

FPGA-Based AES-128 Cryptographic Processor with Resource Optimization

E. Immanuvel Bright³

³Assistant Professor, Department of Electrical and Electronics Engineering, Erode Sengunthar Engineering College, Perundurai, Erode, Tamil Nadu, India - 638057. immanuvelbright@gmail.com

ABSTRACT: For secure and efficient data transmission the increasing demand have created a need for hardware-based cryptographic solutions capable of providing high security with low resource utilization. However, existing software-based encryption techniques introduce computational overhead and increased execution latency, particularly in resource-constrained and high-speed embedded systems. To overcome these issues, this paper proposes a Field-Programmable Gate Array-Advanced Encryption Standard-128 (FPGA-based AES-128) cryptographic processor with resource optimization for secure data encryption and decryption. The proposed architecture implements the AES-128 using a 128-bit plaintext block and a 128-bit secret key. The encryption procedure contains key expansion, initial AddRoundKey, and ten encryption rounds, where every round incorporates the SubBytes, ShiftRows, MixColumns, AddRoundKey transformations. A round counter controls the sequential execution of the encryption operations, which is employed to reduce FPGA resource consumption. The proposed processor is designed and implemented on an FPGA platform, and its performance is evaluated in using proposed FPGA based AES-128 have the encryption throughput of 14,648 ps for ciphertext (128-bits). The cryptographic security and FPGA provide a secure data encryption.

Keywords: AES-128 encryption algorithm, FPGA, 128-bit plaintext block, 128-bit secret key, secure data encryption and decryption.

1. Introduction

FPGA is a reconfigurable digital hardware device that is programmed to implement specific computational operations directly in hardware. It consists of programmable logic blocks, LUTs, flip-flops, interconnections, memory, and other hardware resources. AES operations such as SubBytes, ShiftRows, MixColumns, AddRoundKey, and key expansion are implemented using FPGA logic. This allows parallel processing, high encryption throughput. The quick development of connected embedded and Internet of Things (IoT) [1] devices have increased the need for efficient hardware-based security mechanisms capable of protecting

sensitive data during storage and transmission [2]. In resource-constrained systems, cryptographic implementations must provide strong security whereas maintaining low hardware utilization, power consumption, and processing latency [3-4]. Consequently, FPGA are well suited for this purpose because their reconfigurable architecture allows cryptographic algorithms [5] to be implemented with high parallelism and application-specific optimization. Among the available symmetric encryption techniques, the AES [6] is widely adopted for securing digital information because of its computational efficiency and established security properties [7]. Researchers have developed various methods to secure data encryption and decryption.

2. Literature Survey

Several techniques are utilized for secure data encryption and decryption. In [8] Dhanda, et al (2025) have developed FPGA based AES-8 cipher suitable for application in IoT security. FPGA parallelism accelerates AES encryption and reduces processing latency. However, achieving high operating frequency requires careful optimization of the critical path. Consequently, in [9] Yang et al (2025) have proposed a scalable data-encryption approach evaluates the performance of AES encryption then decryption using Cipher Block Chaining (CBC) as well as Counter (CTR) approaches. AES provides a recognized symmetric encryption mechanism for protecting data confidentiality. Nevertheless, performance varies considerably with architecture, memory bandwidth, and implementation. Additionally, in [10] Zhang et al (2025) have invented AES it provides strong security and efficient encryption. AES provides robust protection against unauthorized access and data breaches. Nonetheless, AES modes have limit throughput when handling massive data streams. Furthermore, in [11] Zhang et al (2026) have developed security evaluations that the Shannon entropy of the generated round keys is high, indicating strong randomness and reduced predictability of the cryptographic key material. However, high Shannon entropy alone does not prove that an AES implementation is secure. Moreover, in [12] Ahmed et al (2025) have proposed Lightweight AES architectural on FPGAs to achieve secure encryption with reduced resource utilization. Nevertheless, performance varies according to the FPGA architecture and available resources. For this reason this paper proposed an FPGA based AES-128 technique.

2.1 Problem statement

The increasing volume of sensitive data transmitted through embedded and IoT systems requires secure and efficient encryption mechanisms that provide strong data protection with low computational and hardware overhead.

Existing AES suffer from securing data encryption and decryption, increased execution latency, particularly in resource-constrained and high-speed embedded systems. Software-based encryption introduce computational overhead, higher processing latency, and increased resource requirements, particularly in resource-constrained embedded and IoT systems. In addition, implementing cryptographic algorithms on hardware requires an appropriate balance between security, processing speed, FPGA resource utilization, and power consumption. To overcome these issues, this paper proposes an FPGA based AES-128 implementation for securing data encryption and decryption.

2.2 Research Contribution

- A FPGA based AES-128 cryptographic processor is developed for secure encryption and decryption whereas reducing FPGA resource utilization.
- The architecture processes 128-bit plaintext blocks using a 128-bit secret key, incorporating key expansion, initial AddRoundKey, and ten standard AES encryption rounds.
- A round counter and control mechanism are employed to coordinate the sequential execution of AES transformations and ensure reliable encryption operation.

3. Proposed Methodology

A resource-optimized FPGA-based AES-128 cryptographic processor is proposed for secure and efficient data encryption and decryption. A 128-bit plaintext block then a 128-bit secret key is provided to the processor, through the plaintext represented as a standard AES state matrix. The 128-bit secret key undergoes AES-128 key expansion to generate eleven 128-bit round keys using the RotWord, SubWord, in addition to Round Constant (Rcon) operations. The encryption begins with an initial AddRoundKey operation, in which the plaintext state is XOR

through the initial round key. Subsequently, ten AES encryption rounds are executed under the control of a dedicated round counter and control logic. In rounds 1–9, the state passes sequentially through SubBytes, ShiftRows, MixColumns, and AddRoundKey, whereas the tenth round performs SubBytes, ShiftRows, as well as AddRoundKey without MixColumns, as specified by the AES standard. To reduce complexity, the AES data path incorporates optimized structures, resource sharing, and efficient control logic, thereby targeting lower LUT, flip-flop, memory, and overall FPGA resource utilization. Following the final round, the resulting 128-bit state is generated as the ciphertext. Figure 1 shows the proposed block diagram for FPGA based AES-128.

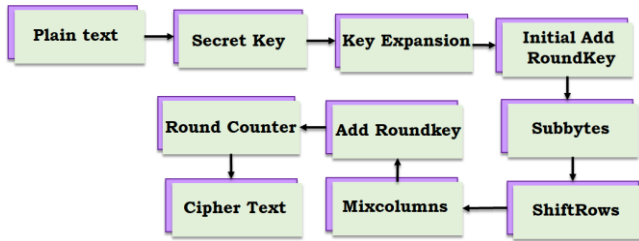


Figure 1: Proposed block diagram for FPGA based AES-128

3.1 FPGA based AES-128

For securing data encryption and decryption on optimized cryptographic resource using FPGA based AES-128. AES-128 is a symmetric-key block cipher that encrypts data using a 128-bit secret key and processes data in 128-bit blocks. It performs 10 encryption rounds, providing strong confidentiality for transmitted and stored data. A plaintext contains 128-bit and secret key are processed through key expansion, initial AddRoundKey, and ten AES rounds involving SubBytes, ShiftRows, MixColumns, and AddRoundKey operations. Figure 2 displays the operation of AES-128.

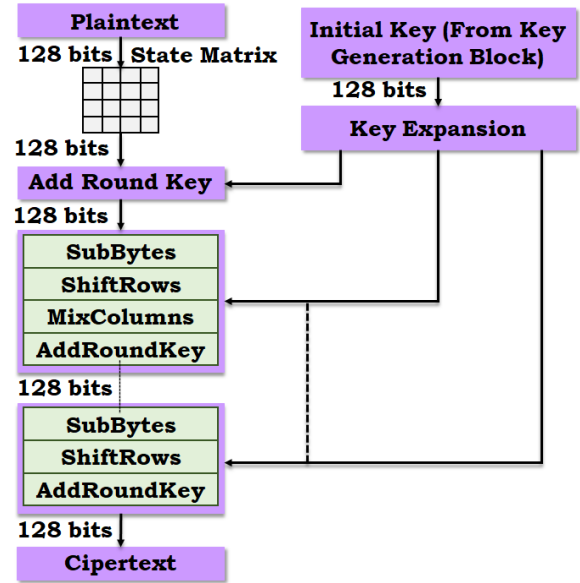


Figure 2: Operation of AES-128

FPGA is a reconfigurable semiconductor device that is programmed after manufacturing to perform specific digital computing and control operations. An FPGA implements operations directly as hardware logic, allowing parallel and application-specific processing. An FPGA generally consists of CLB, LUT, flip-flops, programmable interconnections, input/output blocks, and, depending on the device, embedded memory and DSP resources.

For an FPGA-based AES-128 cryptographic processor, the AES operations are mapped onto these programmable resources. The SubBytes, ShiftRows, MixColumns, AddRoundKey, and key-expansion operations are implemented as dedicated logic circuits. A control unit and round counter coordinate the execution of the ten AES-128 encryption rounds. The AES architecture is designed to minimize the use of LUTs, flip-flops, block RAM and other resources whereas maintaining the required encryption throughput.

3.2 Plaintext

In the plain text the unencrypted input data is divided into blocks of 128 bits for securing the data encryption and decryption. Using the initial stage, the 128-bit plaintext is XOR through the initial round key to generate the initial state matrix. The subsequent AES encryption process

consists of four fundamental transformations. The SubBytes operation achieves a nonlinear byte-level replacement using a secure 8×8 S-box, providing cryptographic confusion in addition to serving as the major component improved in the proposed architecture. The ShiftRows and MixColumns operations contribute to data diffusion and are retained without modification. The AddRoundKey process achieves a bitwise XOR among the transformed state in addition to the equivalent round key, thereby integrating the secret-key information into each encryption round.

3.3 State Matrix

The state matrix gets the data from the plain text as 128 bits. The 128-bit blocks are broken down into 16 bytes. Further, these bytes are arranged into a 4×4 grid called the state matrix. By this state matrix the mathematical operations in the algorithm is modified as matrix.

3.4 Initial Key

In the initial key a 128-bit secret key is supplied by the AES-128. The primary cryptographic key from all required round keys are generated through the AES key expansion procedure. The initial AddRoundKey operation is by performing a bitwise XOR with the 128-bit plaintext state.

During key expansion, the novel 128-bit key is distributed into four 32-bit words. The AES key-scheduling procedure applies RotWord, SubWord, and XOR operations with the round constants (Rcon) to generate new words. These words are combined to produce the next 128-bit round key. This process continues until all eleven round keys required by AES-128 are generated.

After the round keys are generated, the plaintext undergoes the initial AddRoundKey operation. The input plaintext is a 128-bit block and is arranged into a state matrix containing 16 bytes. The initial round key is also represented as a 128-bit state. The resulting state becomes the input to the first AES encryption round. The initial AddRoundKey is not a substitution or diffusion

operation; its primary purpose is to introduce the secret-key information into the plaintext before the main transformation rounds begin.

3.5 Key Expansion

The 128-bit initial key passes through a key expansion. The key expansion expands the single initial key into a series of multiple keys called round keys. For AES-128, the algorithm runs for 10 rounds. Every round requires single round key, meaning the key expansion routine generates 11 round keys in total that is one for the pre-round setup as well as one for every 10 subsequent rounds.

3.6 Add Round Key

In add round key before entering the iterative loops, the 128-bit plaintext state matrix undergoes an initial add round key transformation. Here, the state matrix is mutual through the original initial key by a bitwise XOR operation. After the pre-round step, the data enters an iterative loop. For AES-128, this consists of 9 identical main rounds. Every standard round contains

SubBytes (Substitution) a non-linear substitution step where every byte in the state matrix is changed through another byte using a fixed lookup table termed the S-box. This provides confusion to attack algebraic attacks.

ShiftRows (Permutation) is a transposition step where the rows of the state matrix are lifted regularly to the left by varying offsets. The 1st row isn't lifted; the 2nd row is lifted by 1 byte, the 3rd by 2 bytes, and the 4th by 3 bytes. This distributes the data across columns.

MixColumns (Linear Mixing) is a procedure that activates on the columns of the state matrix, relating the four bytes in every column by a linear mathematical transformation. This provides deep diffusion across the rows.

AddRoundKey is a modified matrix here bitwise XOR with the unique round key generated for that specific round from the key expansion block.

3.7 Ciphertext

For securing data encryption and decryption after the final round key is added, the state matrix is read out as a sequential stream of 128 bits, subsequent in the final scrambled, encrypted ciphertext. In the finishing round, the MixColumns transformation is omitted, then the ciphertext is generated after applying SubBytes, ShiftRows, and AddRoundKey. By this the encryption and decryption data are secured using FPGA based AES-128. In the FPGA operation, the final-round operations are realized using dedicated hardware logic. The round-control unit identifies the tenth round and disables the MixColumns block whereas allowing the required SubBytes, ShiftRows, and AddRoundKey operations. Once processing is completed, the FPGA outputs the 128-bit ciphertext. During decryption, the corresponding inverse transformations InvShiftRows, InvSubBytes, and AddRoundKey, together with InvMixColumns in the appropriate rounds are applied using the required round keys to recover the original plaintext.

Therefore, the FPGA-based AES-128 architecture provides a software mechanism for secure encryption and decryption of 128-bit data blocks, whereas FPGA implementation provides efficient, deterministic processing with optimized resource utilization.

4. Result and Discussion

For securing data encryption and decryption the FPGA based AES-128 is implemented. The simulation is implemented in Xilinx ISE.

Table 1: AES algorithm deviations with equivalent rounds and keys.

AES versions	Number of rounds	Key size	Number of keys
AES-128	10	128	11
AES-192	12	192	13
AES-256	14	256	15

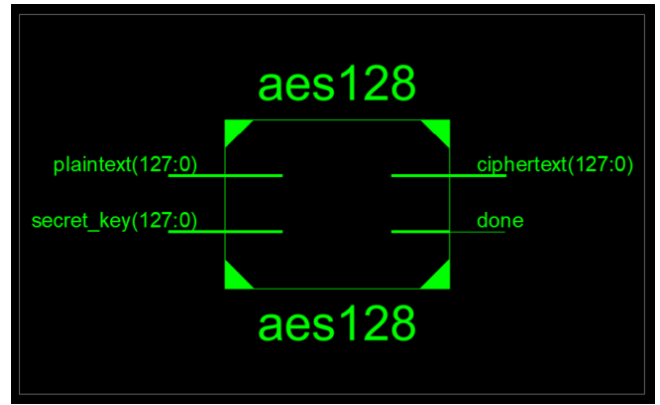


Figure 3: RTL (Register-Transfer Level) schematic block for AES-128

Figure 3 shows an RTL (Register-Transfer Level) schematic block symbol for an AES-128 encryption module, it is generated in FPGA design. The plaintext contains (127:0) input data line that takes a 128-bit block of unencrypted data. In secret_key (127:0) the input key line that takes a 128-bit encryption key, ciphertext (127:0) contains output data line that provides the resulting 128-bit encrypted data. The done output status signal that goes high when the encryption cycle finishes.

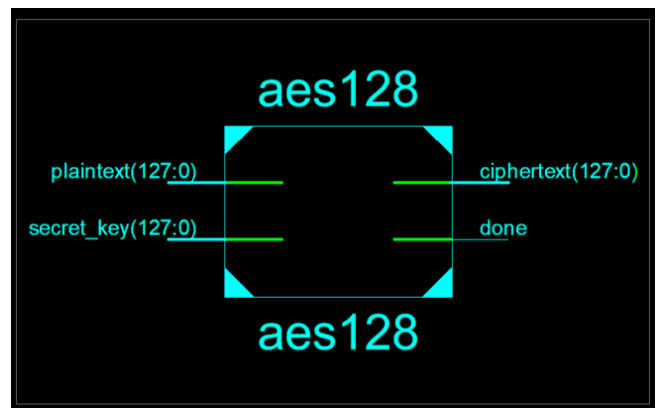


Figure 4: RTL schematic for AES-128

Figure 4 displays an RTL schematic block for an AES-128 encryption module. The plaintext contains (127:0) input data line that takes a 128-bit block of unencrypted data. In secret_key (127:0) the input key line that takes a 128-bit encryption key, the above AES-128 operation is done inside this ciphertext (127:0) contains output data line that provides the resulting 128-bit encrypted data. The done output status signal that goes high when the encryption cycle finishes.

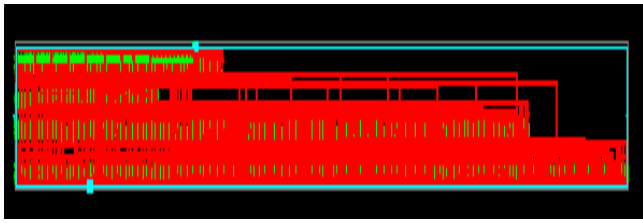


Figure 5: Data path in circuit design

Figure 5 shows the data path in circuit design. The shapes represent interconnected routing wires, cells, logic blocks in an integrated circuit on FPGA based AES-128 for securing data encryption as well as decryption.

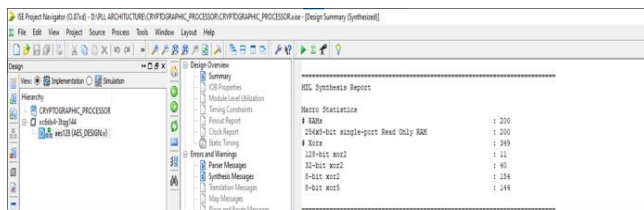


Figure 6: HDL synthesis report

Figure 6 displays the HDL synthesis report. For an AES-128 cryptographic processor the design is implemented in AES S-Box look-up tables. Total XORs is 349, 128-bit xor2: 11, 32-bit xor2: 40, 8-bit xor2: 154, 8-bit xor5: 144 for securing the data.

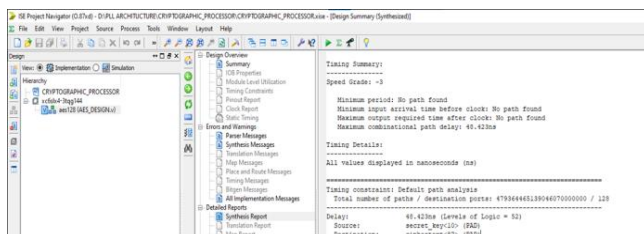


Figure 7: Timing summary

Figure 7 shows the timing summary for an AES-128 cryptographic design. The maximum combinational path delay is 48.423 ns; the signal takes a long time to propagate through 52 levels of logic from the input secret_key 10 to the output ciphertext 87. The path count is analyzed as 4793644651390460700000000 / 128 for secure data encryption.

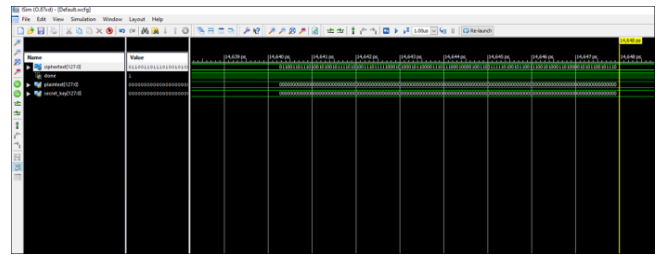


Figure 8: Ciphertext (128-bits)

Figure 8 displays a Ciphertext (128-Bits). The done signal is stated as high (1), and the ciphertext bus demonstrates changing binary data whereas the input plaintext and secret_key remain at zero for securing the data from FPGA based AES-128.

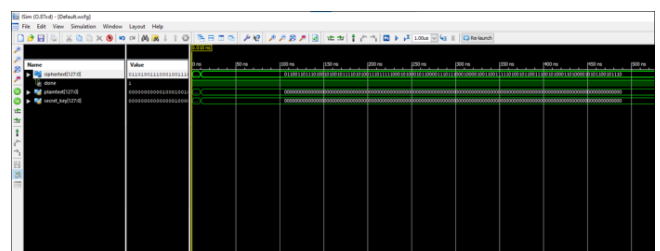


Figure 9: Done signal (1-Bit)

Figure 9 shows the done signal for securing the data encryption. The signals displayed a ciphertext (127:0) with bus value as 011010011100010011..., and done value as 1, time Scale value is 1.00 display stable values across the execution.

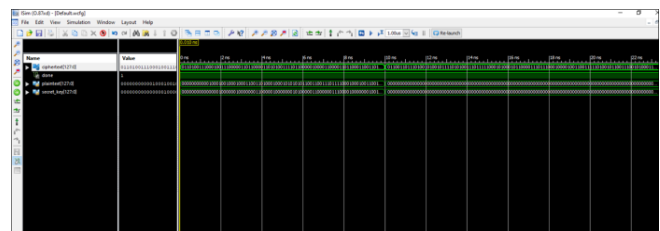


Figure 10: Final output using FPGA based AES-128

Figure 10 displays the final output using FPGA based AES-128. The final ciphertext (127:0) have 128-bit output data bus showing encrypted data values. Done have 1-bit control/status signal that goes high (1) to indicate the operation is finished. The plaintext (127:0) and secret_key (127:00) have the 128-bit input data bus with the original unencrypted message as 0.

4.1 Comparison with the proposed technique

In the evaluation with proposed technique the proposed FPGA based AES-128 method is

compared with the existing techniques as presented in table 2.

Table 2: Comparison with the proposed technique

Study	Encryption throughput (bps)
Zhang et al [10]	3500
Zhang et al [11]	4200
Proposed Work	1,46,480

For secure data encryption and decryption Zhang et al [10] have the encryption throughput of 3500 bps, Zhang et al [11] have the encryption throughput of 4200 bps and proposed FPGA based AES-128 have the encryption throughput of 1,464,80 bps for ciphertest (128-bits).

5. Conclusion

In this paper a FPGA based AES-128 cryptographic processor is proposed for secure and efficient data encryption and decryption. The proposed architecture processes 128-bit plaintext blocks using a 128-bit secret key then performs key expansion, initial AddRoundKey, and ten AES encryption rounds under the control of a dedicated round counter. Optimized structures are incorporated to reduce FPGA resource utilization whereas maintaining reliable cryptographic processing. The implemented processor achieves an encryption throughput of 146,480 bps for 128-bit ciphertext blocks, demonstrating its capability for secure data processing. The combination of AES-128 security, optimized FPGA implementation, and efficient encryption processing for secure communication applications. Future work focuses on AES-256 and other lightweight cryptographic algorithms for applications with different security and resource requirements.

References

1. VSS Pavan Nishith; Adriza Saha; Ashish Patwari, Year: 2025, "A novel modification and hardware implementation of the simplified AES algorithm for IoT applications", Results in Engineering, pp. 106567.
2. Jeya Bright Pankiraj; Vishnuvarthanan Govindaraj; Pallikonda Rajasekaran Murugan; Arun Prasath Thiyagarajan, Year: 2019, "Development of a scalable coding for the encryption of images using Diagonal Min-Max Block Truncation Code", Int J Eng Advan Technol, Vol: 9, pp. 718-723.
3. Jeya Bright Pankiraj; Vishnuvarthanan Govindaraj; Yudong Zhang; Pallikonda Rajasekaran Murugan; Anisha Milton, Year: 2022, "Development of scalable coding of encrypted images using enhanced block truncation code", Webology, Vol: 19, No: 1, pp. 1620-1639.
4. Dilip Kumar Sharma; Ningthoujam Chidananda Singh; Daneshwari A. Noola; Amala Nirmal Doss; Janaki Sivakumar, Year: 2022, "A review on various cryptographic techniques and algorithms", Materials Today: Proceedings, Vol. 51, pp. 104-109.
5. Arun Sekar Rajasekaran; S Shanmuga Priya; Janaki Sivakumar; Akshay Varkale, Year: 2023, "Cybersecurity Measures Using Machine Learning for Business Applications", In 2023 4th International Conference on Computation, Automation and Knowledge Management (ICCAKM), IEEE, pp. 01-08.
6. Sahara AS Almola; Hameed A. Younis; Raidah Khudeyer, Year: 2026, "A Robust Image Encryption Framework Using Deep Feature Extraction and AES Key Optimization", Cryptography, Vol: 10, No: 2, pp. 16.
7. Mohammad A Altahat; Tariq Daradkeh; Anjali Agarwal, Year: 2024 "Optimized encryption-integrated strategy for containers scheduling and secure migration in multi-cloud data centers", IEEE Access, Vol: 12, pp. 51330-51345.
8. Sumit Singh Dhanda; Brahmjit Singh; Poonam Jindal; Vinod Kumar; Sachin Kumar Gupta, Year:

- 2025, “AES-8: A Lightweight AES for Resource-Constrained IoT Devices”, Transactions on Emerging Telecommunications Technologies, Vol: 36, No: 3, pp. e70094.
9. Min Kyu Yang; Jae-Seung Jeong, Year: 2025, “Optimized Hybrid Central Processing Unit–Graphics Processing Unit Workflow for Accelerating Advanced Encryption Standard Encryption: Performance Evaluation and Computational Modeling”, Applied Sciences, Vol: 15, No: 7, pp. 3863.
 10. Yanxin Zhang; Xi Su, Year: 2025, “Application of deep learning in the optimization of aes encryption algorithm”, Journal of Cyber Security and Mobility, Vol: 14, No: 3, pp. 553-576.
 11. Yanxin Zhang, Year: 2026, “Lightweight Deep Learning Optimization of AES for Mobile IoT Security”, Journal of Cyber Security and Mobility, Vol: 15, No: 4, pp. 1133-1160.
 12. Salman Ahmed; Nabihah Ahmad; Nasir Ali Shah; Ghulam E Mustafa Abro; Ardhi Wijayanto; Abdinasir Hirsi; Abdul Rehman Altaf, Year: 2025, “Lightweight aes design for iot applications: Optimizations in fpga and asic with dfa countermeasure strategies”, IEEE Access, Vol: 13, pp. 22489-22509.