Throughput Optimization of the ASCON Lightweight Cryptographic Algorithm on IoT Devices

DOI: https://doi.org/10.54654/isj.v3i26.1172

Chu Thi Ngoc Quynh*, Nguyen Thanh Ngoc, Le Thi Ninh, Pham Thi Thuy An

***Abstract*— ASCON is a lightweight authenticated encryption algorithm standardized by NIST for securing constrained IoT environments. This study presents the implementation of ASCON-128a in a practical IoT architecture, where the ESP32 serves as the central microcontroller at the IoT node and the Raspberry Pi 4 acts as the central microcontroller at the IoT gateway. The ESP32 is chosen for its low-power operation, integrated wireless communication, and suitability for embedded edge processing. At the same time, the Raspberry Pi 4 is selected to support higher computational demands and data aggregation at the network edge. To improve performance, optimization strategies such as CPU frequency scaling, IRAM execution, dual-core parallelism, and cache warm-up were applied on the ESP32, alongside frequency stabilization and cache warm-up on the Raspberry Pi 4. Experimental results indicate that the ESP32 achieves an encryption throughput of 2.42 MB/s, while the Raspberry Pi 4 reaches 124 MB/s. These results validate ASCON-128a as an efficient and secure cryptographic solution for heterogeneous IoT systems.**

***Tóm tắt*— ASCON là một thuật toán mã hóa hạng nhẹ được NIST chuẩn hóa để bảo mật trong môi trường IoT hạn chế tài nguyên. Nghiên cứu này, trình bày việc triển khai ASCON-128a trong một kiến trúc IoT thực tế, trong đó ESP32 đóng vai trò là bộ vi điều khiển trung tâm tại IoT node và Raspberry Pi 4 là bộ vi điều khiển trung tâm tại IoT gateway. ESP32 được chọn nhờ khả năng hoạt động tiết kiệm năng lượng, tích hợp giao tiếp không dây và phù hợp cho xử lý biên nhúng. Trong khi đó, Raspberry Pi 4 được chọn để đáp ứng nhu cầu tính toán cao hơn và tổng hợp dữ liệu tại biên mạng. Để cải thiện hiệu suất, các chiến lược tối ưu hóa như điều chỉnh tần số CPU, thực thi IRAM, song song hóa lõi kép và làm nóng bộ nhớ đệm đã được áp dụng trên ESP32, cùng với ổn định tần số và làm nóng bộ nhớ đệm trên Raspberry Pi 4. Kết quả thực nghiệm cho thấy ESP32 đạt thông lượng mã hóa 2,42 MB/s, trong khi Raspberry Pi 4 đạt 124 MB/s. Những kết quả này, xác nhận ASCON-128a là một giải pháp mã hóa hiệu quả và an toàn cho các hệ thống IoT không đồng nhất**.

***Keywords*— IoT, ASCON, AEAD, lightweight cryptography.**

***Từ khóa*— IoT, ASCON, AEAD, mật mã hạng nhẹ.**

I. INTRODUCTION

Among lightweight cryptographic algorithms, ASCON distinguishes itself through its permutation-based design, offering support for authenticated encryption with associated data (AEAD) and cryptographic hashing. ASCON delivers superior performance on microcontroller platforms while maintaining robust data security. This enables Internet of Things (IoT) systems to achieve effective data protection while adhering to constraints such as energy efficiency, minimal hardware costs, and sustained operational performance in practical deployments. Recent findings from the ASCON development team [1] indicate that the algorithm's performance on 32-bit microcontrollers, such as the ARM Cortex-M, is highly dependent on optimization strategies. In baseline configurations-lacking IRAM utilization, loop unrolling, or execution from Flash-throughput is typically limited to approximately 1 MB/s. On 32-bit microcontrollers like the ARM Cortex-M3 or

This manuscript was received on October 26, 2025. It was reviewed on December 15, 2025, revised on December 23, 2025 and accepted on December 30, 2025.

* Corresponding author

ESP32, performance is more favorable but remains constrained to around 1 MB/s under default settings [2]. Notably, a recent study by Cagua et al. implemented ASCON on both the CupCarbon simulator and the Raspberry Pi ZeroW, a less powerful platform compared to the Raspberry Pi 4 [3]. Their results demonstrated an average processing time of 79.26 ms for a 1KB data packet with low standard deviation, meeting near-real-time requirements for IoT applications. However, their evaluation focused solely on execution time on the Raspberry Pi 4, without assessing throughput. Building on these insights, this study addresses the following objectives:

- Implementation of ASCON-128a within an IoT hardware architecture, comprising an IoT Node employing the ESP32 for sensor data encryption and an IoT Gateway utilizing the Raspberry Pi 4 for data decryption and re-encryption.

- Evaluation of the algorithm's execution time and throughput within the designed IoT system.

The paper is structured as follows: Section II outlines the proposed methodology, Section III presents the experimental results, and the concluding section summarizes the findings.

II. Proposed Method

A. Overview of the ASCON Lightweight Cryptographic Algorithm

ASCON is a family of authenticated encryption algorithms denoted as $\text{Ascon}_{a,b,-k,-r}$. Each member of the family is parameterized by key length k (≤ 128 bits), bit rate r, and the number of internal rounds a and b. Each design specifies an authenticated encryption algorithm $\mathcal{E}_{a,b,-k,-r}$ and a corresponding decryption algorithm $\mathcal{D}_{a,b,-k,-r}$. The primary variant used in this study is ASCON-128a [4]. ASCON-128a utilizes a 128-bit key (K), a 128-bit nonce, variable-length associated data (AD), and variable-length plaintext. It operates in two main modes: encryption and decryption, as illustrated in Figure 1.

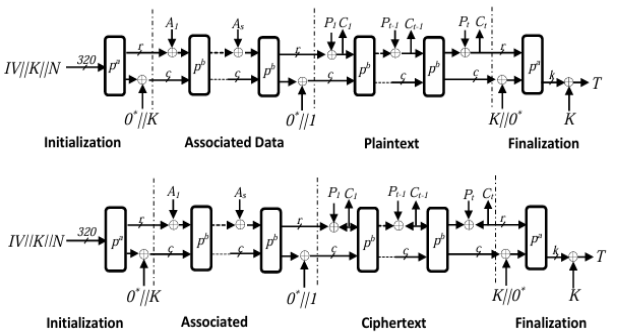

Figure 1. Encryption and Decryption Mechanism of ASCON-128a

B. IoT Hardware Model Design

The proposed IoT hardware model consists of two main components: the IoT Gateway kit and the IoT Node kit [5]. The system architecture is illustrated in Figure 2**:**

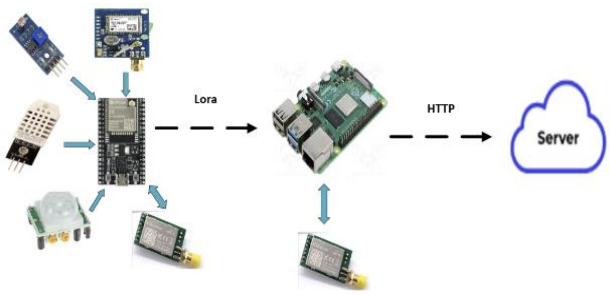

Figure 2. Experimental ASCON-based model for secure communication in IoT systems

- Sensor Data Block: Responsible for collecting environmental data through various sensors and forwarding it to the IoT Node.

- IoT Node Block: The core microcontroller is an ESP32**,** which receives input from the sensor block including motion sensors, temperature and humidity sensors, light sensors, and GPS positioning. It then encrypts the data using the ASCON-128a algorithm and transmits it to the central node via LoRa communication**.**

- IoT Gateway Block: Built on a Raspberry Pi 4 microcontroller, this block identifies data from different sensor nodes, analyzes and processes the data, and performs decryption of the received data using the ASCON-128a algorithm. It also uses LoRa communication to interface with the IoT Node. After decryption, the data is re-encrypted using ASCON-128a before being uploaded to the server to enhance security.

C. Implementation of the ASCON Algorithm on IoT Devices

The objective of this study is to implement the ASCON-128a algorithm on the hardware model described in Section B, which includes two main components: the IoT Node and the IoT Gateway. The performance of the ASCON algorithm on this hardware platform is then evaluated based on metrics such as processing time and throughput**.**

- Algorithm Flow on the IoT Node: The ESP32 collects environmental data from sensors and the GPS module. This data is then normalized into a string format. Each parameter collected from the sensors is represented as a 1-byte value. The data is encrypted using the ASCON-128a algorithm with a 128-bit symmetric key (shared between both nodes) and a randomly generated 128-bit nonce. The algorithm flow is illustrated in Figure 3**.**

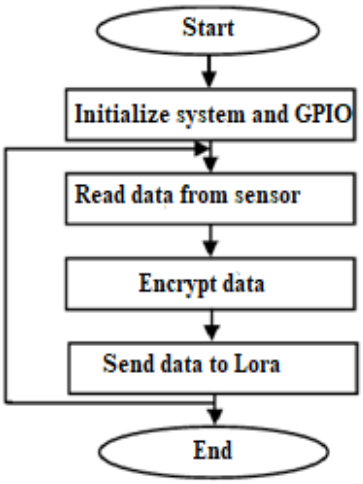

Figure 3. Algorithm Flowchart of the IoT Node Block

- Algorithm Flowchart for the IoT Gateway:

Encrypted data is received via the UART port through LoRa communication. The Raspberry Pi 4 uses the same ASCON-128a algorithm to perform decryption. The key is pre-shared, and the nonce is derived from the last 128 bits of the received data stream. After successful decryption, the data is re-encrypted using a different key and nonce to add an additional layer of security before it leaves the internal system. Finally, the data is transmitted to a web server for remote system monitoring. The algorithm flow is illustrated in Figure 4**.**

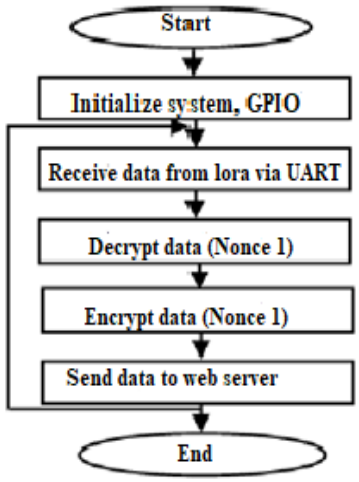

Figure 4. Algorithm flowchart of the IoT gateway block

D. Throughput Optimization Strategies on ESP32 and Raspberry Pi

a. Throughput Optimization on ESP32

- Utilizing High-Speed IRAM: IRAM (Internal RAM) is the on-chip RAM integrated directly within the microcontroller. On the ESP32, this memory region offers higher speed and lower latency compared to PSRAM (external RAM) [6]. Programs executed from IRAM run directly from RAM instead of flash. The ASCON-128a algorithm frequently accesses memory through `load/store` functions (reading/writing a 320-bit state) and during permutation operations. IRAM typically has a latency of 1–2 CPU cycles, while PSRAM incurs 10–20 cycles, resulting in significantly faster memory access. For example, accessing 12 bytes in PSRAM may require up to 120 cycles, whereas the same operation in IRAM takes only 12–16 cycles up to 6× faster. Running the program from IRAM also reduces cache misses and improves throughput by approximately 2 to 3 times compared to flash memory, which has higher latency. In this implementation, memory for message, ciphertext, and decrypted buffers is allocated using the *heap_caps_malloc()* function with the flags *MALLOC_CAP_8BIT* | *MALLOC_CAP_INTERNAL* to ensure storage in IRAM. The *benchmark_core()* function is also

marked with the *IRAM_ATTR* attribute to guarantee execution from IRAM.

- Utilizing Multi-Core Processing

The ESP32 features two Xtensa LX6 cores, each capable of independently processing data [7]. When encryption and decryption operations are executed in parallel across both cores, throughput is nearly doubled. A mutex is used to safely synchronize access to *core_stats*, preventing race conditions and ensuring high performance, as each access is short in duration. The code creates two tasks that run *benchmark_core()* on core 0 and core 1 respectively, using the *xTaskCreatePinnedToCore()* function. This design ensures that both cores execute cryptographic operations concurrently and independently, without sharing input or output data. Since each encryption task processes a separate message buffer, no data dependency exists between the two cores, enabling embarrassingly parallel execution and near-linear throughput scaling. Although the encryption tasks operate on independent data buffers, certain global variables such as execution time counters, cycle statistics, and aggregated throughput results are shared across tasks for performance measurement and reporting. To prevent race conditions and ensure measurement correctness, a mutex-based synchronization mechanism is employed. A FreeRTOS mutex *(SemaphoreHandle_t)* is created during system initialization using *xSemaphoreCreateMutex()*. Each task performs the cryptographic computation independently and only enters a critical section when updating shared statistical variables. Access to these shared resources is protected by acquiring the mutex with *xSemaphoreTake()* and releasing it with *xSemaphoreGive()*. The critical sections are intentionally kept short and lightweight, containing only simple arithmetic updates and counter increments. As a result, mutex contention is minimal, and the synchronization overhead has negligible impact on the overall cryptographic throughput. This design choice balances correctness and performance, ensuring accurate benchmarking without undermining the benefits of parallel execution.

- Warm-up Cache and Multiple Iterations:

CPU often use cache memory to speed up data access. During the first run of an algorithm, if the data is not yet in the cache, the CPU must fetch it from RAM, resulting in slower cache access. Warming up ensures that the key and nonce are already in the L1 cache. The method of repeating multiple times and averaging smooths out system noise, ensuring accurate throughput measurements. The code performs one encryption/decryption run before benchmarking to warm up the cache, then measures the average over 10 iterations

- Program Optimization:

The permutation (P), LOAD, and STORE functions are marked static inline to avoid function call overhead. Rotation operations use macros (ROT, EXT_BYTE64, INS_BYTE64): Inlining replaces function calls with direct code, reducing overhead (~10-20 cycles/call). With hundreds of P/LOAD calls in encryption, this saves significantly. The ROT macro performs bit rotation directly in code, avoiding helper functions, saving ~2-5 cycles per rotation. ASCON has 10 rotations/round, with 12/6 rounds, saving ~120-300 cycles per block.

b. Throughput Optimization on Raspberry Pi

The IoT Gateway functions as the receiver of encrypted data packets from IoT nodes. The ciphertext is transmitted from the ESP32 via LoRa communication to the Raspberry Pi 4, where it is decrypted and then re-encrypted before being sent to the server. The Raspberry Pi 4 is an embedded computer based on a Cortex-A72 core architecture operating at a frequency of 1.5 GHz. According to reference [8], the benchmark performance of ASCON-128a reaches 7 cycles per byte (c/B). However, when implemented in a real system with linked data communication, the actual throughput decreases significantly. Therefore, this study applies overclocking and warm-up mechanisms on the Raspberry Pi 4 to improve the execution efficiency of the algorithm in terms of processing time and throughput.

During the benchmarking of the ASCON-128a encryption and decryption algorithm on the

Raspberry Pi 4, the measured execution time can be affected by several factors such as an unfilled cache state (cold cache) during the first execution, CPU frequency scaling or automatic frequency adjustment, and interference from the operating system. Therefore, a warm-up step is required to "preheat" the L1 and L2 caches, stabilize the CPU temperature and frequency, and improve the accuracy of throughput and execution time measurements. The warm-up implementation on the Raspberry Pi 4 consists of the following steps: Before conducting the official benchmark, the encryption and decryption functions are executed several times to warm up the cache. This process helps load both code and data into the cache, thereby increasing the likelihood of cache hits during subsequent benchmark runs. The second step involves pinning the process to a fixed CPU core to ensure that it always runs on the same core, avoiding interference from unwanted background loads. Additionally, the CPU is configured to operate at its maximum frequency, preventing any dynamic frequency scaling during benchmarking. To achieve the highest performance and stable clock speed on the Raspberry Pi 4, an overclocking mechanism is applied as described in [9].

- Implementation of the ASCON-128a algorithm: The Raspberry Pi 4 receives the ciphertext from the ESP via LoRa communication, then performs the following steps:

- Step 1: Decrypt using the ASCON-128 algorithm with KEY/AD to obtain the true plaintext.

- Step 2: Re-encrypt the plaintext with NEW_KEY/NEW_AD before sending it to the server.

- Step 3: Measure throughput across two phases: decryption and re-encryption.

Throughput measurement is performed only once on the first valid packet and does not affect the subsequent processing pipeline. The measurement technique applies the following factors: stable operating frequency, warm-up to heat the cache/branch predictor, fixed size using the actual data just decrypted, and padding to a size of 1024 bytes**.** Random noise (RNG) elimination: for decryption, generate ct||tag once before measuring; the measurement loop only calls *crypto_aead_decrypt()*. For re-encryption, generate a random base_nonce and then increment the nonce each round - no RNG calls inside the measurement loop - while still ensuring nonce uniqueness.

F. Benchmarking Methodology and Message Size Considerations

In the proposed IoT system, the size of the encrypted data is closely tied to the functional role of each component in the architecture. On the IoT node based on the ESP32, the ASCON-128a algorithm is applied directly to realistic sensor data frames with a small payload size of 12 bytes, accurately reflecting the characteristics of periodic IoT sensor data. Consequently, performance measurements on the ESP32 focus on processing latency and throughput for short messages, rather than large data blocks.

In contrast, at the IoT gateway based on the Raspberry Pi 4, decrypted data is typically aggregated, processed, or re-packaged before being forwarded to the server. Therefore, the performance of ASCON-128a on the Raspberry Pi 4 is evaluated using 1 KB data blocks, which are representative of aggregated payloads commonly encountered in practical gateway-level processing.

While this difference in message size reflects realistic deployment scenarios, it also makes direct throughput comparison between the two platforms challenging. For this reason, in this paper, throughput is analyzed separately for each platform and is not used for direct numerical comparison between the ESP32 and the Raspberry Pi 4. The primary objective of the benchmarking is to evaluate the effectiveness of ASCON-128a within each system role, rather than to compare absolute performance across heterogeneous hardware architectures.

In addition, to mitigate the impact of fixed overheads (e.g., initialization costs) when processing short messages, all measurements are repeated multiple times and averaged. A cache warm-up mechanism, as described in Section

II.D, is also applied to improve measurement stability and reproducibility.

III. Experimental Results

A. Implementation Results on ESP32

With various methods applied to increase the algorithm's throughput on the ESP32, the ASCON-128a algorithm was implemented on both cores of the ESP32 operating at a CPU frequency of 240 MHz. The data frame has a length of 12 bytes, containing the collected sensor data as shown in Figure 5:

LDR Data (1byte)	MOVE Data (1byte)	DHT-Humidity Data (1byte)	DHT-Temperature Data (1byte)	Latitude Data (4 byte)	Longitude Data (4 byte)

Figure 5. Encrypted data size on ESP32

The results of data encryption throughput are shown in Table 1.

Table 1. Encryption execution time and throughput results on ESP32

Core / Throughput	Core 0	Core 1	Total
Time (µs)	7883	7883	15766
Throughput (MB/s)	1.21	1.21	2.42
Cycle/byte (C/B)	189.43	189.17	378.6

Table 2. ASCON execution results on ARM Cortex-M3 and ESP32 platforms with a data size of 16 bytes [10]

Algorithm	ARM Cortex-M3 (84MHz)		ESP32 (240MHz)-1 core	
	Encrypt (MB/s)	Decrypt (MB/s)	Encrypt (MB/s)	Decrypt (MB/s)
ASCON-128a	1.86	1.7	0.92	0.93
ASCON-128	1.54	1.44	0.86	0.66
ASCON-80pq	1.52	1.43	0.84	0.61

Table 3. Comparison of ASCON-128 execution results with the AES-GCM and ChaCha20-Poly1305 algorithms

Algorithm	Encypt (MB/s)	Decrypt (MB/s)
ASCON-128a	2.42	2.4
AES-GCM [11]	1.25	1.245
Chacha20-Poly1350 [11]	3.076	3.086

From Tables 1 and 2, it is evident that the measured parameters of the study on the ESP32 platform are optimized due to the mechanisms described in Section D. Table 3 compares the encryption and decryption performance of ASCON-128a with AES-GCM and ChaCha20-Poly1305. The results show that ChaCha20-Poly1305 achieves the highest encryption/decryption speed. However, in terms of balancing security and execution speed, ASCON-128a is more suitable for deployment on IoT devices.

With an achieved throughput of 2.42 MB/s on the ESP while the system uses the LoRa communication protocol-which supports only tens of Kbps-there is a substantial mismatch. This can cause data queue congestion if proper buffering mechanisms are not in place. Therefore, before transmission, an optimized in-RAM queue management strategy is required. Additionally, data compression or bundling multiple records into a single, reasonably sized packet should be applied.

B. Implementation Results on Raspberry Pi4

The decryption process successfully recovered the original plaintext at the IoT node. Subsequently, the data was re-encrypted before being sent to the server. Table 4 below presents the results of the study conducted on the Raspberry Pi 4 and compares them with the Raspberry Pi Zero W, as referenced in [3].

TABLE 4. ENCRYPTION AND DECRYPTION RESULTS ON THE RASPBERRY PI ZERO W AND RASPBERRY PI 4 PLATFORMS

Hardware	Time		Throughput	
	Encrypt (1KB)	Decrypt (1KB)	Encrypt(MB/s	Decrypt (MB/s)
Rasberry Pi zero W	79.26ms	79.27ms	12.6	12.7
Rasberry Pi4	7.91µs	7.85µs	122.2	124.33

The results demonstrate that ASCON-128a on the Raspberry Pi 4 outperforms the Raspberry Pi Zero W by approximately 103 times in encryption/decryption time and roughly 10 times in throughput.

IV. CONCLUSION

This study achieved its main objective of implementing and evaluating the performance of the lightweight authenticated encryption algorithm ASCON-128a in a practical IoT system composed of an ESP32-based IoT node and a Raspberry Pi 4-based IoT gateway. The results confirm that ASCON-128a can be efficiently deployed in heterogeneous IoT environments when appropriate implementation-level optimizations are applied.

Several optimization techniques, including IRAM utilization, cache warm-up, and dual-core parallel execution on the ESP32, as well as cache warm-up and CPU frequency stabilization on the Raspberry Pi 4, significantly improved throughput, achieving 2.42 MB/**s** on the ESP32 for realistic 12-byte sensor data and up to 124 MB/**s** on the Raspberry Pi 4 for 1 KB data blocks. These findings demonstrate the effectiveness of the proposed methods for both constrained IoT nodes and high-throughput edge gateways. The applied optimization strategies offer clear benefits in reducing memory latency and improving execution efficiency; however, their effectiveness is platform-dependent and may increase system complexity. Moreover, the current evaluation focuses primarily on throughput, while energy consumption and power efficiency are not directly measured.

From a security perspective, the implementation leverages the AEAD properties of ASCON-128a, but a comprehensive system-level security analysis remains limited. Future work will therefore focus on strengthening security and safety aspect**s**, including formal threat modeling, side-channel resistance analysis, robust nonce management, and evaluation of gateway trust assumptions. In addition, energy measurements, extension to platforms such as RISC-V and FPGA [12, 13], and integration with IoT protocols like MQTT or CoAP will be pursued to further validate the practicality and security of ASCON-128a in real-world deployments.

REFERENCES

[1] C. Dobraunig, M. Eichlseder, F. Mendel, M. Schläffer, "Ascon Status Update (Finalist)", *NIST LWC Workshop*, 2021.

[2] A. Sarasa Laborda et al., "Study About the Performance of Ascon in Arduino Devices, Applied Sciences", *MDPI*, 2025.

[3] G. Cagua, V. Gauthier-Umaña, C. Lozano-Garzon, "Implementation and Performance of *Lightweight Authentication Encryption ASCON on IoT Devices"*, IEEE Access, 2025.

[4] NIST, "NIST Special Publication 800-232: Recommendation for the Ascon Family of Lightweight Cryptography Algorithms", *National Institute of Standards and Technology*, Gaithersburg, MD, 2023. [Online]. Available: https://csrc.nist.gov/pubs/sp/800/232/final.

[5] P. Q. Tri, N. N. Son và C. V. Kien, "Thiết kế mô hình thí nghiệm IoT ứng dụng trong giảng dạy bậc đại học", *Tạp chí khoa học và công nghệ,* số 45A, 2020.

[6] Espressif, "memory types", 2025, Access time:14/9/2025, . https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-guides/memory-types.html.

[7] Espressif, "FreeRTOS overview", 2025, Access time: 14/9/2025 . https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/system/freertos.html.

[8] Ascon, "Software reference implementations", 2025, Access time: 10/9/2025, Ascon – Implementations.

[9] S. Arjun, "How to safely overclock Raspberry Pi 4", 2025, Access time: 10/9/2025, How to Safely

Overclock Raspberry Pi 4 in 2022 [Guide] | Beebom.

[10] Rweather, "Lightweight cryptography primitives", 2021, Access 10/9/2025, Lightweight Cryptography Primitives: Performance on 32-bit platforms.

[11] Oryx-embedded, "Crypto benchmark on ESP32 MCU", 2023, Access 25/9/2025. https://www.oryx-embedded.com/benchmark/espressif/crypto-esp32.html.

[12] Ky.P. V and Phuc. L. H, "A Secure connection management solution for IPSEC on FPGA", *Journal of Science and Technology on Imfomation Security*, no 13, vol 1, pp 3-11, 1-2022, DOI: https://doi.org/10.54654/isj.v1i13.142.

[13] Huy. T. Q, Bac. D. T, Trinh. B. D, Linh. L. T. K, Hao. H. L. H and D. Phan. D. P, "Proposed optimized, hardware, implementation, for the S-Box of the PRESENT algorithm using combinational logic circuits", *Journal of Science and Technology on Infomation Security*, no 3, vol 23, 43-52, 12-2024, DOI: https://doi.org/10.54654/isj.v3i23.1070.

ABOUT THE AUTHOR

Chu Thi Ngoc Quynh

Workplace: Academy of Cryptography Techniques, Vietnam Government Information Security Commission.

Email: ctnquynh@gmail.com

Education: She received her master-degree in Telecommunications Engineering from Posts and Telecommunications Institute of Technology, Vietnam in 2012.

Recent research direction: His research interests include digital signal processing, telecommunications, IoT.

Tên tác giả: **Chu Thị Ngọc Quỳnh**

Cơ quan công tác: Học viện kỹ thuật mật mã, Ban Cơ yếu Chính phủ, Việt Nam.

Email: ctnquynh@gmail.com

Quá trình đào tạo: Tốt nghiệp thạc sỹ ngành kỹ thuật điện tử năm 2012, tại trường Học viện công nghệ bưu chính viễn thông.

Hướng nghiên cứu hiện nay: xử lý tín hiệu số, viễn thông, IoT.

Nguyen Thanh Ngoc

Workplace: Academy of Cryptography Techniques, Vietnam Government Information Security Commission.

Email: thanhngoc8818@gmail.com

Education: She received her master-degree in Telecommunications Engineering from Posts and Telecommunications Institute of Technology, Vietnam in 2015.

Recent research direction: Digital electronics, FPGA design, digital signal processing.

Tên tác giả: **Nguyễn Thanh Ngọc**

Đơn vị công tác: Học viện kỹ thuật mật mã, Ban Cơ yếu Chính phủ, Việt Nam.

Email: thanhngoc8818@gmail.com

Quá trình đào tạo: Tốt nghiệp thạc sỹ ngành kỹ thuật điện tử tại Học viện công nghệ bưu chính viễn thông năm 2015, hiện đang là nghiên cứu sinh ngành kỹ thuật mật mã tại Học viện kỹ thuật mật mã.

Hướng nghiên cứu hiện nay: Điện tử số, thiết kế FPGA, xử lý tín hiệu số.

Le Thi Ninh

Workplace: Academy of Cryptography Techniques, Vietnam Government Information Security Commission.

Email: Dt050223@actvn.edu.vn

Education: She is a student at the Academy of Cryptographic Technology.

Recent research direction: His research interests include digital electronics, FPGA design, and digital signal processing.

Tên tác giả: **Lê Thị Ninh**

Cơ quan công tác: Học viện kỹ thuật mật mã , Ban Cơ yếu Chính phủ, Việt Nam.

Email: Dt050223@actvn.edu.vn

Quá trình đào tạo: Sinh viên năm thứ 5 Học viện kỹ thuật mật mã.

Hướng nghiên cứu hiện nay: Điện tử số, thiết kế FPGA, xử lý tín hiệu số.

Pham Thi Thuy An

Workplace: Academy of Cryptography Techniques, Vietnam Government Information Security Commission.

Email: anphamdtvtil.com

Education: She received her master-degree in Military Technical Academy,Vietnam in 2015.

Recent research direction: His research interests include digital electronics, IoT, and digital signal processing.

Tên tác giả: **Phạm Thị Thúy An**

Cơ quan công tác: Học viện kỹ thuật mật mã, Ban Cơ yếu Chính phủ, Việt Nam.

Email: anphamdtvt@gmail.com

Quá trình đạo tạo: Tốt nghiệp thạc sỹ ngành Kỹ thuật điện tử tại Học viện Kỹ thuật quân sự năm 2015.

Hướng nghiên cứu hiện nay: Điện tử số, IoT và xử lý tín hiệu số.