Decision Making: Applications in Management and Engineering

ISSN: 2560-6018, eISSN: 2620-0104

AES ON MODBUS ASCII TO PREVENT AGAINST ACTIVE ATTACKS

Dr Sameer S Nagtilak
Department of CSE, DYPSEM,Kolhapur, India, nagtilak.sameer@kitcoek.in

Mr Tanaji B Patil
Department of CSE, DYPSEM,Kolhapur, India, tbpatil.sem@dypgroup.edu.in

Mrs Archana S Shinde- Sawant
Department of CSE, DYPSEM,Kolhapur, India, assawant.sem@dypgroup.edu.in

Dr G V Otari
Department of CSE, DYPSEM,Kolhapur, India, gvotari.sem@dypgroup.edu.in

Abstract
The integration of industrial control systems with standard network infrastructure has amplified their vulnerability to cyberattacks, thereby underscoring the urgent need for robust security mechanisms. Modbus, a widely adopted communication protocol in industrial automation, is particularly susceptible to various cyber threats due to its inherent lack of security features. This research delves into the application of Advanced Encryption Standard on Modbus ASCII to fortify industrial control systems against evolving cyber threats, especially active attacks. Active attacks targeting Modbus networks can have dire consequences, ranging from financial losses for control system operators to safety concerns for the public who rely on these services. In light of the escalating number and complexity of cybersecurity incidents and threats, it has become crucial to analyze the cybersecurity of Modbus/TCP and to develop cryptography-based solutions. This study aims to investigate the effectiveness of AES encryption in mitigating these risks, focusing on its implementation within the Modbus ASCII framework. Man-in-the-middle attacks, intercepted information replay, and abnormal byte transferring are just a few of the many potential attacks that SCADA communication may face.

I. Introduction

The integration of Advanced Encryption Standard into Modbus ASCII communication protocols addresses critical security vulnerabilities inherent in traditional industrial control systems. Modbus, initially developed in 1979, is a widely adopted communication protocol within industrial environments [1], [2]. However, its original design lacked robust security features, rendering it susceptible to various cyber threats [3]. The simplicity and widespread deployment of Modbus in both legacy and contemporary systems have spurred considerable research into methods for enhancing its security capabilities [4]. By implementing AES encryption, particularly within the Modbus ASCII framework, a substantial layer of defense

can be added against eavesdropping, tampering, and unauthorized access, thereby safeguarding critical infrastructure and sensitive data [5]. The transition from proprietary industrial control systems to standard protocols and general-purpose products has widened the attack surface, making connected control systems increasingly vulnerable to cyber attacks via network connections [6]. The inherent vulnerabilities of Modbus, combined with its ubiquitous presence in critical infrastructure, highlight the pressing need for robust security measures such as AES encryption to mitigate potential risks and ensure the integrity and reliability of industrial operations. The seamless integration of information technologies and operational technologies enhances the management of industrial environments. However, the incorporation of new technologies, which increases the number of internet connections, new communication protocols, and interfaces that run on open-source software, introduces new threats and challenges in addition to existing vulnerabilities in these classical legacy-based heterogeneous hardware and software systems.

II. AES Encryption

Advanced Encryption Standard is a symmetric block cipher extensively employed to secure sensitive data across numerous applications and industries [7]. AES operates on fixed-size blocks of data and utilizes cryptographic keys of varying lengths, such as 128, 192, or 256 bits, to encrypt and decrypt information. The algorithm's strength lies in its resistance to known cryptanalytic attacks, owing to its intricate mathematical structure and key agility [8]. AES is used in many different applications and sectors because of its speed, security, and relatively low memory needs [9]. By transforming plaintext into ciphertext and vice versa, the encryption process involves a series of substitution, permutation, and mixing operations. The key length determines the number of rounds in the encryption process, with longer keys providing higher levels of security. AES has become the standard for safeguarding digital data because of its effectiveness, adaptability, and demonstrated security. Confidentiality, integrity, and availability are the cornerstones of data and information security. Because each of these pillars is essential, security solutions should address all of them. The use of encryption guarantees data confidentiality, ensuring that only authorized parties can access sensitive information.

One strategy is to create a unique encryption algorithm based on the data's properties and application scenarios, as AES is not always applicable for encryption in every situation [10]. In certain applications, symmetric key algorithms such as AES and DES require the sender and receiver to generate and exchange a shared key. The possibility of an intruder copying or stealing the secret key when it is sent over an insecure channel is a major issue with symmetric key algorithms [7]. The security afforded by a 1024-bit key utilizing asymmetric RSA is regarded to be roughly equivalent to that of an 80-bit key from a symmetric algorithm [11].

2.1 Modbus System attacks

Cyberattacks against industrial control systems that use the Modbus application layer network protocol, can result in financial loss for control system operators and economic and safety issues for the citizens who use these services [2]. The increasing number of these cybersecurity incidents and threats necessitates an analysis of the cybersecurity of Modbus/TCP and the creation of cryptography-based solutions. Cyberattacks, such as side-channel attacks, may be launched because embedded devices are frequently inexpensive, low-power devices with

limited memory and processing capability [12]. To guarantee asset security in industrial automation, hazards must be identified, risks must be assessed, and suitable safeguards must be implemented [8]. Since Modbus is frequently used to retrieve data from numerous devices, it is essential to secure Modbus communications. Attackers can target industrial control systems using a variety of techniques, such as malware injection, denial-of-service attacks, and man-in-the-middle attacks. These attacks can disrupt operations, compromise data integrity, and even cause physical damage to equipment. Securing Modbus communications is essential to preventing unauthorized access, maintaining data integrity, and assuring the availability of industrial processes[13].

2.2 Comparative study of AES with other Cryptography tools

AES is favored in high-security situations, while Blowfish is favored in high-performance situations [8]. AES is frequently used to safeguard data because it offers a high degree of security. AES is appropriate for applications that demand a high level of security since it has been approved by the US government for protecting sensitive data. In contrast, Blowfish is renowned for its speed and efficiency, making it an excellent choice for applications where performance is crucial. Blowfish is well suited to applications with sluggish CPUs because of its quick encryption and decryption times. The Advanced Encryption Standard is the most used cryptographic algorithm because of its great level of security [14].

The Advanced Encryption Standard has a set block size, supports a variety of key sizes, and is appropriate for a variety of applications. The Blowfish encryption technique is better suited to the security of wireless network applications [8]. AES-based algorithms designed specifically for constrained devices have demonstrated significantly better performance than those that are not optimized, utilizing approximately 0.16 times the energy and time required for encryption and decryption [15]. AES and Blowfish are versatile algorithms that may be used in a variety of applications and are appropriate for usage on embedded devices.

2.3 Enhancing Modbus ASCII Security with AES

The Advanced Encryption Standard is a modern block cipher that offers a high level of security and efficiency. It is implemented to improve the security of Modbus ASCII communication. AES offers a safe means to protect Modbus data because of its resistance to known cryptanalytic attacks and adjustable key sizes [16]. When incorporated into Modbus ASCII systems, AES encryption secures sensitive data against eavesdropping, tampering, and unauthorized access [16]. The key length affects the degree of security, with longer keys providing greater protection against brute-force attacks [17]. AES may be included into Modbus ASCII systems by adding cryptographic libraries to Modbus devices and adjusting the communication protocol to allow encryption and decryption. AES is a good fit for Modbus ASCII systems since it offers a strong and flexible encryption option that improves data security and integrity in industrial applications [18]. It is essential to emphasize that the security of any cryptographic system is contingent on the strength and secure management of the encryption keys used. To guarantee long-term security against developing threats, strong key generation techniques, secure key exchange methods, and routine key rotation procedures should all be used.

Because of the sensitive nature of the data conveyed by industrial control systems, security is crucial [19], [20]. The Advanced Encryption Standard is capable of using cryptographic keys of varying bit lengths—128, 192, and 256—to encrypt and decrypt data in 128-bit blocks [16]. AES is a symmetric block cypher that is widely used in distributed systems and offers a high level of security. Distributed systems have grown rapidly over the last two decades [21]. Modbus security can be improved by integrating AES encryption. AES, a block cypher with a relatively rapid encryption method and a robust algorithm against hacking, has a number of advantages when it comes to encrypting text [22]. It is important to analyze the Modbus protocol in depth and use the PDU field values to create rules that strengthen the firewall's security and dependability. When implemented in Modbus systems, AES encryption secures data against unauthorized access, guaranteeing confidentiality and integrity.

Modbus, being a widely used protocol in industrial automation, traditionally lacks inherent security features, making it vulnerable to various cyber threats [23]. Implementing AES encryption on top of the Modbus ASCII protocol introduces a robust security layer, effectively mitigating risks such as eavesdropping and data tampering. Due to the growing risk of attacks on industrial infrastructure, it's important to ensure the safety of Modbus networks. One strategy is to combine AES encryption with the Modbus ASCII protocol to provide a strong security solution that protects sensitive data.

III. Modbus ASCII Protocol

Modbus ASCII is a serial communication protocol widely used in industrial automation for communication between devices. The Modbus protocol has two versions: Modbus RTU and Modbus ASCII. The Modbus ASCII protocol is a communication protocol used in industrial automation systems. It uses ASCII characters to represent data, making it more human-readable but less efficient than Modbus RTU, which uses binary representation [4]. Modbus enables communication between various devices, including programmable logic controllers, human-machine interfaces, and sensors [24]. Modbus ASCII is a character-based protocol that uses ASCII characters to represent data. Modbus ASCII is more verbose and less efficient than Modbus RTU because each byte of data is represented by two ASCII characters. The Modbus ASCII protocol employs a master/slave architecture, wherein a single master device initiates communication with one or more slave devices. The master sends commands to the slaves, and the slaves respond with data. The simplicity and ease of implementation of Modbus ASCII have contributed to its widespread adoption in industrial settings.

START	ADDRESS	FUNCTION	DATA	LRC CHECK	END
1 CHAR :	2 CHARS	2 CHARS	n CHARS	2 CHARS	2 CHARS CRLF

Fig 1. Modbus ASCII Header

Modbus ASCII is preferred in scenarios where human readability and ease of debugging are important. The usage of ASCII characters makes it simpler to understand and troubleshoot communication issues, particularly in systems where data needs to be readily interpreted by

human operators. Modbus ASCII's text-based format, while facilitating debugging, introduces overhead due to its lower data density compared to binary protocols like Modbus RTU. The choice between Modbus ASCII and Modbus RTU depends on the specific requirements of the application. In resource-constrained environments or applications where bandwidth is limited, Modbus RTU's compact binary format is generally preferred [25].

IV. Integrating AES and Modbus ASCII

The integration of AES encryption with Modbus ASCII involves several key steps to ensure secure communication without compromising the protocol's functionality. The implementation process begins with establishing a secure key exchange mechanism between the master and slave devices. Implementing AES encryption into Modbus ASCII involves several important steps to ensure secure communication while maintaining protocol functionality. AES encryption keys must be exchanged securely between Modbus devices.

The first step in integrating AES with Modbus ASCII is to establish a secure key exchange mechanism. Diffie-Hellman or pre-shared keys can be used to securely exchange encryption keys between the master and slave devices. One method is to utilize the Diffie-Hellman key exchange algorithm, which enables the secure exchange of cryptographic keys over a public channel. Another approach involves pre-shared keys, where the encryption keys are manually configured on both the master and slave devices.

After a safe key exchange, the Modbus data is encrypted utilizing the AES algorithm before transmission. Modbus Application Protocol defines the structure of messages in a client/server mode at the application layer [3]. Before transmission, the Modbus data is encrypted using the negotiated AES key. The encrypted data is then encapsulated within the Modbus ASCII frame, adhering to the protocol's syntax and structure. The Modbus data is encrypted using the AES algorithm and the negotiated key. The encrypted data is then included in the Modbus ASCII frame, following the protocol's syntax. Upon receiving the encrypted Modbus ASCII frame, the receiving device extracts the encrypted data and decrypts it using the AES algorithm and the shared key. The receiving device decrypts the data using the AES algorithm and the shared key.

Integrating AES encryption with Modbus ASCII provides a robust security layer for industrial communication networks. When correctly implemented, this technique protects data against unauthorized access and tampering, preserving the integrity and confidentiality of Modbus communications [26].

Integrating AES encryption into Modbus ASCII demands careful consideration of performance overhead. The encryption and decryption processes introduce computational overhead, which can impact the real-time performance of Modbus communication, so it's important to consider the performance effect of encryption and decryption.

The selection of AES key size also plays a crucial role in determining the security strength and performance of the encryption scheme. Larger key sizes, such as AES-256, offer higher security but require more computational resources compared to smaller key sizes like AES-128.

V. Implementation Considerations

When implementing AES encryption with Modbus ASCII, several key considerations must be addressed to ensure optimal performance, security, and compatibility. The real-time

performance of Modbus communication can be impacted by the computational overhead associated with encryption and decryption. When implementing AES encryption with Modbus ASCII, key management is an important aspect to consider. Secure storage and distribution of AES keys are essential to prevent unauthorized access and guarantee the confidentiality of Modbus communication. To secure Modbus communication, it is essential to carefully manage AES keys, including secure storage and distribution to prevent unauthorized access and maintain confidentiality[27].

Selecting an appropriate AES key size is crucial for balancing security strength and performance overhead. Although bigger key sizes provide greater security, they also need more computing resources, which might affect Modbus communication's real-time performance. Consider the trade-off between security strength and performance when choosing an AES key size. AES-128, AES-192, and AES-256 are typical key sizes, each with distinct security and performance properties.

Selecting an appropriate AES key size is crucial for balancing security strength and performance overhead. Typical key sizes include AES-128, AES-192, and AES-256, each offering different security and performance characteristics [28]. It's crucial to assess the Modbus network's security requirements and performance limits before deciding on a key size. To avoid compatibility difficulties and guarantee interoperability between Modbus devices, adherence to industry standards and protocols is crucial.

Adherence to industry standards and protocols is crucial to avoid compatibility issues and ensure interoperability between Modbus devices. To guarantee that encrypted Modbus ASCII communication is compatible with existing Modbus infrastructure, use established standards and protocols like the Modbus Application Protocol Specification.

Selecting the right AES operating mode, such as Cipher Block Chaining or Counter, is essential to maximizing security and performance in Modbus ASCII applications. Each mode has distinct security properties and performance features, so it's important to pick one that fits the network's unique needs.

Selecting the right AES operating mode, such as CBC or CTR, is essential for maximizing security and performance in Modbus ASCII applications. Each mode has distinct security properties and performance features, so it's important to pick one that fits the network's unique needs. By addressing these implementation considerations, developers can effectively integrate AES encryption with Modbus ASCII, providing a robust security layer for industrial communication networks [5].

VI. Error checking field in RTU and ASCII

Modbus RTU and Modbus ASCII are two distinct serial communication protocols used for transmitting data between devices in industrial automation systems. Cyclic Redundancy Check is used by Modbus RTU to detect errors, whereas Longitudinal Redundancy Check is used by Modbus ASCII. The integrity of the transmitted data is ensured by these error-checking mechanisms, which spot mistakes made during communication. The checksum, a crucial component of Modbus RTU and ASCII protocols, guarantees the dependability of data transmission across industrial networks. Cyclic Redundancy Check is used by Modbus RTU to detect errors, whereas Longitudinal Redundancy Check is used by Modbus ASCII.

An essential component of Modbus RTU and ASCII protocols is the checksum, which guarantees the dependability of data transmission across industrial networks. When choosing between Modbus RTU and Modbus ASCII, one must consider the trade-offs between transmission efficiency, error detection capabilities, and compatibility with existing systems. Understanding the nuances of error checking in each protocol is crucial for ensuring reliable communication in industrial automation applications[29].

Industrial automation applications require an understanding of the subtleties of error detection in each protocol in order to ensure dependable communication. The method of mistake detection is one of the main distinctions between Modbus RTU and Modbus ASCII. Modbus RTU employs cyclic redundancy check, or CRC, a strong error-detection method that offers a high degree of assurance for data integrity. A mathematical algorithm is used to produce a fixed-size checksum value based on the data being transmitted. The receiver computes its checksum value using the same algorithm and compares it to the checksum value that was sent. If the computed and received checksum values match, the data is regarded as error-free; otherwise, an error has happened during transmission.

The receiver computes its checksum value using the same algorithm and compares it to the checksum value that was sent. If the computed and received checksum values match, the data is regarded as error-free; otherwise, an error has happened during transmission. For dependable data transmission, CRC is particularly well-suited for noisy industrial environments where electrical interference or signal degradation may occur [30].

Longitudinal Redundancy Check, or LRC, is used by Modbus ASCII for error checking [31]. LRC is a simpler error-detection method than CRC. It entails adding up all the bytes in the message and then subtracting the result from 256. The resultant value is appended to the message as the LRC checksum. The receiver performs the same calculation on the received data and compares the computed LRC value to the received LRC value. The data is regarded as valid if the two values match; otherwise, an error has happened during transmission. LRC offers a fundamental degree of error detection, although it is less resilient to errors than CRC.

The reliability of industrial communication networks depends on the error-checking mechanisms used by Modbus RTU and Modbus ASCII. Modbus RTU's CRC provides superior error detection capabilities and is appropriate for applications requiring high data integrity, even though Modbus ASCII's LRC offers a simpler error-detection method.

VII. Modbus ASCII execution code

Modbus ASCII uses ASCII characters to represent data, making it human-readable but less efficient than Modbus RTU, which uses binary representation. When implementing Modbus ASCII, it's crucial to consider the execution code, which dictates the specific actions performed by the Modbus slave device. Understanding the different execution codes is essential for programming Modbus master devices to interact effectively with slave devices. Error detection is crucial for ensuring the reliability of data transmitted over Modbus networks [32], [33]. The execution code in Modbus ASCII determines the operation to be performed on the slave device, such as reading input registers, writing to holding registers, or reading diagnostic information. Forward error correction techniques are crucial in wireless communication to identify and fix bit errors that happen during transmission [34]. Modern wireless communication systems use methods like low-density parity check codes and convolutional codes to handle the problem of

erroneous information transmission over wireless channels [34]. When creating Modbus ASCII systems, developers must carefully choose the execution codes that match the intended operations and data kinds.

The function code instructs the server on the precise action to take [35]. The data field comprises the information the server needs to carry out the action the function code specifies. The error-checking field enables the server to verify the accuracy of the message it received. In Modbus, function codes are used to initiate actions on the slave device.

The Modbus protocol defines a set of standard function codes for reading and writing data to different types of registers and coils. For instance, function code 0x03 is used to read holding registers, while function code 0x06 is used to write a single holding register. The software on the receiving end is designed to detect the start byte before storing subsequent bytes. These bytes are then analyzed to differentiate between clean data and errors [36]. By adhering to these standard function codes, Modbus devices from different manufacturers can interoperate seamlessly [35].

Modbus function codes specify the operations that a master device can request from a slave device, such as reading input registers, writing to holding registers, or reading coil status. In addition to the standard function codes, Modbus also allows for the use of custom function codes to implement vendor-specific features or functionalities. When defining custom function codes, it's essential to follow the Modbus specification guidelines to avoid conflicts with existing standard function codes . To guarantee interoperability with already-existing Modbus systems, it is imperative to adhere to the Modbus standard when implementing custom function codes. [37]

VIII. Conclusion

AES encryption provides a strong defense against unauthorized access and tampering, ensuring the security and integrity of Modbus communication in industrial applications. By encrypting Modbus data with AES, organizations can prevent eavesdropping, data manipulation, and other cyberattacks that could compromise critical industrial processes.

AES encryption provides a strong defense against unauthorized access and tampering, ensuring the security and integrity of Modbus communication in industrial applications.

AES encryption provides a strong defense against unauthorized access and tampering, ensuring the security and integrity of Modbus communication in industrial applications. By adhering to industry standards, carefully managing AES keys, and selecting appropriate key sizes and operating modes, organizations can optimize the performance, security, and compatibility of encrypted Modbus ASCII communication.The Advanced Encryption Standard is implemented in Electronic Code Book mode and Ciphered Message Authentication Code mode. When correctly implemented, AES encryption strengthens the security posture of industrial control systems and protects critical infrastructure from cyber threats.

Although asymmetric encryption may offer advantages in certain scenarios, it might not be appropriate for broadcasting communication because of the increased cryptographic key requirements between participating nodes and the necessity for a certificate authority . When correctly implemented, AES encryption strengthens the security posture of industrial control systems and protects critical infrastructure from cyber threats, especially when considering the limitations of hardware like memory and battery life . Modbus is a widely used communication

protocol in industrial settings, connecting devices within a network . Industrial control systems are tempting targets for cyberattacks because of their vital role in essential infrastructure .
As industrial automation and control systems become more interconnected, they face a growing number of cyberattacks . As a result, security solutions must be implemented to protect the communication networks that these systems rely on . Integrating information and operational technologies enhances industrial environment management; however, it introduces new threats and challenges alongside existing vulnerabilities in legacy systems . Managing security issues and software updates is essential for maintaining protection and resilience against cyberattacks throughout the lifespan of industrial systems .
Integrating security measures into industrial control systems can potentially impact safety, highlighting the complex nature of industrial control system security and the challenges of measuring its features . As the number of internet connections grows, along with the use of new communication protocols and open-source software interfaces, industrial control systems face increasing security incidents . Implementing strong security measures and reducing the attack surface are essential for protecting industrial automation systems from cyberattacks . Cyberattacks on building control systems can cause failures or inoperability, emphasizing the growing implications of cybersecurity in construction and building systems due to the increasing use of information and communication technology . By using AES encryption, organizations can improve the security of their Modbus networks, reduce the risk of cyberattacks, and protect critical industrial processes.

References

[1] C. Parian, T. Guldimann, and S. Bhatia, “Fooling the Master: Exploiting Weaknesses in the Modbus Protocol,” Procedia Computer Science, vol. 171, p. 2453, Jan. 2020, doi: 10.1016/j.procs.2020.04.265.
[2] P. Huitsing, R. Chandia, M. Papa, and S. Shenoi, “Attack taxonomies for the Modbus protocols,” International Journal of Critical Infrastructure Protection, vol. 1, p. 37, Nov. 2008, doi: 10.1016/j.ijcip.2008.08.003.
[3] W. Gao and T. H. Morris, “On Cyber Attacks and Signature Based Intrusion Detection for Modbus Based Industrial Control Systems,” The journal of digital forensics, security and law, Jan. 2014, doi: 10.15394/jdfsl.2014.1162.
[4] C. Urrea, C. Morales, and J. Kern, “Implementation of error detection and correction in the Modbus-RTU serial protocol,” International Journal of Critical Infrastructure Protection, vol. 15, p. 27, Jul. 2016, doi: 10.1016/j.ijcip.2016.07.001.
[5] A. Shahzad et al., “Real Time MODBUS Transmissions and Cryptography Security Designs and Enhancements of Protocol Sensitive Information,” Symmetry, vol. 7, no. 3, p. 1176, Jul. 2015, doi: 10.3390/sym7031176.
[6] Y.-H. Chun, “Analysis of Cyber Threats in the Connection Section of the Control System and Countermeasures Required,” Türk bilgisayar ve matematik eğitimi dergisi, vol. 12, no. 6, p. 412, Apr. 2021, doi: 10.17762/turcomat.v12i6.1831.
[7] C. Biswas, Md. M. Haque, and U. D. Gupta, “A Modified Key Sifting Scheme With Artificial Neural Network Based Key Reconciliation Analysis in Quantum Cryptography,” IEEE Access, vol. 10, p. 72743, Jan. 2022, doi: 10.1109/access.2022.3188798.

[8] A. E. Adeniyi, S. Misra, E. Daniel, and B. Anthony, “Computational Complexity of Modified Blowfish Cryptographic Algorithm on Video Data,” Algorithms, vol. 15, no. 10, p. 373, Oct. 2022, doi: 10.3390/a15100373.
[9] S. Bhatia, N. Kush, C. Djamaludin, A. J. Akande, and E. Foo, “Practical modbus flooding attack and detection,” in Information Security Conference, Jan. 2014, p. 57. Accessed: Mar. 2025. [Online]. Available: http://crpit.com/confpapers/CRPITV149Bhatia.pdf
[10] X. Sun and Z. Chen, “A Novel Chaotic Image Encryption Algorithm Based on Coordinate Descent and SHA-256,” IEEE Access, vol. 10, p. 114597, Jan. 2022, doi: 10.1109/access.2022.3217520.
[11] A. D. Jurcut, T. Niculcea, P. Ranaweera, and N. Le-Khac, “Security Considerations for Internet of Things: A Survey,” SN Computer Science, vol. 1, no. 4, Jun. 2020, doi: 10.1007/s42979-020-00201-3.
[12] M. O’Neill, “Insecurity by Design: Today’s IoT Device Security Problem,” Engineering, vol. 2, no. 1, p. 48, Mar. 2016, doi: 10.1016/j.eng.2016.01.014.
[13] “Cryptography Security Against Instrusion In Automation Protocol ,” vol. 18, no. 4, p. 26, Aug. 2023.
[14] W. Wang et al., “Securing Cryptographic Chips against Scan-Based Attacks in Wireless Sensor Network Applications,” Sensors, vol. 19, no. 20, p. 4598, Oct. 2019, doi: 10.3390/s19204598.
[15] B. Tsao, Y. Liu, and B. Dezfouli, “Analysis of the duration and energy consumption of AES algorithms on a contiki-based IoT device,” p. 483, Nov. 2019, doi: 10.1145/3360774.3368202.
[16] S. Heron, “Advanced Encryption Standard (AES),” Network Security, vol. 2009, no. 12, p. 8, Dec. 2009, doi: 10.1016/s1353-4858(10)70006-4.
[17] D. Mukhopadhyay, G. Sonawane, P. Gupta, S. Bhavsar, and V. Mittal, “Enhanced Security for Cloud Storage using File Encryption,” arXiv (Cornell University), Jan. 2013, doi: 10.48550/arxiv.1303.7075.
[18] A. Biryukov and C. D. Cannière, “Data Encryption Standard (DES),” in Springer eBooks, Springer Nature, 2011, p. 295. doi: 10.1007/978-1-4419-5906-5_568.
[19] R. Venugopalan, P. Ganesan, P. Peddabachagari, A. G. Dean, F. Mueller, and M. L. Sichitiu, “Encryption overhead in embedded systems and sensor network nodes,” Oct. 2003, doi: 10.1145/951710.951737.
[20] P. Ganesan, R. Venugopalan, P. Peddabachagari, A. G. Dean, F. Mueller, and M. L. Sichitiu, “Analyzing and modeling encryption overhead for sensor network nodes,” Sep. 2003, doi: 10.1145/941350.941372.
[21] R. Babu, A. George, and K. Borasia, “A Review On Securing Distributed Systems Using Symmetric Key Cryptography,” arXiv (Cornell University). Cornell University, Jan. 01, 2013. doi: 10.48550/arxiv.1303.0351.
[22] M. Á. Platas-Garza, E. Zambrano-Serrano, J. R. Rodríguez-Cruz, and C. Posadas–Castillo, “Implementation of an encrypted-compressed image wireless transmission scheme based on chaotic fractional-order systems,” Chinese Journal of Physics, vol. 71, p. 22, Dec. 2020, doi: 10.1016/j.cjph.2020.11.014.

[23] W. Shang, Q. Qiao, M. Wan, and P. Zeng, “Design and Implementation of Industrial Firewall for Modbus/TCP,” Journal of Computers, p. 432, Jan. 2016, doi: 10.17706/jcp.11.5.432-438.
[24] E. D. Knapp and J. T. Langill, “Industrial Network Protocols,” in Elsevier eBooks, Elsevier BV, 2014, p. 121. doi: 10.1016/b978-0-12-420114-9.00006-x.
[25] Sameer S Nagtilak and Dr S R Chougule, “Modbus Hardware Architecture for Client server Configuration in Automation Industry,” vol. 10, no. 3, p. 258, 2023.
[26] F. Katulić, D. Sumina, S. Groš, and I. Erceg, “Protecting Modbus/TCP-Based Industrial Automation and Control Systems Using Message Authentication Codes,” IEEE Access, vol. 11, p. 47007, Jan. 2023, doi: 10.1109/access.2023.3275443.
[27] Sameer S Nagtilak and Dr S R Chougule, “Review of MODBUS and Various Components with Its Application along with Security Study Against Attacks ,” vol. 8, no. 6, 2021.
[28] F. J. Farsana, V. R. Devi, and K. Gopakumar, “An audio encryption scheme based on Fast Walsh Hadamard Transform and mixed chaotic keystream.” Oct. 11, 2019.
[29] Sameer S Nagtilak and Dr S R Chougule, “Security Design for Modbus protocol against intrusions during implementation on Industrial applications ,” vol. 12, no. 13, p. 2379, 2021.
[30] X. R. Tong and Z. B. Sheng, “Design of UART with CRC Check Based on FPGA,” Advanced materials research, p. 1241, Mar. 2012, doi: 10.4028/www.scientific.net/amr.490-495.1241.
[31] “Glossary of Security Terms,” in Elsevier eBooks, Elsevier BV, 2004, p. 711. doi: 10.1016/b978-155558306-4/50025-1.
[32] A. Nakassis, “Fletcher’s error detection algorithm: how to implement it efficiently and how toavoid the most common pitfalls,” ACM SIGCOMM Computer Communication Review, vol. 18, no. 5, p. 63, Oct. 1988, doi: 10.1145/53644.53648.
[33] S. Deep, X. Zheng, A. Jolfaei, D. Yu, P. Ostovari, and A. K. Bashir, “A survey of security and privacy issues in the Internet of Things from the layered context,” arXiv (Cornell University), Jan. 2019, doi: 10.48550/arxiv.1903.00846.
[34] X. Ma, H. Sun, R. Q. Hu, and Y. Qian, “Approximate Wireless Communication for Federated Learning,” p. 21, Jun. 2023, doi: 10.1145/3586209.3591399.
[35] J. Camerini and D. Dube, “MODBUS Application Protocol,” May 2002, Accessed: Feb. 2025. [Online]. Available: https://tools.ietf.org/html/draft-dube-modbus-applproto-00
[36] M. Lister and J. Wunderlich, “Development of software for mobile robot control over a radio frequency communications link,” p. 414, Jul. 2003, doi: 10.1109/secon.2002.995630.
[37] Sameer S Nagtilak and Dr S R Chougule, “Implementing Modbus along on PIC32MX795F512H for Data monitoring and control of sensor nodes using RS 485 ,” vol. 8, no. 2, p. 1740, 2021.