prime numbers (p) and (q) are initially selected as a fundamental security measure. This selection is at the core of the algorithm's security. The product of these prime numbers (n) defines the modulus employed in encryption and decryption operations, as expressed in Equation 1.

$$n = p \times q \tag{1}$$

Euler's totient function is calculated, as depicted in Equation 2, and is an integral part of security based on the difficulty of factoring prime numbers.

$$\varphi(n) = (p-1) \times (q-1) \tag{2}$$

The public key (e) is typically chosen as a small, commonly used prime number such as 65537, and is employed for encryption. The private key (d), calculated as specified in Equation 3, ensures the secure decryption of data.

$$e \times d \equiv 1 \ (mod\ \varphi(n)) \tag{3}$$

The encryption process involves encrypting the message (M) using the public key (e, n), expressed in Equation 4. This process ensures data protection and security against unauthorized access.

$$C = M^e \pmod n \tag{4}$$

The decryption process, on the other hand, decrypts the encrypted message (C) using the private key (d, n), as indicated in Equation 5. This enables only authorized recipients to access the original content of the message.

$$M = C^d \pmod n \tag{5}$$

These processes of RSA establish a robust security foundation based on the complexity of factoring large numbers, making it one of the fundamental security tools in the digital world.

2.1.2. ECC (Elliptic Curve Cryptography)

Elliptic Curve Cryptography (ECC) is an advanced asymmetric encryption technology commonly employed in digital signature generation and encrypted data exchange. This method operates using two keys: a private key, known only to the individual, and a public key, openly shared with everyone.

While the public key is influential in data encryption, the corresponding private key is solely used during the decryption process. ECC's robustness relies on the mathematical operations involving elliptic curves, which are considered one-way functions. This characteristic allows for superior security with shorter key lengths. ECC's essence lies in the ease of performing these mathematical operations while making it considerably challenging to calculate their inverses, especially scalar multiplication on elliptic curves.

Thanks to these features, ECC ensures secure key distribution and data integrity, rendering it a popular choice in wireless communication and smart device technologies. ECC's unique advantages include its energy efficiency and the ability to provide high-level security and performance even in devices with limited computational capacity. The mathematical complexity of operations defined on elliptic curves produces flexible secure solutions that meet modern encryption requirements [21]. The ECC encryption and decryption process consists of key generation, encryption, and decryption steps [22]. In the key generation phase, an elliptic curve and a starting point (G) are initially selected. Then, a random number (k) is chosen as the private key. Finally, the public key is calculated as the product of (k) and (G), as shown in Equation 6.

$$P = k \times G \tag{6}$$

During the encryption phase, the message (M) is encrypted using the recipient's public key (P) and a randomly chosen number (r). An interim point is calculated as demonstrated in Equation 7.

$$R = r \times G \tag{7}$$

Another point is determined using the recipient's public key, as depicted in Equation 8.

$$S = r \times P \tag{8}$$

The message (M) is encrypted as a function of point (S). In the decryption phase, the encrypted message (C) is decrypted using the private key (k). A multiplication of the private key (k) and the point (R) received from the sender is computed, as illustrated in Equation 9.

$$T = k \times R \tag{9}$$

The multiplication given in Equation 9 is equivalent to the point (S) created during encryption. Message (M) is then decoded using (T). Both systems are founded on mathematical complexities: factoring large prime numbers in RSA and finding the inverse of scalar multiplication on elliptic curves in ECC. These attributes make both systems powerful tools in modern cryptography.

From a security and efficiency perspective, ECC is particularly suitable for devices with constrained energy consumption and computational capacity. Hence, ECC is a preferred method in blockchain technologies, smart contracts, and numerous advanced cryptographic applications.

2.2. Symmetric encryption

Symmetric encryption, also known as symmetric cryptography, is a encryption method where both encryption and decryption processes are performed using the same key. This method requires less computational power compared to asymmetric cryptography, making it faster. However, the weakness of symmetric encryption lies in the necessity of securely sharing the key between two parties. If the key falls into the hands of malicious individuals, the encrypted data can be easily decrypted. Symmetric encryption is particularly advantageous when there is a need to encrypt a large amount of data quickly [23]. Figure 2 illustrates the Symmetric Encryption Scheme.

The fundamental principles of symmetric encryption are as follows:

- Key Protection: The security of encryption relies on the careful protection of the key by both the sender and the recipient. The compromise of the key under any circumstances can jeopardize the security of the entire encryption system.

- Processing Speed and Efficiency: Symmetric encryption offers a fast and efficient solution for encryption and decryption processes due to the use of the same key for both operations. This feature makes it ideal for scenarios that require the processing of large volumes of data.
- Operation on Data Blocks: These systems typically divide data into fixed-size blocks and encrypt each block separately. This block-based approach ensures the organized and secure processing of data [24].

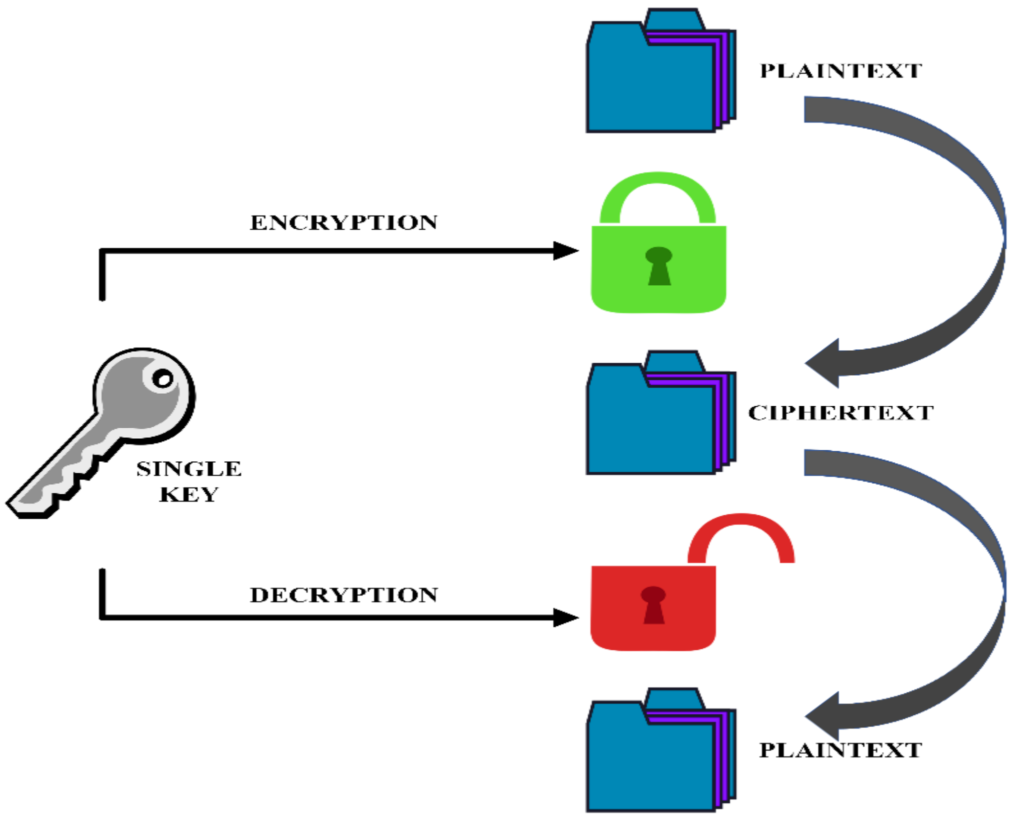

Figure 2. Symmetric encryption scheme

Thanks to these fundamental principles, symmetric encryption has become an effective tool in various fields, especially in internet security and network encryption. This method offers speed and efficiency while also requiring the secure management of keys, making it a domain that demands technical expertise.
In the field of data protection, symmetric encryption plays a fundamental role,

encompassing a variety of algorithms. Each algorithm offers unique characteristics and benefits, making them suitable for different requirements. For instance, the Advanced Encryption Standard (AES) is renowned for its wide acceptance as a security standard and its robust protection features.

Most symmetric encryption methods have emerged as alternatives to AES. Blowfish is ideal for small-scale projects due to its ease of adaptation. Twofish is considered a strong competitor to AES due to its flexibility in key management. RC6, with its unique structure, is also a strong contender against AES. Serpent, like Twofish and RC6, is another rival to AES, known for its high security performance. Camellia, as a Japan-based encryption standard, offers high security and speed. Salsa20 is known for its speed and efficiency, while ChaCha20, as an improved version of Salsa20, provides superior security and speed features [25-27]. The comparison of encryption methods in terms of performance and security has been thoroughly investigated in a study by Indla et al. [28]. Table 1 provides a Comparison of Symmetric Encryption Methods. The table presented in the study compares various algorithms based on key length, block size, processing speed, and security levels. By examining only the test results of symmetric encryption methods, the table clearly shows the advantages and disadvantages of these methods relative to each other.

These comparisons form an important guide in the selection of symmetric encryption methods. Selecting the appropriate encryption method according to different application requirements and security needs is of critical importance.

This table provides a comprehensive comparison of various symmetric encryption methods, highlighting key aspects such as security level, performance speed, key length options, block size, implementation complexity, and availability status. The table serves as an invaluable resource for understanding the differences and suitability of each encryption method for specific applications. It distinctly outlines how each algorithm, from widely used ones like AES to more niche options like Blowfish and Twofish, varies in terms of efficiency, security robustness, and practical deployment considerations, thereby aiding in the informed selection of encryption techniques for diverse security needs.

ChaCha20 is an advanced symmetric key encryption algorithm that encrypts data using a series of cyclic operations. Designed by Daniel J. Bernstein, this algorithm offers both high speed and security [29]. One of the key advantages of ChaCha20 is its strong security while being able to operate efficiently even on low-cost hardware. These features make it an ideal solution for systems that prioritize security and have limited processing power, such as industrial robotic arms.

Table 1. Comparison of Symmetric Encryption Methods [28]

Algorithm	**Security**	**Performance**	**Key Length**	**Block Size**	**Implementation Complexity**	**Availability**
AES	High	Fast	128, 192, or 256 bits	128 bits	Low	Widely Available
Blowfish	Medium	Fast	32 to 448 bits	64 bits	Low	Widely Available
Twofish	High	Fast	128 to 256 bits	128 bits	Moderate	Limited
RC6	High	Fast	128, 192, or 256 bits	128 bits	Moderate	Limited
Serpent	High	Moderate	128 to 256 bits	128 bits	High	Limited
Camellia	High	Fast	128, 192, or 256 bits	128 bits	Moderate	Limited
Salsa20	High	Fast	128 or 256 bits	64 or 128 bits	Low	Limited
ChaCha20	High	Fast	256 bits	64 or 128 bits	Low	Limited

Additionally, it seamlessly integrates with MAC algorithms. Therefore, it was chosen as the symmetric encryption algorithm for the project.

ChaCha20 relies on complex cyclic operations and sequence generation. The fundamental steps of this algorithm include key and nonce initialization, creating an initial state matrix, the cyclic operation process, and encryption and decryption. The steps of ChaCha20 can be outlined as follows:

- Key: A 256-bit key (K) is used.
- Nonce and Block Counter: A 64-bit nonce and a block counter are used for security.
- ChaCha20 operates on a 512-bit state matrix.
- The initial state consists of a "magic constant" (σ), the key (K), block counter, and nonce.
- The state matrix is processed for a specified number of rounds.
- Each round involves addition, XOR operations, and bit shifting; Addition is performed *mod* 2^{32}, XOR operation involves bitwise XOR of two words, and bit shifting plays a crucial role in data encryption.
- Encryption is achieved by adding and XORing the processed state matrix with the original state matrix.
- Decryption is performed in the reverse manner of encryption.

These mathematical characteristics and its relationship with MAC algorithms make ChaCha20 a secure and efficient encryption option. Additionally, its low hardware requirements and high parallelizability allow for effective implementation across various platforms [30].

Message Authentication Code (MAC) algorithms play a critical role in ensuring data integrity and authentication. These algorithms are used to verify that a message has reached its destination without being altered or subject to unauthorized interference. The primary function of MAC algorithms is to produce a fixed-size output using a message and a secret key. This output serves as a digest of the message and is used to verify the integrity of the message. The security of MAC algorithms relies on the secrecy of the secret key and the cryptographic strength of the algorithm [31, 32]. The characteristics of MAC algorithms are as follows:

- Integrity and Identity Verification: MAC algorithms guarantee both the integrity of the message and the authenticity of its source. This helps detect possible alterations or unauthorized interventions during data transmission.
- Secret Key Usage: MAC algorithms produce a digest of the message using a secret key, allowing both the sender and receiver to verify it.
- Collision Resistance: An effective MAC algorithm makes it difficult to find two different messages that produce the same MAC value. This feature enhances the algorithm's reliability.
- Application Variety: MAC algorithms are used across a wide range of applications, from financial transactions to network security.

Popular MAC algorithms include HMAC, CMAC, and Poly1305. These algorithms can serve various requirements and security levels. These fundamental characteristics of MAC algorithms play a crucial role in data security and authentication processes, making their usage an essential part of preserving data integrity and ensuring secure communication in the digital world.

Poly1305 is a MAC (Message Authentication Code) algorithm designed to ensure the integrity and authenticity of messages in secure communications. Developed by Daniel J. Bernstein, this algorithm sets high standards for both speed and security [33]. Poly1305 is particularly used in encrypted data transmission and works in conjunction with encryption algorithms such as ChaCha20 to secure data integrity [34, 35]. The fundamentals of Poly1305 are as follows:

- Key Generation and Usage: Poly1305 uses a 256-bit key, consisting of two parts: a 128-bit key and a 128-bit nonce. The key is used in the calculation of the

MAC value, while the nonce must be unique for each message.

- MAC Value Computation: Poly1305 computes the MAC value for a given message. This process is repeated for each block of the message, and the results are combined. Polynomial multiplication is performed on each block using arithmetic m*od* *(2^{130}-5*). This forms the basis of Poly1305's security.
- Verification Process: The recipient calculates the MAC value for the received message using the same key and nonce. If the calculated MAC value matches the transmitted MAC value, the integrity and authenticity of the message are verified.

Poly1305 is a strong MAC algorithm in terms of security. Proper use of the key and nonce provides a high level of security. Poly1305 is designed to operate quickly on modern processors, making it particularly suitable for real-time applications. It finds a wide range of applications for preserving data integrity, especially in encrypted communications and network security. These features make Poly1305 a popular choice in modern cryptography applications. The mathematical foundations and practicality of the algorithm provide an effective and reliable data authentication solution.

2.3. Hybrid encryption

Hybrid encryption combines asymmetric and symmetric encryption techniques to create a robust security protocol. This method blends the advantages of symmetric and asymmetric encryption methods to provide a comprehensive security solution.

The ChaCha20-Poly1305 Combination is a merger of two powerful cryptographic algorithms designed by Daniel J. Bernstein. This combination is used as a vital element of data security in hybrid encryption systems. While ChaCha20 is an efficient symmetric encryption algorithm, Poly1305 serves as a reliable Message Authentication Code (MAC) algorithm. The combination of these two algorithms offers an effective solution for both encryption and data integrity security [36-38]. The fundamentals of the ChaCha20-Poly1305 Combination are as follows:

- Enhanced Security and Performance: ChaCha20 offers high standards of speed and security, especially functioning effectively on low-cost hardware. Poly1305 ensures message integrity and authenticity by using a unique nonce for each message.
- Advantages of the Combination: This combination brings together ChaCha20's fast encryption capabilities and Poly1305's robust data verification mechanisms. Both encryption and MAC operations are performed using the same key set, simplifying the process and enhancing security.
- Application Areas: ChaCha20-Poly1305 is often preferred in applications requiring encrypted communication, network security, and data integrity. It provides high-level security and performance, particularly in devices with limited energy consumption and processing capacity.

ChaCha20 encrypts data, while Poly1305 calculates the MAC value over the encrypted data. ChaCha20 relies on XOR operations and bit shifts, whereas Poly1305 provides security through arithmetic *mod (2^{130}-5)*. This hybrid system stands out in the field of modern cryptography, particularly as a product of Bernstein's work. The efficiency of ChaCha20-Poly1305 offers a powerful and flexible solution that meets complex encryption requirements.

ChaCha20-Poly1305-ECC Encryption Method is an advanced hybrid encryption technique that combines three powerful technologies of modern cryptography. This combination integrates the ChaCha20 and Poly1305 algorithms developed by Daniel J. Bernstein with Elliptic Curve Cryptography (ECC) [39-42]. This triple combination offers an excellent balance in data security: the fast and efficient encryption of ChaCha20, the robust message authentication of Poly1305, and the superior security and efficiency features of ECC. ECC, especially with short key lengths, provides high-security levels. Mathematical operations on elliptic curves enhance encryption security. While ChaCha20 effectively encrypts data, Poly1305 is used to ensure the integrity and authenticity of this encrypted data. ChaCha20 stands out with low hardware requirements and high parallelizability. ECC is advantageous, particularly in fields like blockchain technologies and smart contracts. This hybrid system combines the strengths of each algorithm to provide a comprehensive security solution, offering an integrated approach in data encryption, authentication, and key management processes. The Encryption and Decryption Processes of ChaCha20-Poly1305-ECC Combination are as follows:

- ECC is used to establish a secure key exchange. During encryption and decryption operations, elliptic curve-based mathematical operations are used to securely create a shared common key between parties. This common key then serves as the basis for symmetric encryption.
- ChaCha20 is a symmetric encryption algorithm. The common key generated with ECC is used by the ChaCha20 algorithm for encrypting or decrypting data. ChaCha20 is designed to efficiently and securely encrypt data.
- Poly1305 is a Message Authentication Code (MAC) algorithm used in conjunction with ChaCha20 to verify the integrity and authenticity of encrypted data. Poly1305 produces a tag appended to the encrypted data, allowing for the verification of whether the data has been manipulated during decryption.

3. Material and Method

This study focuses on the comparison of asymmetric encryption methods as applications, because asymmetric encryption, which uses two different keys (a public and a private one), enhances security and data integrity in the transmission of sensitive data. Unlike symmetric encryption methods, asymmetric encryption has a structure that does not require the same key to be present at both ends of the communication [43]. Therefore, examining the advantages of asymmetric encryption for systems containing sensitive data, such as industrial robotic arms, is the main purpose of this study.

To compare asymmetric encryption methods, the RSA encryption method has been chosen. The proposed hybrid encryption algorithm has been compared with the same encryption method's algorithm that uses RSA instead of ECC. The reason for choosing RSA as the asymmetric encryption method to be compared is that RSA is a long-established, reliable asymmetric encryption method in the field of cryptography [44]. The RSA Algorithm is known for its security comparable to the ECC encryption method. Additionally, RSA stands out with its key generation process based on complex mathematical calculations and the use of large numbers. In contrast, ECC offers high security with shorter key sizes, making it more efficient in terms of energy and processing power. This comparison is critical in evaluating the performance of the proposed hybrid encryption method [45].

The use of .bin files in the comparison of the proposed hybrid encryption algorithm and the algorithm with RSA instead of ECC is due to the fact that such files typically contain large, unstructured data. This allows for testing the effectiveness of encryption algorithms on large and unstructured data sets.

The sizes of 100 MB, 500 MB, and 1 GB represent a broad range of data sizes and are important in assessing how encryption algorithms perform across various data magnitudes. These sizes are used to compare the processing times and efficiencies of the algorithms on large and small data sets. This choice is a standardization approach to understand the overall performance and scalability of the algorithms. Particularly, large file sizes (like 1 GB) more clearly reveal the processing times and efficiencies of the algorithms on large data sets, an important factor in understanding how the algorithms perform in real-world scenarios.

Testing the proposed hybrid encryption algorithm on different file types (MP4, PDF, TXT) with varying sizes such as 50, 100, and 200 MB allows for a detailed examination of how encryption methods perform relative to changing data sizes. The use of 50 MB is due to it being one of the standard sizes frequently used in encryption algorithm performance tests. For example, a study has compared the encryption and decryption times of symmetric encryption algorithms for files of 3 MB and 50 MB [46]. The reason for using 100 to 200 MB is to evaluate the scalability of the algorithms and their performance under larger data loads, while not deviating too far from the 50 MB size.

3.1. ChaCha20-Poly1305-ECC algorithm

In this hybrid encryption algorithm developed using the Python language, the following libraries, modules, and packages have been used:

- "cryptography.hazmat.primitives.asymmetric.ec"
- "cryptography.hazmat.primitives.serialization"
- "cryptography.hazmat.primitives.kdf.hkdf"
- "Crypto.Cipher.ChaCha20_Poly1305"
- "Crypto.Random"

A private key is generated using ECC (Elliptic Curve Cryptography) in this hybrid encryption algorithm. This process, which is a form of asymmetric encryption, results in both a private key and a public key. The generated private key is exported in PEM format and saved to a file. This key possesses the authority to decrypt data and should be securely stored. During the testing phase, the data at the specified path is read, representing the original data to be encrypted. A shared key for the ChaCha20-Poly1305 symmetric encryption algorithm is derived from the common key generated using ECC with ECDH (Elliptic Curve Diffie-Hellman) and HKDF (HMAC-based Key Derivation Function). A ChaCha20-Poly1305 encryption object is created, and the data is encrypted using the encrypt_and_digest method. This process results in encrypted data and a tag (MAC). The encrypted data, nonce, and tag are written to a file named "encrypted_data.bin." This file is used to store and securely transmit the encrypted data.

Decryption is performed in the reverse order of the encryption process. Firstly, the private key is read from the file. The encrypted data is read from the file, and the encryption applied by ChaCha20-Poly1305 is deciphered using the key. If the deciphered data matches the generated tag, the integrity of the data is verified, and the decrypted data is presented to the user. This process ensures the secure transfer of data and restricts access to authorized individuals. The hybrid encryption method combines various encryption techniques to safeguard the security and privacy of data. The flowchart for this method is provided in Figure 3.

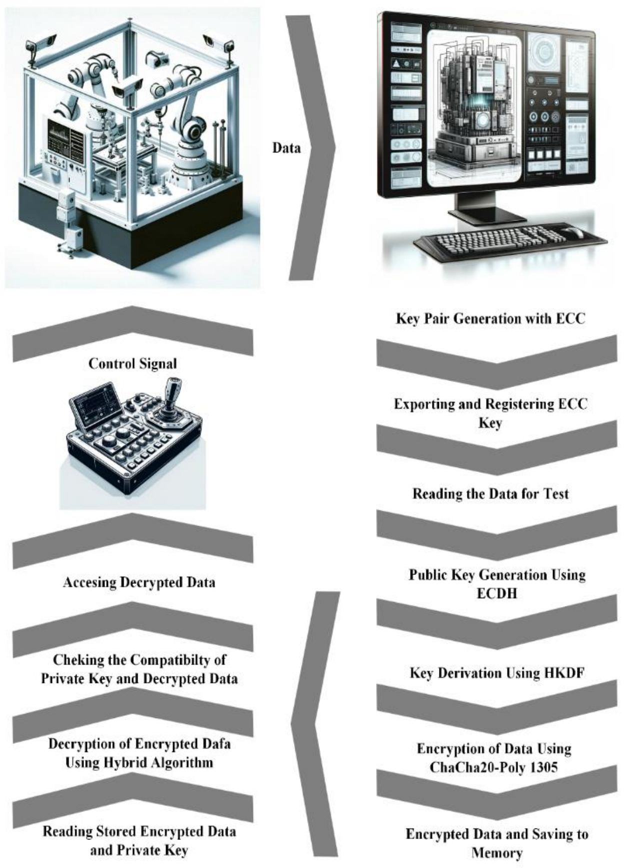
Data
Key Pair Generation with ECC
Exporting and Registering ECC Key
Reading the Data for Test
Public Key Generation Using ECDH
Key Derivation Using HKDF
Encryption of Data Using ChaCha20-Poly 1305
Encrypted Data and Saving to Memory
Reading Stored Encrypted Data and Private Key
Decryption of Encrypted Dafa Using Hybrid Algorithm
Cheking the Compatibilty of Private Key and Decrypted Data
Accesing Decrypted Data
Control Signal

Figure 3. ChaCha20-Poly1305-ECC Encryption-Decryption Flowchart

3.2. ChaCha20-Poly1305-RSA algorithm

In this hybrid encryption algorithm developed using the Python language, the PyCryptodome library is utilized, including the following modules:

- "Crypto.PublicKey.RSA"
- "Crypto.Cipher.PKCS1_OAEP"
- "Crypto.Cipher.ChaCha20_Poly1305"
- "Crypto.Random"

To begin, a 2048-bit RSA key pair is generated. This is achieved using the RSA algorithm, which is an asymmetric encryption method, resulting in both a private key and a public key. The generated private key is exported in PEM format and saved to a file. This key possesses the authority to decrypt data and must be securely stored. To read the data to be tested, a file at the specified path is accessed. This data represents the original data to be encrypted. A key and nonce are generated for the ChaCha20-Poly1305 symmetric encryption algorithm. These key and nonce values are used for encrypting the data. Subsequently, a ChaCha20-Poly1305 encryption object is created, and the data is encrypted. This process results in encrypted data and a tag (MAC). An RSA encryption object is then created, and the ChaCha20 key is encrypted with it. This step ensures that only the owner of the private key can decrypt the data. The encrypted ChaCha20 key, nonce, encrypted data, and tag are written to a file. This file is used for storing the encrypted data and securely transmitting it. Decryption is performed in the reverse order of the encryption process. Firstly, the private key is read from the file and loaded using RSA. The encrypted data is then read from the file, and the key is decrypted using RSA. Subsequently, this decrypted key is used to decrypt the data with ChaCha20-Poly1305. If the decrypted data matches the generated tag, the data's integrity is verified, and the decrypted data is presented to the user. This process ensures the secure transfer of data and restricts access to authorized individuals. The hybrid encryption method combines various encryption techniques to safeguard the security and privacy of data.

4. Results

Robotic systems have the capability to communicate with various data sources such as sensor readings, command instructions, visual media, and mapping data. When performing actions utilizing these capabilities, the algorithm's execution speed plays a crucial role. Data that originates from a source may experience delays if it is encrypted and then decrypted before reaching its destination, instead of directly reaching the target. Therefore, the ChaCha20 symmetric encryption method has been employed for this purpose. In conjunction with ChaCha20, Poly1305 is used to ensure data integrity.

Asymmetric encryption algorithms tend to operate slightly slower than symmetric encryption algorithms. Hence, when choosing among asymmetric encryption methods, regardless of the degree of difference, it is essential to select the algorithm that operates more efficiently. In this test, the combination of ChaCha20-Poly1305 is compared with ECC, which is recommended for use, and RSA, one of the most commonly used asymmetric encryption methods. The test involves encrypting and then decrypting data in .bin format with sizes of approximately 100 MB, 500 MB, and 1 GB, using the ChaCha20-Poly1305 combination with RSA and ECC methods. The timing test was conducted using the "time" library built-in Python. The test results for the ChaCha20-Poly1305-ECC and ChaCha20-Poly1305-RSA algorithms are presented in Table 2.

Table 2. The test results for the ChaCha20-Poly1305-ECC and ChaCha20-Poly1305-RSA algorithms

Algorithm	Processing Time for 100 MB (s)	Processing Time for 500 MB (s)	Processing Time for 1 GB (s)
ChaCha20-Poly1305-ECC	0.89412	3.36814	6.49236
ChaCha20-Poly1305-RSA	1.63905	4.09316	8.13015

According to the results presented in Table 2, it is observed that the ChaCha20-Poly1305-ECC algorithm encrypts and decrypts data faster than the ChaCha20-Poly1305-RSA algorithm. In summary, it has been observed that the ECC asymmetric encryption method operates faster than the RSA asymmetric encryption method.

It is well-known that robotic systems have numerous functions such as communicating with sensor data, command instructions, camera feeds, and mapping data. While performing these functions, communication with different types of data is quite common. For example, data obtained from a camera is of a different type compared to data acquired from a distance sensor. Is the encryption and decryption speed, as well as memory usage, the same for data of different types but the same size, when using the ChaCha20-Poly1305-ECC algorithm? To answer this question, files with sizes approximately 50 MB, 100 MB, and 200 MB, in PDF, MP4, and TXT formats, were tested using the ChaCha20-Poly1305-ECC algorithm. Data of the same size but different types were compared in terms of memory usage and processing time. The built-in "time" library of Python was used for measuring the algorithm's processing time, while the "memory_profiler" library was utilized for memory usage. The obtained results are presented in Table 3.

Table 3. Performance Metrics by File Type and Data Size

Data Size (MB)	File Type	Memory Usage (MB)	Processing Time (Seconds)
50 MB	PDF	4.03125	0.62131
	MP4	4.109375	0.60822
	TXT	4.03515625	0.5942
100 MB	PDF	3.98046875	1.17811
	MP4	4.0234375	0.97805
	TXT	4.0078125	1.02113
200 MB	PDF	3.90625	1.50388
	MP4	3.953125	1.5814
	TXT	4.00390625	1.57084

Table 3 provides a detailed analysis of performance metrics across various file types and data sizes. This analysis primarily focuses on two key parameters: memory usage (in MB) and processing time (in seconds). These parameters have been examined for three different file types (PDF, MP4, TXT) and different data sizes (50 MB, 100 MB, 200MB.).

From an initial observation, it appears that the memory usage remains relatively stable across different file types and sizes, indicating a consistent memory footprint regardless of file type or data size. However, there are slight variations in memory usage, which could be attributed to the inherent differences in the data structure and compression algorithms used by each file type. Processing time, another critical metric, shows a variation that correlates with the data size and file type. It suggests that as the data size increases, the processing time also increases, which is expected. The processing time is a crucial factor in scenarios where time efficiency is a priority, such as real-time data processing or large-scale data analysis. This table provides valuable insights, particularly for applications where memory efficiency and processing speed are crucial. It allows one to anticipate the system resources required and processing time for handling different file types and sizes, which is vital for optimizing performance in data-intensive applications. Analyzing such data helps in making informed decisions about resource allocation, system design, and choice of file types based on the specific requirements of an application. These results are depicted in the graph found in Figure 4.

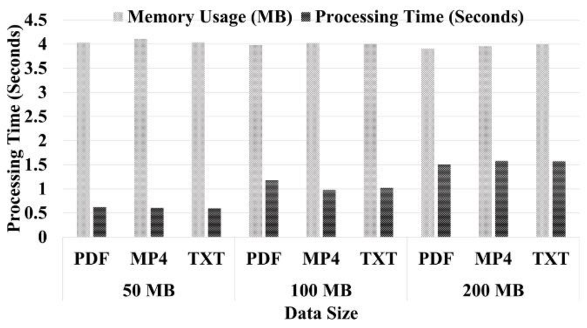

Figure 4. Comparative Analysis of Memory Usage and Processing Time Across Different File Types and Data Sizes

The graph presents a dual-axis bar chart comparing memory usage (in megabytes) and processing time (in seconds) for three different file types—PDF, MP4, and TXT—across three data sizes: 50 MB, 100 MB, and 200 MB. The left vertical axis corresponds to the processing time, while the right vertical axis presumably represents memory usage, although the right axis is not labeled in the image provided. Observations indicate that for all three file types, as the data size increases, the processing time escalates as well. This trend is consistent with computational theory, where larger data volumes typically require longer processing durations.

However, memory usage does not exhibit a proportional increase with data size; instead, it remains relatively constant or varies slightly. This could suggest an efficient memory management system where the memory footprint does not significantly increase with larger data sizes. The PDF and MP4 files display similar behaviors in terms of processing time, with only minor differences, which might be attributed to the complexity of data contained within each file type. TXT files, by contrast, have a markedly lower processing time across all data sizes, reflecting their simpler structure and the less intensive processing required. In conclusion, the data conveyed in the graph supports the premise that while processing time is susceptible to changes in data volume, memory usage does not necessarily correlate directly with data size. This implies an optimized use of memory resources, which is particularly advantageous for systems with limited memory capacity. Additionally, the distinct difference in processing times between the file types underscores the importance of considering file structure and complexity when designing and optimizing data processing systems.

5. Conclusion

To improve the clarity and fluency of the paper, attention has been paid to clarify the distinction between encryption technologies and applications of industrial robot arms. By detailing the main features of encryption technologies and their integration into industrial applications, a comprehensive understanding of the transformation of theoretical concepts into practical applications is presented. This transition through sub-headings is intended to strengthen the overall coherence of the article and transform it into a more user-friendly and application-oriented resource.

The article uses material such as diagrams and example scenarios to clarify concepts and relate technical information to practical applications. The quality of images and tables has been improved and explanatory comments have been added so that readers can clearly see the topics. Care has been taken with grammar and punctuation, and consistency in technical terms has been ensured. Sentences have been shortened to improve clarity and a glossary of technical jargon has been included. Each term and concept is explained the first time it is used, and transitions between topics are made clear. These steps have improved the overall comprehensibility of the article. The detailed research systematically evaluated the effectiveness of a novel hybrid encryption method designed to enhance data security for industrial robot arms.

The method synergistically leverages the strengths of Elliptic Curve Cryptography (ECC) for asymmetric encryption capabilities, ChaCha20 for fast symmetric encryption, and Poly1305 for reliable message authentication to establish a robust security posture against data attacks during both wired and wireless communications. In this work, the importance of cybersecurity approaches, assets, and applications for protection is discussed [47, 48].

The encryption protocol developed for industrial robot arms is supported by detailed analysis and testing to offer robustness against potential security threats. Simulations and penetration tests validate the effectiveness of the protocol and its solutions to industry challenges. By highlighting the security advantages of the algorithm and its effectiveness in application scenarios, this study illuminates how to integrate encryption methods in industrial applications.

The study also analyzes how encryption methods scale on industrial robot arms by testing performance on data types such as .bin files, providing valuable results, especially with file sizes ranging from 100 MB to 1 GB. This information demonstrates the practical applicability of encryption methodologies by comparing between large and small datasets. Related works in the literature [49] highlight the performance and flexibility advantages of algorithms such as AES and BlowFish and provide a reference for determining the security potential of encryption methods. The selected algorithms fit the specific requirements of the project, with unique advantages such as ECC's high-security small keys, ChaCha20's balance of performance and security, and Poly1305's fast message authentication. These choices are based on an extensive analysis of the existing literature, including comparisons of these algorithms within and against each other, with critical factors such as data integrity and security.Empirical analysis shows that the ECC componentsignificantly enhances the level of security, while ChaCha20 and Poly1305 contribute to high-speed data processing, providing not only secure communication channels but also minimal latency in data transfer operations.These findings underline the potential of the ECC-ChaCha20-

Poly1305 combination as a superior encryption strategy, especially for industrial applications where speed and security are crucial.

Furthermore, this paper evaluates the effects of encryption methods on performance metrics such as real-time processing time and memory utilization of industrial robotic arms and examines the effects of ECC on system performance compared to RSA [50].

This investigation contributes to the broader cryptographic community by providing a comparative analysis with existing methods, highlighting improved performance metrics in terms of both encryption speed and resource efficiency. In order to understand the place of the encryption method proposed in the paper in the current technological landscape, in addition to comparisons between common methods such as ECC and RSA, a wide range of symmetric encryption algorithms are also examined. In this context, a detailed analysis of the advantages and application scenarios of algorithms such as Advanced Encryption Standard (AES), Blowfish, Twofish, RC6, Serpent, Camellia, and many more is presented [51].

The insights gained from this research can serve as a cornerstone for future innovations in the field of encryption, especially in industrial automation and robotics, where data security is becoming increasingly critical. In summary, this research not only demonstrates that the hybrid method is a viable security solution, but also paves the way for its adoption in complex industrial ecosystems by providing a detailed academic and practical framework for implementation and evaluation.

Article Information Form

Funding
The author (s) has no received any financial support for the research, authorship or publication of this study.

Authors' Contribution
Conceptualization, M.Ö. and H.C.B.; methodology, H.C.B.; software, M.E.E.; validation, M.Ö., H.C.B. and M.E.E.; investigation, M.E.E.; data curation, M.Ö.; writing—original draft preparation, M.E.E.; writing—review and editing, M.Ö. and H.C.B.; visualization, M.Ö. All authors have read and agreed to the published version of the manuscript.

The Declaration of Conflict of Interest/ Common Interest
No conflict of interest or common interest has been declared by the authors.

The Declaration of Ethics Committee Approval
This study does not require ethics committee permission or any special permission.

The Declaration of Research and Publication Ethics
The authors of the paper declare that they comply with the scientific, ethical and quotation rules of SAUJS in all processes of the paper and that they do not make any falsification on the data collected. In addition, they declare that Sakarya University Journal of Science and its editorial board have no responsibility for any ethical violations that may be encountered, and that this study has not been evaluated in any academic publication environment other than Sakarya University Journal of Science.

Copyright Statement
Authors own the copyright of their work published in the journal and their work is published under the CC BY-NC 4.0 license.

References

[1] M. C. Cengiz, B. Kaftanoğlu, "Endüstriyel Bir Robot İçin İnsan Makina Arayüz Programının Geliştirilmesi," Makina Tasarım ve İmalat Dergisi, cilt. 6, no. 2, ss. 99-107, 2004.

[2] M. E. Erbil, A. A. Süzen ve H. C. Bayrakçı, "Otonom mobil robotların güvenli veri iletimi için hibrit şifreleme yaklaşımı," UTBD, cilt 15, no. 2, s. 64-72, 2023.

[3] B. Dieber, B. Breiling, S. Taurer, S. Kacianka, S. Rass ve P. Schartner, "Security for the Robot Operating System," Robotics and Autonomous Systems, cilt 98, s. 192-203, 2017.

[4] M. P. Groover, “Automation, Production Systems, and Computer-Integrated Manufacturing”, Prentice Hall, 2008.

[5] J. J. Craig, “Introduction to Robotics: Mechanics and Control”, Pearson/Prentice Hall, 2005.

[6] G. J. Olling, R. E. Merritt, Eds., “The Factory Automation Handbook: History”, Trends, and Forecasts, CRC Press, 1993.

[7] J. N. Pires, J. R. Azinheira, Eds., “Progress in Robotics”, Springer, 2008.

[8] B. Siciliano, O. Khatib, Eds., “Springer Handbook of Robotics”, Springer, 2008.

[9] S. Kalpakjian, S. R. Schmid, “Manufacturing Engineering and Technology”, Pearson Prentice Hall, 2006.

[10] J. Elkington, “Cannibals with Forks: The Triple Bottom Line of 21st Century Business”, Capstone, 1997.

[11] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone, “Handbook of Applied Cryptography”, CRC Press, 1996.

[12] W. Ding, L. Yan, R. H. Deng, "A survey on hybrid encryption schemes in vehicular ad-hoc networks," IEEE Transactions on Intelligent Transportation Systems, 18, no. 3, ss. 655-667, 2017.

[13] B. Libert, M. Yung, "Efficient identity-based encryption without random oracles and its application to asymmetric searchable encryption," Annual International Cryptology Conference, ss. 600-619, Springer, Berlin, Heidelberg, 2009.

[14] H. Wang, B. Qin, Q. Wu, J. Domingo-Ferrer, L. Zhang, "Privacy-preserving hybrid cloud with a homomorphic encryption," IEEE Transactions on Cloud Computing, 9, no. 3, ss. 1014-1026, 2019.

[15] X. Wu, G. Revadigar, "A secure and efficient hybrid encryption scheme for securing RFID tag communications," Journal of Network and Computer Applications, cilt. 42, ss. 109-116, 2014.

[16] A. S. Tanenbaum, M. Van Steen, “Distributed Systems: Principles and Paradigms”, Prentice-Hall, 2002.

[17] N. Koblitz, “A Course in Number Theory and Cryptography”, 2. edition, Springer-Verlag, 1994.

[18] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone, “Handbook of Applied Cryptography”, CRC Press, 1996.

[19] R. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM, 21, no. 2, ss. 120-126, 1977.

[20] D. R. Stinson, “Cryptography: Theory and Practice”, 3, CRC Press, 2005.

[21] A. J. Menezes, P. C. Van Oorschot, S. A. Vanstone, “Handbook of Applied Cryptography”, CRC Press, 2001.

[22] D. R. Stinson, “Cryptography: Theory and Practice”, 3, CRC Press, 2005.

[23] W. Stallings, “Cryptography and Network Security: Principles and Practice”, Pearson Education, 2016.

[24] J. Katz, Y. Lindell, “Introduction to Modern Cryptography”, 3, CRC Press, 2020.

[25] B. Schneier, “Applied Cryptography: Protocols, Algorithms, and Source Code in C”, John Wiley & Sons, 1996.

[26] D. J. Bernstein, "ChaCha, a variant of Salsa20," 2008.

[27] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)," FIPS PUB 197, 2001.

[28] S. Indla, A. Donald, A. T. Aditya, T. A. Srinivas, G. Thippanna, "Locking Down Big Data: A Comprehensive Survey of Data Encryption Methods," International Journal of Advanced Research in Science, Communication and Technology, 10, 48175, 2023.

[29] D. J. Bernstein, "ChaCha, a variant of Salsa20," 2008.

[30] D. J. Bernstein, "ChaCha, a variant of Salsa20," Workshop Record of SASC, sayı 4, 2008.

[31] A. J. Menezes, P. C. Van Oorschot, S. A. Vanstone, “Handbook of Applied Cryptography”, CRC Press, 1996.

[32] M. Bellare, R. Canetti ve H. Krawczyk, "Keying Hash Functions for Message Authentication," Advances in Cryptology, CRYPTO '96.

[33] D. J. Bernstein, "The Poly1305-AES message-authentication code," Fast Software Encryption, ss. 32-49, 2005.

[34] D. J. Bernstein, "The Poly1305-AES message-authentication code," 2005.

[35] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)," FIPS PUB 197, 2001.

[36] D. J. Bernstein, "The Poly1305-AES message-authentication code," 2005.

[37] D. J. Bernstein, "ChaCha, a variant of Salsa20," 2008.

[38] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)," FIPS PUB 197, 2001.

[39] D. J. Bernstein, "The Poly1305-AES message-authentication code," 2005.

[40] D. J. Bernstein, "ChaCha, a variant of Salsa20," 2008.

[41] A. J. Menezes, P. C. Van Oorschot, S. A. Vanstone, “Handbook of Applied Cryptography”, CRC Press, 2001.

[42] D. R. Stinson, “Cryptography: Theory and Practice”, 3, CRC Press, 2005.

[43] S. Padhiar, "A Comparative Study on Symmetric and Asymmetric Key Encryption Techniques," in 2021.

[44] S. Asjad, “The RSA Algorithm,” 2019.

[45] M. Gobi, S. R. Sridevi, R. Rahini, "A Comparative Study on the Performance and the Security of RSA and ECC Algorithm," 2020.

[46] A. Boicea, C.-O. Truică, F. Rădulescu, D.-C. Popeangă, I.-M. Radulescu ve C. Costea, "Cryptographic Algorithms Benchmarking: A Case Study," 2019.

[47] M. Abutaha, B. Atawneh, L. Hammouri ve diğerleri, "Secure lightweight cryptosystem for IoT and pervasive computing," Sci Rep, cilt 12, no. 19649, 2022.

[48] Ayman Alissa, Duarte Bacelar Begonha, Jim Boehm, Duarte Braga, Joana Candina, Hugo Espírito Santo, Wolf Richter ve Benjamim Vieira, "How to enhance the cybersecurity of operational technology environments," McKinsey & Company, 23 Mart 2023.

[49] M. Alenezi, H. Alabdulrazzaq ve N. Mohammad, "Symmetric Encryption Algorithms: Review and Evaluation study," International Journal of Communication Networks and Information Security, cilt 12, s. 256, 2020.

[50] Ayman Alissa, Duarte Bacelar Begonha, Jim Boehm, Duarte Braga, Joana Candina, Hugo Espírito Santo, Wolf Richter ve Benjamim Vieira, "How to enhance the cybersecurity of operational technology environments," McKinsey & Company, 23 Mart 2023.

[51] P. Patil, P. Narayankar, D.G. Narayan ve M. S. Meena, "A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish," Procedia Computer Science, cilt 78, s. 617-624, 2016.