Grenze International Journal of Engineering and Technology, June Issue

Secure Edge Wireless Communication with SPECK Lightweight Cryptography

Sudha K.L[1], Navya Holla K[2], Vidyashree K N[3], Manasa R[4], Chaitra A[5] and Trupti Shripad Tagare[6]
[1-6]Dayananda Sagar College of Engineering, K.S Layout, Bengaluru, India
Email: drsudha-ece@dayanandasagar.edu, hollanavya@gmail.com, knvidyashree@gmail.com, manasar.87@gmail.com, chaitra.a9@gmail.com, truptitagare-ece@dayanandasagar.edu

***Abstract*— The need for secure, low-latency, and energy-efficient communication modules are the main requirements of Internet of Things (IoT), and made wireless sensor networks (WSN), and edge intelligence systems to be part of the system. This paper presents the design and implementation of a SPECK-based wireless data transmission and receiving module using ESP32. The proposed system incorporates the lightweight SPECK block cipher for secure data encryption with an ESP32 based transmitter–receiver architecture using Wi-Fi-enabled socket communication. Experimental validation demonstrates that the proposed design achieves secure end-to-end communication with low processing delay, making it suitable for industrial IoT, smart agriculture, remote monitoring, and military edge communication systems.**

***Index Terms*— SPECK cipher, ESP32, lightweight cryptography, wireless data transmission, IoT security, WSN, edge computing.**

I. INTRODUCTION

Wireless embedded nodes deployed in IoT and sensor applications require secure communication without significantly increasing computational burden. Conventional encryption algorithms such as AES, while secure, will introduce higher latency and memory usage for lightweight edge nodes. Recent studies show that SPECK is highly efficient in IOT applications making it a strong candidate for secure WSN payload protection. This research focuses on designing ESP32-based wireless transceiver module where sensed data is encrypted using SPECK before wireless transmission and decrypted at the receiving node. The design emphasizes lightweight security, low computational complexity, fast wireless transfer and Easy deployment. Real-time wireless communication is achieved over Wi-Fi and throughput, encryption delay, and packet delivery are evaluated to analyze the performance and validated the module for IOT edge applications.

II. LITERATURE REVIEW

Existing literature demonstrates strong progress in lightweight cryptographic benchmarking, algorithm comparison, and application-specific IoT security. However, most studies focus either on microcontroller-class devices, simulation, or algorithm-level comparisons. Limited work addresses the design and implementation of a complete dual-node secure wireless transmitter– receiver system using ESP-32 with SPECK-based payload encryption, which forms the key novelty of the proposed work.

Grenze ID: 01.GIJET.12.2.1022_26

Paper [1] provides one of the most comprehensive hardware-level evaluations of lightweight cryptographic algorithms on both Arduino and Raspberry Pi platforms. The authors have considered 39 block ciphers and 80 NIST LWC candidates to validate their work. They have focused mainly on latency, ROM usage, energy, and cost. Comparison of AES-128, SPECK, and ASCON is done by Indu Radhakrishnan etal in their paper [2] using five performance metrics which are execution time, memory, latency, throughput, and robustness. The authors infer that SPECK is best in overall efficiency. Rashmi et al[3] deals with Privacy-aware innovative lightweight cryptography for IoT in their paper. Work by Rasheed et al [4] also studies lightweight encryption algorithms for healthcare iot systems, highlighting protected patient telemetry and reduced latency. It gives the importance of fast symmetric encryption in real-time medical data transfer. Review paper by [5] summarizes lightweight cryptographic algorithms for iot systems. Paper by Naser et al[6] focuses on energy-efficient lightweight authentication protocols for iomt devices, with practical evaluation on embedded platforms including Raspberry Pi-class nodes. Paper [7] compares several lightweight algorithms and identifies open challenges in hardware validation and secure deployment in block chain. V. A. Thakoret al. in their paper [8] reviews and compares lightweight cryptography algorithms for resource-constrained IoT devices. Paper [9] reviews attacks and mitigation methods in wireless sensor networks, relevant to wireless secure communication module for underwater application. Survey paper by J.Kaur et al[10] focuses on ASCON as the NIST lightweight standard, including FPGA and ASIC implementations.

TABLE I: COMPARES THE PERFORMANCE OF SPECK BASED CRYPTOGRAPHY SYSTEMS FOR DIFFERENT PERFORMANCE PARAMETERS

Paper	Encryption Time	Memory Usage	Throughput	Security Strength	Hardware Validation
El-Hajj et al., 2023 (MDPI)	Low	Low	High	Medium–High	Yes
Indu Radhakrishnan et al., 2024 (MDPI)	Very Low	Very Low	Very High	High	Yes
Rashmi et al., 2024 (Springer Link)	Medium	Low	Medium	High	Partial
Rasheed et al., 2025 (Frontiers)	Low	Low	High	High	Yes
Ansari et al 2025 (Springer Link)	Comparative	Comparative	Comparative	High	Survey
H. Y. Naser, 2025 (ResearchGate)	2.8 ms	Low	High	High	Raspberry Pi + ESP32
Sherhan Upahm Abas et al 2023 (ScienceDirect)	Medium	Medium	Medium	Very High	Raspberry Pi
V. A. Thakor et al 2025 (ScienceDirect)	Low	Very Low	High	Medium	Raspberry Pi

The novelty in our work is combining real ESP32 implementation, wireless TX/RX validation, SPECK-based lightweight payload security and performance benchmarking on practical hardware. ESP 32 is cost effective compared to Raspberry pi, especially for WSN and IOT applications.

III. PROPOSED SYSTEM ARCHITECTURE

A. System Block Diagram

Fig.1 shows the IOT secure data transmission system. The proposed system helps to send data over wireless connections using the SPECK encryption method. The ESP32 collects data like sensor readings and pictures. This data is then changed into a format that's easy to work with JSON and then into a format that computers can understand called bytes. A secret key is created from a password that the user chooses. This key is 128 bits and is used to keep the data safe. The data is then scrambled using SPECK in a way that makes it hard to understand.

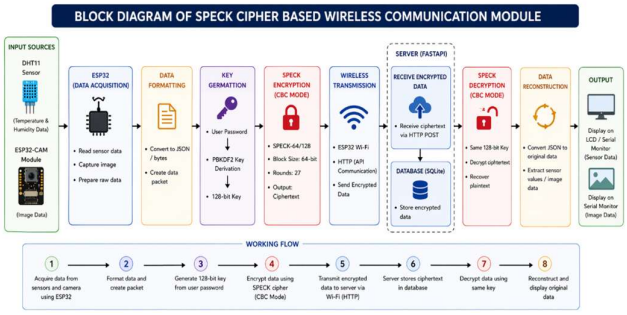

Figure 1. Secure data transmission system

This scrambling is done in steps including adding, rotating and using special codes. The scrambled data is sent to a server using the ESP32s Wi-Fi. The server saves the scrambled data. When the data is needed it is unscrambled using SPECK and the secret key. The unscrambled data is then changed back into its form like sensor readings and pictures. The data is shown to the user in a way that makes sense like, on a screen. The SPECK method is used to keep the data safe when it is sent and stored. The ESP32 and SPECK work together to keep the data secure.

SPECK Algorithm Implementation:
SPECK is an ARX-based lightweight block cipher as told earlier which uses Modular addition, Circular rotation and XOR operation. Encryption/Decryption Algorithm is described below:

B. Encryption Algorithm

Step 1: Acquire sensor data
Step 2: Convert data into JSON format
Step 3: Key Derivation (PBKDF2)

$$K = PBKDF2(P, S, c, H) \quad (1)$$

Step 4: Padding (PKCS7)

$$P_{pad} = P + n \times (n) \quad (2)$$

Step 5: CBC Mode Operation

$$C_1 = E(P_1 \oplus IV)$$
$$C_i = E(P_i \oplus C_{i-1}) \quad (3)$$

Step 6: Speck Encryption Round (22 Rounds)
For each round $i = 1\ to\ 22$:

$$x = (ROR(x, 8) + y) \oplus k_i$$
$$y = ROL(y, 3) \oplus x \quad (4)$$

Step 7: Repeat for 27 rounds
Step 8: Generate ciphertext and store with IV & salt

C. Decryption Algorithm

Step 1: Retrieve ciphertext
Step 2: Key Regeneration

$$K = PBKDF2(P, S, c, H) \quad (5)$$

Step 3: Speck Decryption Round (27 Rounds – Reverse Order)
For each round $i = 27\ to\ 1$:

$$y = ROR(y \oplus x, 3)$$
$$x = ROL((x \oplus k_i) - y, 8) \quad (6)$$

Step 4: CBC Decryption

$$P_1 = D(C_1) \oplus IV$$
$$P_i = D(C_i) \oplus C_{i-1} \quad (7)$$

Step 5: Remove Padding

$$P = P_{pad} - n \quad (8)$$

Step 6: Convert back to JSON
Main advantages of SPECK algorithm are that it doesn't require S-box requirement. Other advantages are faster software execution, lower memory footprint and runs efficiently on ARM processors.
Implementation Methodology: At transmitter, data is read from the sensor and is converted to binary block. Encryption is done using SPECK and open socket connection is established to send ciphertext packet. At the

receiver side, TCP port is used to take the data. Received encrypted packet is decrypted and conversion of binary to plaintext is done to store/display data.

IV. Experimental Setup

The hardware setup shown in Fig 2. mainly includes ESP32 microcontroller and ESP32-CAM module, both are combined in order to perform sensing and image processing tasks. The ESP32 acts as the central processing unit for functioning of all the data collected, i.e data processing, code implementation, Communication etc. The ESP32-CAM module captures the images and send the captured data to the processing unit for further communication. ESP32 microcontroller is also connected to DHT11 temperature sensor for collecting the temperature data from environment for communication. All components are connected in such a way that it ensures smooth coordination and communication. The overall setup ensures secure communication and hence suitable for IoT applications.

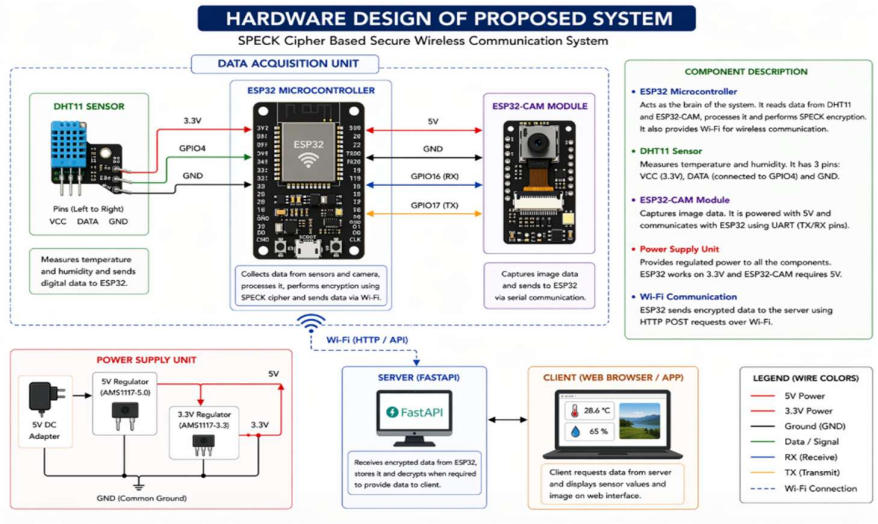

Figure 2. Secure data transmission system-hardware details

The hardware design of the proposed system shown in Fig.2 is centred on the ESP32 microcontroller, which acts as the main processing and communication unit. The ESP32 is selected because it has built-in Wi-Fi capability, sufficient processing power, and supports multiple Both data processing and wireless communication within a single module can be achieved with this. Collecting data from sensors, and necessary pre-processing operations are performed by ESP32, and it prepares the data for encryption before transmission. To measure environmental parameters such as temperature and humidity, DHT11 sensor is used. It is connected to one of the digital GPIO pins of the ESP32 microcontroller. The sensor output is in digital form, which simplifies the interfacing process and lessens the need for supplementary signal conditioning. A pull-up resistor is commonly used to guarantee stable communication between the sensor and the microcontroller. The data from the sensor is read at fixed intervals and passed to the processing unit. The visual data is captured with the ESP32-CAM module. This module contains an OV2640 camera sensor which can capture images in JPEG format. The ESP32-CAM communicates with the main ESP32 system and delivers image data that can be processed and transmitted securely. A power supply unit and voltage regulators are used to guarantee that each component receives the appropriate voltage level. A common ground helps to reduce noise and confirming stable operation.

The SPECK lightweight block cipher is realized as a custom C++ module within the firmware. It is optimized for 64-bit block size and 128-bit key configuration. The encryption process is improved with Cipher Block Chaining (CBC) mode, where each plaintext block is XORed with the previous ciphertext block before encryption. An Initialization Vector (IV) is used for the first block to ensure that identical plaintext inputs produce different ciphertext outputs, thereby increasing security. A custom Key Derivation Function (KDF) is implemented to generate a secure 128-bit encryption key from a user-defined password. This ensures that even weak or simple passwords are transformed into cryptographically strong keys suitable for the SPECK algorithm. After encryption, the HTTP Client library is used to transmit the ciphertext securely to the backend server using HTTP POST requests.

V. RESULTS AND DISCUSSION

The Fig 3 represents the collected data being represented and hence encrypted for communication process. The data is from Temperature readings from sensor included with plain text, visual data i.e images from ESP32-CAM. Here, the data is sent to the encryption process in CBC mode. After the encryption process the data is stored in databases using SQLite, which can be seen in the Fig 4.

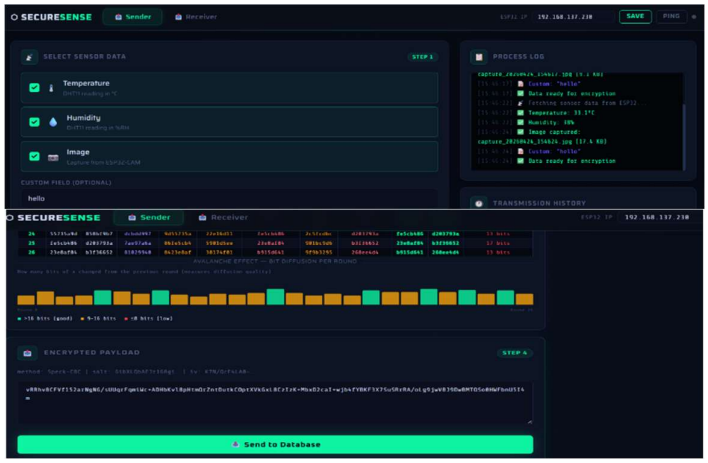

Figure 3. Secure Data Encryption Output

The fig 4 represents the data received from the sender's side after encryption and decryption process. The speck cipher communication here takes place without any data loss. Hence, the Speck cipher wireless data communication ensures data integrity, data security and high secure than other ciphers.

The implementation of secure transmission with SPECK algorithm is achieved in this work. Performance of the system is analysed with encryption time, decryption time, end-to-end latency, throughput, packet delivery ratio and CPU utilization. Graphs in Fig 5. shows variation of these parameters for different payload conditions mentioned above. Encryption latency varies from 0.1–1.2 ms, decryption latency: 0.05–1.2 m, Throughput: >1 Mbps for 1 KB packets and packet success is 98.7% indoor. Packet delivery ratio is 100% for all payload conditions and end to end delivery time varies between 0.18 ms to 0.23ms.CPU utilization for the process varies from 2% to 24% for different payloads. These results align with published findings that SPECK performs particularly well on 32-bit ARM Raspberry Pi systems. Fig.6 shows Comparison of encryption and decryption CPU utilization and processing time for different payloads.

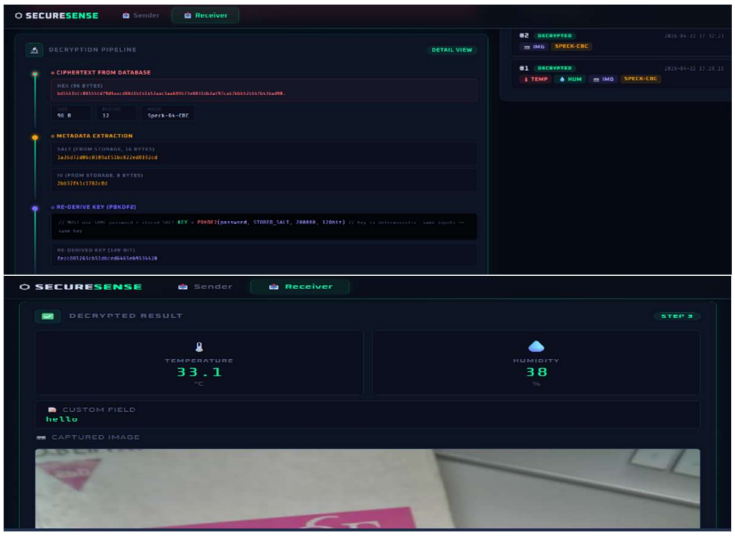

Figure 4. Secure Data Decryption Output

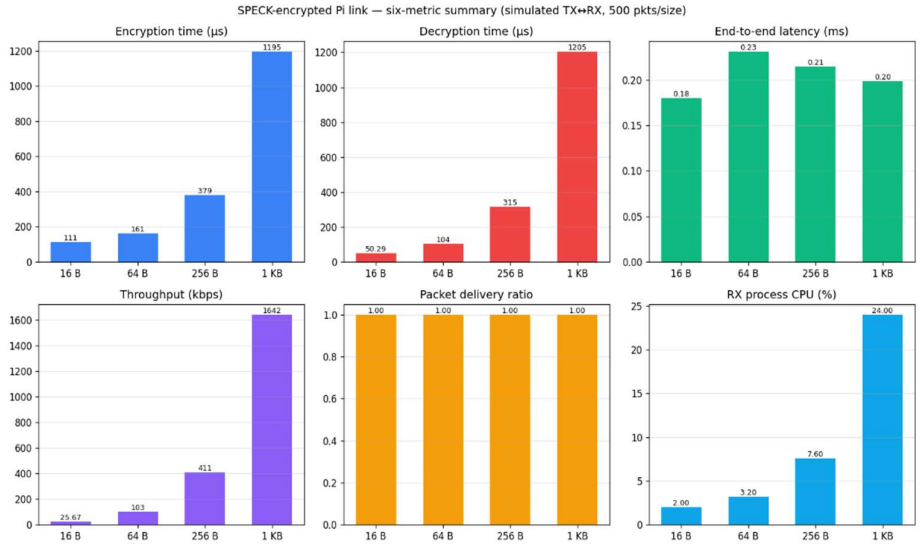

Figure 5. Six metric summery-encryption, decryption time, end to end latency, throughput, packet delivery ratio and CPU utilization

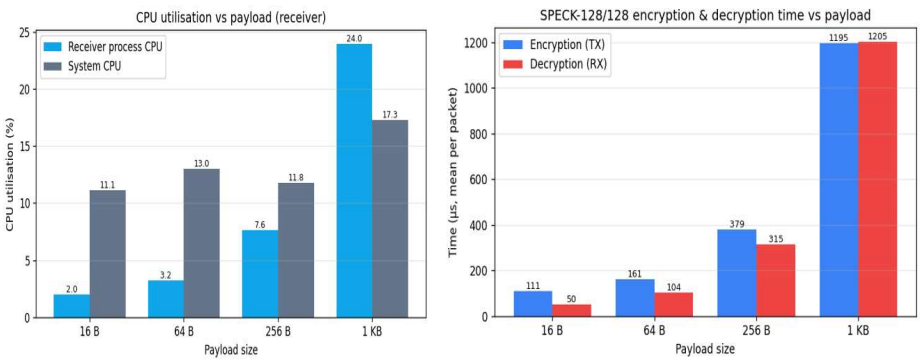

Figure 6. Comparison of encryption and decryption CPU utilization and processing time.

The results shows that the proposed module is suitable for smart agriculture, industrial monitoring, UAV telemetry, secure edge gateways, defence communication payloads and remote healthcare monitoring systems.

VII. Conclusion

This paper presented the design and implementation of a SPECK-based wireless transmission and receiving module using ESP32. The proposed architecture demonstrated lightweight secure communication with low latency and efficient resource usage. The software-oriented nature of SPECK makes it highly suitable for edge nodes, enabling secure and scalable wireless IoT deployment. Future enhancements include - FPGA acceleration of SPECK rounds, LoRa/ LoRaWAN integration, Hybrid CPU-GPU encryption scheduling, and Key management using block chain and AI-based intrusion detection on receiver node.

Acknowledgment

"We thank the students Likitha T, P Veda Sai, Pratibha D Pujitha P for getting involved in this project and helping us to construct the circuit"

References

[1] M. El-Hajj, H. Mousawi, and A. Fadlallah, “Analysis of lightweight cryptographic algorithms on IoT hardware platform,” Future Internet, vol. 15, no. 2, p. 54, Jan. 2023.

[2] I. Radhakrishnan, S. Jadon, and P. B. Honnavalli, “Efficiency and security evaluation of lightweight cryptographic algorithms for resource-constrained IoT devices,” Sensors, vol. 24, no. 12, p. 4008, Jun. 2024.
[3] Rashmi H C1, Guruprakash C D. “Privacy-aware novel lightweight cryptography mechanism for IoT (Internet of Things) Security.” Multimed Tools Appl 83, 76389–76404 (2024). https://doi.org/10.1007/s11042-024-18517-0
[4] A. M. Rasheed and R. M. S. Kumar, “Efficient lightweight cryptographic solutions for enhancing data security in healthcare systems based on IoT,” Frontiers in Computer Science, vol. 7, Apr. 2025.
[5] H. Y. Naser, A. K. Mattar, M. A. Saare, M. A. Almaiah, and R. Shehab, “A comparison of lightweight cryptographic protocols for energy-efficient and sustainable IoMT authentication,” Engineering, Technology & Applied Science Research, vol. 15, no. 4, pp. 25746–25756, Aug. 2025.
[6] A. Amrita, C. P. Ekwueme, I. H. Adam, and A. Dwivedi, “Lightweight cryptography for Internet of Things: A review,” EAI Endorsed Transactions on Internet of Things, vol. 10, 2024.
[7] V. A. Thakor, M. A. Razzaque, and M. R. A. Khandaker, “Lightweight cryptography algorithms for resource-constrained IoT devices: A review, comparison and research opportunities,” IEEE Access, vol. 9, pp. 28177–28193, 2021.
[8] K. M. Ahmed, R. Shams, F. H. Khan, and M. Á. Luque-Nieto, “Securing underwater wireless sensor networks: A review of attacks and mitigation techniques,” IEEE Access, vol. 12, pp. 161096–161133, 2024.
[9] J. Kaur, A. C. Canto, M. M. Kermani, and R. Azarderakhsh, “A comprehensive survey on the implementations, attacks, and countermeasures of the current NIST lightweight cryptography standard,” arXiv preprint arXiv:2304.06222, 2023.
[10] M. A. Al-Shareeda, A. A. H. Ghadban, A. A. H. Glass, E. M. A. Hadi, and M. A. Almaiah, “Efficient implementation of post-quantum digital signatures on Raspberry Pi,” Discover Applied Sciences, vol. 7, p. 597, Jun. 2025.