International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 04 | April - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Implementation of the Lightweight AES Algorithm with Parity Bit Fault Detection

Govinda Eswar[1], Badana Pravallika[2], Arangi Sai Krishna[3] , Pamidisetti Jyothika[4]

[1]*Department of ECE & GMR Institute of Technology*
[2]*Department of ECE & GMR Institute of Technology*
[3]*Department of ECE & GMR Institute of Technology*
[4]*Department of ECE & GMR Institute of Technology*

Abstract - In today's world, with the rising use of online communications tools and applications available on the internet regarding the existence of the Internet of Things, it is mandatorily required to ensure the safety of such sensitive information. In this scenario, regarding the secure encryption technique, the most essential task is to use the Advanced Encryption Standards or the AES Algorithm because this algorithm holds the ability to embody strong features regarding this entity. It would be the AES process that would involve an initial round, followed by some rounds, and ultimately end with a final round. During the diffusion phase, operations involving Sub Bytes, Shift Rows, Mix Columns, and Add Round Key would be caught up in the AES processes for the purpose of encrypting AES. It is obvious that for decryption processes, there would come operations involving Inv Sub Bytes, Inv Shift Rows, Inv Mix Columns, and ultimately, Add Round Key related to explaining operations for decryption processes. Another module would be incorporated to expand keys. The round keys would be produced with absolute synchronization and security.

Key Words: Advanced Encryption Standard (AES), Internet of Things (IoT), Data Encryption, Key Expansion, Round key Operations.

1. INTRODUCTION

Internet of Things (IoT) is the latest trend for the current generation of computing, as it provides free communication interface for many intelligent embedded systems for various applications such as smart homes, health care, industrial automation, intelligent transport systems, wireless sensor networks, and many others. In Internet of Things the constraints with respect to computing power, memory capacity, silicon space or power budget constraint is utilized for the device. Security and privacy also play a crucial role in IoT environments. Since devices are interconnected through the internet, they are vulnerable to various cyber threats such as data breaches, unauthorized access, and denial-of-service attacks. Therefore, lightweight cryptographic algorithms and secure authentication mechanisms are essential for protecting sensitive data in IoT applications.

For the problem related to various devices for various applications of internet of things, an issue has been generated, which is related to the implementation of the classified information, and therefore the confidentiality or integrity or security is required in cryptography to prevent cyber threats for authenticity is required for internet of things services. Communication technologies, coupled with the widespread adoption of Internet of Things (IoT) devices, has significantly increased the volume of data exchanged over shared communication networks. These interconnected devices are widely used in applications such as smart homes, healthcare monitoring, industrial automation, and wireless sensor networks, where sensitive information is frequently transmitted. Due to the open and distributed nature of these networks, communication systems are increasingly vulnerable to security threats such as unauthorized access, data interception, modification, and replay attacks. Consequently, ensuring secure data transmission has become a fundamental requirement in modern communication infrastructures.

Cryptography plays a vital role in protecting sensitive information by transforming plaintext data into an unintelligible form, thereby preventing unauthorized access. Effective cryptographic mechanisms are designed to ensure confidentiality, integrity, and authenticity of transmitted data. Among various cryptographic techniques, symmetric key encryption algorithms are widely preferred due to their high processing speed, reduced computational complexity, and lower resource requirements. These advantages make symmetric encryption particularly suitable for real-time and resource constrained environments, such as IoT and embedded systems, where power efficiency and fast execution are critical design considerations. The Advanced Encryption Standard (AES) is one of the most

widely adopted symmetric key encryption algorithms owing to its proven security, robustness against cryptanalytic attacks, and global standardization.

AES achieves a high level of security through multiple rounds of encryption that involve a combination of non-linear and linear transformations applied to fixed-size data blocks. These transformations introduce confusion and diffusion, which are essential properties for secure encryption. Due to its strong security guarantees and implementation of flexibility, AES is extensively employed in secure communication protocols, data storage systems, and embedded security applications. The AES algorithm follows a well-defined sequence of operations, including Sub Bytes, Shift Rows, Mix Columns, and Add Round Key, supported by a key expansion process that generates unique round keys for both encryption and decryption [1]. The Sub Bytes operation introduces nonlinearity through byte substitution using an S-box, while Shift Rows and Mix Columns ensure diffusion by rearranging and mixing data bytes across the state matrix. The Add Round Key operation integrates the secret key into the encryption process through bitwise XOR operations.

The decryption process performs the inverse of these operations in a similar structured manner, enabling secure recovery of the original plaintext. In this work, a round-based block diagram approach is adopted to present a comprehensive design of the AES encryption and decryption processes. This approach clearly illustrates the functional interaction between the encryption module, decryption module, and key expansion unit, thereby improving the understanding of the internal architecture of AES. By emphasizing the operational flow of each round and the role of round keys, the proposed design enhances clarity and modularity, facilitating efficient implementation, analysis, and optimization. Such structured representation is particularly beneficial for hardware-based realizations and further research aimed at developing secure and efficient cryptographic solutions for IoT applications. The proposed architectural representation emphasizes the operational flow of each AES round and clearly illustrates the interaction between the encryption module, decryption module, and key expansion unit [2].

By decomposing the AES algorithm into modular functional blocks, the design enhances clarity, scalability, and ease of implementation. This structured approach is particularly advantageous for hardware realization on platforms such as Field-Programmable Gate Arrays (FPGAs) and Application Specific Integrated Circuits (ASICs), were efficient resource utilization, high throughput, and low power consumption are critical. Furthermore, the improved clarity offered by the round based design facilitates future optimizations and the integration of additional security features, making it well suited for secure and efficient cryptographic implementations in IoT and embedded system applications. There has been a demand for the best possible cryptographic algorithms that can support the power needed by the networks that are used for the applications of the IoT. This has, therefore, ensured that the efficiency of the devices is kept for a longer period with improved energy efficiency.

The cryptographic algorithms that can work well with the hardware are anticipated to ensure that a significant amount of data is processed within a second for the security of the devices. It should be noted that the secure architecture is able to ensure the services of both authentication and updating of the devices. Besides the performance and power demands, the debate that involves scalability, malleability, and ability and capability to thwart threats and ability and capability, as mentioned above, will be needed in the future concerning the development and development of cryptography systems, based on the IoT perspective. This, in essence, is based on the fact that it is because of the processing levels that have resulted in an increased need for the carrying capacity of such systems in unique and different situations regarding interaction and communication that there has been a willingness or a belief in the necessity of developing systems and solutions that, apart from being power-efficient, have to be adaptable and have ability and capability to perform and accomplish in different situations, not necessarily having to undergo a process and procedure regarding implementation phase.

The whole development and Current trends in cryptographic security for Internet of Things (IoT) networks involve the development of lightweight encryption algorithms. Although the Advanced Encryption Standard (AES) algorithm is largely used because of its strong cryptographic primitives, its computationally complex operations remain a challenge to be overcome by IoT devices. One of the most vital and crucial needs that also need to be regarded and implemented is that there also needs to be proper and lacked understanding with regards to security that also needs to contain the easy integration ability regarding concepts and systems related to cryptography, access

control, and of course, without any costs involved in the process.

It is here, however, that the correspondence concerning concepts such as secure startup and remote firmware update processing falls in relevance within the context of capability enhancement of the devices in view of the vulnerabilities created because of the completion of the development phases of the devices.

2. RELATED WORK

S. Ahmed, N. Ahmad, et al. [1] presented an enhanced version of the lightweight algorithm for 128-bit AES. This is related to power, as well as memory and logic resources of the FPGAs, as presented above. The aim is to decrease power consumption as well as memory and logic resource requirements for a few cryptographic operations of AES algorithm reduction directly related to the complexity reduction of the algorithm, besides the hardware architecture optimization. The results show remarkable power reduction as well as LUT resources for both encryption and decryption processes. Though there is an enhancement regarding hardware complexity, there is an effect on complexity and security strength of the AES algorithm.

In the article referred to as Salman R. S. et al. [3], this problem has been described in a detailed comparative study related to the differences in AES for the IoT environment among lightweight algorithms. The various existing optimization algorithms to be compared for forming a logical deduction are minimization of processing operations, steps related to Mix Columns, Sub Bytes algorithm, and key scheduling algorithms. In general, one can form a conclusion related to over-efficiency being a cause for attack, even though efficiency is attained to some extent in AES among the various cited algorithms.

3. METHODOLOGY

AES is a symmetric key algorithm, which implies that for the process of encrypting as well as decrypting the data, the same key needs to be used. The main purpose for which AES was designed was to secure sensitive information by making any readable information (or, for the matter of fact, the plaintext) into any form of unreadable information, which is commonly known as ciphertext, in such a way that no unauthorized person can generate any useful information out of it. The truth is that AES is among the algorithms that are used the most because it has an excellent performance complexity ratio with respect to its usage of processing power as well as memory [4].

AES operates on fixed-size data blocks of 128 bits, which means that the input data is divided into chunks of 128 bits each for processing. It has to be noticed that the 128-bit chunk of data is processed through an internal 4x4 matrix of bytes, and it is common to refer to the 128bit matrix or state. Encryption and decryption in AES are performed through rounds, including specific mathematical and logical operations.

Such operations are designed according to the paradigm of the substitution permutation network, which introduces security by providing confusion-make the relationship between the key and the encrypted message complicated-and diffusion, the influence of one input bit on several encrypted bits. The entire encrypting function with a distinct level of security for every round in the encrypting and decrypting function. In truth, AES block diagram is significant for comprehending the AES operations in a systematic manner.

Next to it follows the process of encryption operation carried out by AES, where the input Plaintext of 128-bit size is given in a state matrix form, along with other operations carried out by the Initial Add Round Key process, considering key dependency too. Next, it follows the process of rounds carried out by AES and can be explained as follows using Verilog HDL, which revealed the improvements in performance, speed, and power consumption over single-cycle architectures [12].

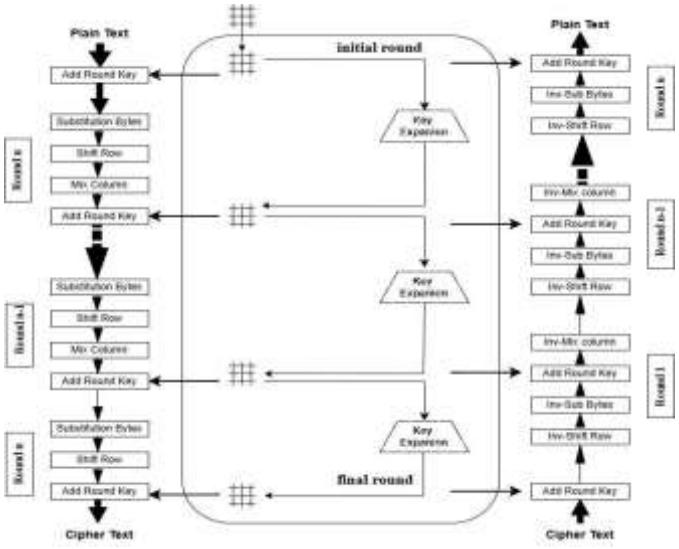

Fig-1: Block Diagram of Standard AES Algorithm

AES block diagram explains the design structure created for AES operations done during the encrypting and decrypting processes of a data piece utilizing the symmetric key. Basically, the design structure architecture for AES is comprised of three functions: the key expansion function, encrypting function, and

decrypting function. A key expansion function requires the use of the initial ciphering key for obtaining a sequence of keys for encrypting and decrypting in each round during the encrypting and decrypting functions because it intensifies the level of encryption bypassed the process of Mix Columns, and finally, the state matrix Ciphertext is acquired. Lastly, in conclusion, part of the encryption and decryption process in AES. The decryption process of AES in the AES module further continues with the reverse process of rounds of encryption of AES in a process similar to the above, utilizing the InvShiftRows, InvSubBytes, InvMixColumns, and AddRoundKey functions in the round keys of AES [1].

Lastly, the block diagram of the AES module is very informative about AES operations, and it spends a considerable amount of time discussing how various operations, such as described by encrypt, decrypt, and AES round keys of AES operations, have managed to get adequate clarity and focus regarding the radical effect of AES concerning providing solutions regarding the issue of scalability and ease of implement ability present in the domain of IoT devices with adequate clarity and insight.

The proposed methodology goes with the use of the complete AES algorithm for secure data transfer that works well for IoT and embedded systems [2]. In fact, the AES algorithm is a symmetric algorithm that works well in supporting data processing operations involving 128-bit data blocks. Symmetric keys used in the AES algorithm include those that measure either 128 bits, 192 bits, or 256 bits. This method is made up of the AES algorithm that supports the use of a substitution-permutation network design.

3.1 Plaintext Processing and State Matrix Formation

The AES (Advanced Encryption Standard) architecture consists of three main functional modules: Text Processing, Key Processing, and the Encryption & Decryption Process. These modules work together to perform secure symmetric key encryption and decryption operations. In AES, the input plaintext of 128 bits is arranged into a 4 × 4-byte matrix, known as the State Matrix. This state matrix represents the fundamental data unit on which AES operations are performed. All transformation steps involved in both encryption and decryption—such as substitution, permutation, and key addition—are executed on this state matrix at either the bit level or the word level.

3.2 Cipher Key Input and Key Expansion

The key expansion algorithm generates a set of round keys from the input cipher key that is to be used during the different AES rounds. It produces (Nr + 1) round keys, where Nr depends on the selected key size. This procedure consists of a cyclic word rotation denoted as Rot Word, nonlinear byte substitution by means of the S-box denoted as Sub Word, and addition of round constants denoted as Rcon. Each generated round key has a width of 128 bits and differs for every round.

3.3 AES ENCRYPTION PROCESS

The encryption process begins with the Add Round Key operation, known as Add Round Key, which uses the XOR of the plaintext state matrix and the key appropriate for the selected round of encryption. This is followed by the rounds of encryption. Each round of encryption consists of the performance of four operations, namely SubBytes, Shift Rows, MixColumns, and Add Round Key, in order. The SubBytes operation uses byte substitution as implemented by the predefined S-box. The Shift Rows operation entails the circular shift of the rows of the state matrix as implemented by the interbyte diffusion. The Add Round Key operation adds the key by the XOR operation. However, the last round of encryption omits the MixColumns operation to ensure the reverse feasibility of the process, hence obtaining the state matrix transformed into the ciphertext.

3.4 AES Decryption Process

The decryption algorithm performs the reverse operations for the entire process conducted in the decryption process using the inverse operations. The decryption process initializes with the initial Add Round Key step, whose requirement is the last round key. In the decryption round operation, the requirement involves conducting the operations of InvShift Rows, InvSubBytes, Add Round Key, and InvMixColumns[1]. All these reverse operations perform the reverse of the entire process conducted in the decryption operation. ing key for obtaining a sequence of keys for encrypting and decrypting in each round during the encrypting.

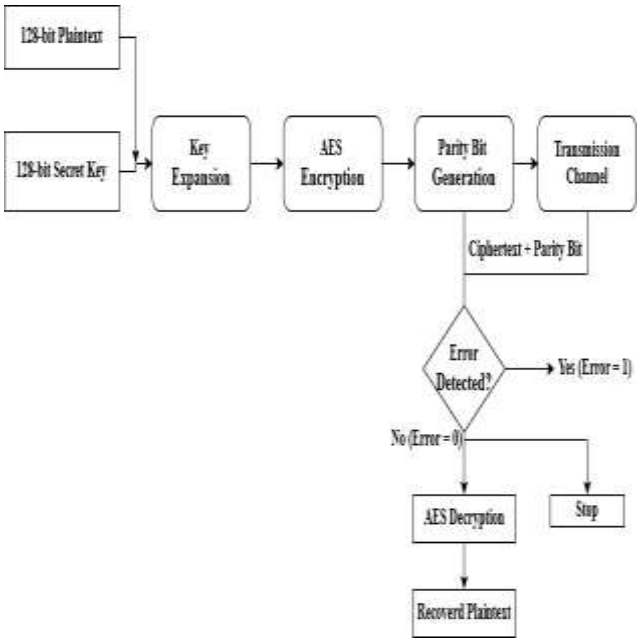

Fig-2: Process Flow of the Proposed Lightweight AES Architecture

The proposed system consists of multiple stages including input processing, key expansion, AES encryption, parity bit generation, transmission channel, error detection, and AES decryption. Initially, a 128-bit plaintext and a 128-bit secret key are provided as inputs to the system. The key is processed through the key expansion module to generate round keys required for encryption.

The AES encryption block performs multiple transformation rounds to convert plaintext into ciphertext. After encryption, a parity bit is generated from the ciphertext, which is used for error detection. The ciphertext along with the parity bit is transmitted through the communication channel.

At the receiver's side, the system checks whether an error has occurred during transmission. If no error is detected (Error = 0), the ciphertext is passed to the AES decryption module to recover the original plaintext. If an error is detected (Error = 1), the system stops the process, preventing incorrect output. This ensures reliable communication by allowing only error-free data to be decrypted.

3.5 Parity Bit Generation

The parity bit is generated using a simple XOR operation across all bits of the ciphertext. This ensures that the parity represents the overall bit structure of the data.

$$P = \bigoplus_{i=1}^{n} b_i$$

where b_i represents each bit of the ciphertext. The generated parity bit is transmitted along with the ciphertext to enable error detection at the receiver's side.

3.6 Error Detection Mechanism

At the receiver, the parity is recalculated and compared with the received parity value.

$$P_{received} = P_{calculated}$$

If both values are equal, the data is considered error-free, and the system proceeds with decryption. If a mismatch occurs, an error is detected, and the system halts further processing. This mechanism ensures that corrupted data is not decrypted, thereby improving system reliability.

3.7 Design Flow of 128-bit Lightweight AES algorithm using Parity bit

The working of the proposed system begins with the input of plaintext and secret key, which are processed through the key expansion module. The AES encryption block then converts the plaintext into a ciphertext using multiple rounds of transformation. After encryption, a parity bit is generated and transmitted along with the ciphertext through the communication channel.

At the receiver's side, the system evaluates whether an error has occurred using parity comparison. If no error is detected, the AES decryption module is activated to recover the original plaintext. If an error is detected, the system will stop the process. This flow ensures both secure encryption and reliable data transmission.

3.8 Difference Between Standard AES and Proposed Lightweight AES

The proposed lightweight AES system differs from the standard AES algorithm in terms of architectural design, efficiency, and reliability. While the standard AES algorithm focuses primarily on providing strong encryption through multiple transformation rounds, it does not include any mechanism for detecting errors during data transmission. In contrast, the proposed system enhances the AES architecture by integrating a parity-bit-based fault detection mechanism, which improves data integrity and system reliability. This additional feature ensures that errors occurring during transmission are identified before decryption, thereby

preventing incorrect outputs and enhancing overall system performance.

4 RESULTS

This section discusses the simulation results, resource utilization, and power analysis of the proposed lightweight AES encryption and decryption architecture with an integrated parity bit-based security mechanism. The simulation results are obtained through functional simulation and FPGA synthesis to ensure the correctness, efficiency, and reliability of the proposed design. simulation results are analyzed to determine the applicability of the proposed design for resource-constrained Internet of Things (IoT) and embedded systems, where low area overhead, low power consumption, and improved data integrity are essential requirements. The proposed design with the parity bit-based security mechanism offers an additional layer of security and reliability compared to conventional AES designs, making it more robust against transmission errors and fault attacks.

Fig-3: Simulation Waveform of the AES Encryption and Decryption Process

Figure 3 illustrates the simulation waveform of the proposed lightweight AES encryption and decryption architecture integrated with a parity bit–based error detection mechanism. The waveform represents the behavior of different signals involved in the encryption and decryption processes during functional simulation. It provides a clear visualization of how the system processes the plaintext input, encryption operation, parity verification, and final decryption output over time.

Fig-4: Simulation waveform illustrating parity bit–based error detection, where decryption is enabled only when error = 0

In this particular simulation scenario, a fault signal is introduced to simulate an error or corruption in the encrypted data. Such faults may occur due to transmission errors, hardware faults, or malicious fault injection attacks in practical systems. The introduction of this fault causes a mismatch during the parity verification stage, which is responsible for checking the integrity of the ciphertext generated by the encryption module.The parity checking mechanism analyzes the output data and compares the calculated parity with the expected parity value. Because a fault has been introduced, the parity values no longer match. As a result, the error signal becomes high (error = 1), indicating that the ciphertext has been corrupted or altered.

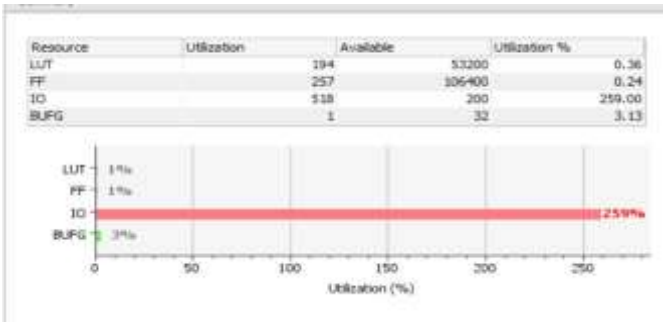

Fig-5: FPGA slice LUT utilization report of the proposed lightweight AES

The report shows that 194 LUTs are utilized out of available LUTs, which corresponds to approximately 0.36% utilization. LUTs are the primary combinational logic resources in FPGA devices and are used to implement logic operations required for AES transformations such as substitution, permutation, and key processing. The very small percentage of LUT usage indicates that the proposed AES architecture is highly resource-efficient and occupies only a small portion of the available logic resources.

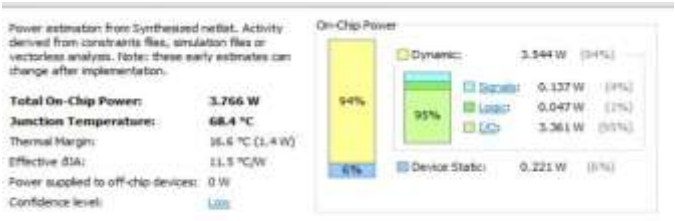

Fig 4.4: On-chip power consumption analysis of the proposed
AES design

According to the report, the estimated total on-chip power consumption is 3.766 W. This total power consists of two main components: dynamic power and static power. The dynamic power consumption is 3.544

W, which accounts for approximately 94% of the total power consumption.

Table 1: Comparison table of standard AES with the proposed lightweight

Parameter	Standard AES	Proposed Lightweight AES
LUT Usage	2800	194
Registers	1900	257
Area Utilization	3.5%	0.36%
Power Consumption	9W	3.766W

5. CONCLUSION

In conclusion, the proposed AES encryption and decryption system is a secure and efficient means of protecting sensitive information in modern communication networks, especially in Internet of Things (IoT) communication networks. The AES algorithm has a well-structured round-by round process that ensures high levels of confidentiality through operations such as SubBytes, Shift Rows, MixColumns, and Add Round Key. The final round of encryption does not involve the MixColumns operation, while the decryption process involves the reverse operation of the encryption process. To enhance security, an error detection system using the parity bit is also included in the system. The parity bit is generated during the encryption process and transmitted alongside the encrypted message. At the receiving end, the parity bit is verified before proceeding with the decryption process. If error = 0, the decryption process is undertaken otherwise; decryption is blocked, preventing faulty or tampered data from being processed. This mechanism improves data integrity and resistance to transmission errors and fault-based attacks.

ACKNOWLEDGEMENT

The authors would like to express their sincere gratitude to the project guide, Dr. B.G. B.S. R Naidu Sir, for his valuable guidance, continuous support, and encouragement throughout the completion of this work.

The authors also extend they're thanks to the Head of the Department, Dr. V. Jagan Naveen, for providing the necessary facilities and support.

The authors are thankful to all faculty members for their guidance and cooperation.

Finally, the authors express their gratitude to their family and friends for their constant support and motivation.

REFERENCES

1. S. Ahmed, N. Ahmad, N. A. Shah, G. E. M. Abro, A. Wijayanto, A. Hirsi, and A. R. Altaf, “Lightweight AES design for IoT applications: Optimizations in FPGA and ASIC with DFA countermeasure strategies,” IEEE Access, vol. 13, 10.1109/ACCESS.2025.3533611. pp. 22489–22503, 2025, doi:
2. X. Huo and X. Wang, “Internet of Things for smart manufacturing based on advanced encryption standard (AES) algorithm with chaotic system,” Results in
Engineering, vol. 20, Art. no. 101589, 10.1016/j.rineng.2023.101589. Nov. 2023, doi:
3. R. S. Salman, A. K. Farhan, and A. Shakir, “Lightweight modifications in the advanced encryption standard (AES) for IoT applications: A comparative survey,” in Proc. 2022 Int. Conf. Computer Science and Software Engineering (CSASE), Duhok, Iraq, 2022, pp. 325–330, doi: 10.1109/CSASE51777.2022.9759828.
4. N. Su, Y. Zhang, and M. Li, “Research on data encryption standard based on AES algorithm in Internet of Things environment,” in Proc. 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conf. (ITNEC), Chongqing,
China, 2019, pp. 2071–2075, doi: 10.1109/ITNEC.2019.8729310.
5. T. B. Singha, R. P. Palathinkal, and S. R. Ahamed, “Securing AES designs against power analysis attacks: A survey,” IEEE Internet of Things Journal, vol. 10, no. 16, pp. 14332–14356, Aug. 2023.
6. S. Pandey and B. Bhushan, “Recent lightweight cryptography–based security advances for resource constrained IoT networks,” Wireless Networks, vol. 30, no. 4, pp. 2987–3026, Mar. 2024.
7. S.-N. Tran, V.-T. Hoang, and D.-H. Bui, “A hardware architecture of NIST lightweight cryptography applied in IPsec to secure high-throughput, low latency IoT networks,” IEEE Access, vol. 11, pp. 89240–89248, 2023.
8. R. Serrano, C. Duran, M. Sarmiento, C.-K. Pham, and T.-T. Hoang, “ChaCha20–Poly1305 authenticated encryption with additional data for transport layer security 1.3,” Cryptography, vol. 6, no. 2, p. 30, Jun. 2022.

9. Y. Zhong and J. Gu, “Lightweight block ciphers for resource constrained environments: A comprehensive survey,” Future Generation Computer Systems, vol. 157, pp. 288–302, Aug. 2024.
10. H. Wen, Y. Lin, L. Yang, and R. Chen, “Cryptanalysis of an image encryption scheme using variant Hill cipher and chaos,” Expert Systems with Applications, vol. 250, Art. no. 123748, Sep. 2024.
11. Y. Cheng, Y. Liu, Z. Zhang, and Y. Li, “An asymmetric encryption-based key distribution method for wireless sensor networks,” Sensors, vol. 23, no. 14, p. 6460, Jul. 2023. A. S. D. Alluhaidan and P. Prabu, “End-to-end encryption in resource constrained IoT devices,” IEEE Access, vol. 11, pp. 70040–70051, 2023.
12. J. Daemen and V. Rijmen, The Design of Rijndael: AES— The Advanced Encryption Standard. Cham, Switzerland: Springer, 2001.
13. A. S. D. Alluhaidan and P. Prabu, “End-to-end encryption in resource constrained IoT devices,” IEEE Access, vol. 11, pp. 70040–70051, 2023.
14. J. Daemen and V. Rijmen, The Design of Rijndael: AES— The Advanced Encryption Standard. Cham, Switzerland: Springer, 2001.
15. M. Katagi and S. Moriai, “Lightweight cryptography for the Internet of Things,” IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. E100-A, no. 1, pp. 1–12, Jan. 2017.