Secure Federated Intrusion Detection for Resource-Constrained IoT Devices Using Lightweight Cryptography: A Hardware-Validated Study

Yerlan Tursynbek , Nurtay Albanbay * , Djamel Djenouri , Shahid Latif , Ainur Akhmediyarova , Zhibek Alibiyeva , Janna Alimkulova , Dina Oralbekova

Posted Date: 1 May 2026

doi: 10.20944/preprints202605.0029.v1

Keywords: federated learning (FL); internet of things (IoT); intrusion detection; ESP32; edge computing; secure model update; embedded systems; data integrity

 doi:10.20944/preprints202605.0029.v1

Article

Secure Federated Intrusion Detection for Resource-Constrained IoT Devices Using Lightweight Cryptography: A Hardware-Validated Study

Yerlan Tursynbek [1], Nurtay Albanbay [1,*], Djamel Djenouri [2], Shahid Latif [2], Ainur Akhmediyarova [1], Zhibek Alibiyeva [1], Janna Alimkulova [3] and Dina Oralbekova [4]

[1] Institute of Automation and Information Technologies, Satbayev University, Satbayev str.22, Almaty 050013, Kazakhstan

[2] School of Computing and Creative Technology, University of the West of England, Bristol, United Kingdom

[3] Turan University, Department of Computer Engineering, Almaty, Kazakhstan

[4] Institute of Information and Computational Technologies, Almaty, Kazakhstan

* Correspondence: n.albanbay@satbayev.university

Abstract

Federated learning (FL) enables distributed model training in IoT environments while keeping raw data on local devices. However, protecting model-update exchange is difficult on microcontroller-class devices due to strict latency, memory, and energy constraints. Existing studies often evaluate lightweight cryptography outside complete FL pipelines or on more powerful hardware, leaving its practical overhead on MCU-class devices insufficiently explored. This paper presents an end-to-end, hardware-validated secure framework for exchanging model updates in federated learning on resource-constrained IoT microcontrollers. Implemented on ESP32-based edge devices, the framework combines lightweight block ciphers (SPECK, SIMON, and PRESENT), HMAC-SHA256 for integrity verification, and ECDH-HKDF for session-key establishment. The evaluation assessed latency, throughput, RAM/ROM footprint, and energy consumption. Results show that SPECK provides the lowest overhead (0.13 μs/byte, 8.68 MB/s, 138.3 mJ), SIMON offers intermediate performance (0.41 μs/byte, 1.96 MB/s, 184.9 mJ), and PRESENT incurs the highest computational cost (89.37 μs/byte, 0.011 MB/s, 446.2 mJ). In the CICIoT2023 federated intrusion-detection evaluation, the secure model maintained stable convergence and achieved 85.43% accuracy after 20 rounds, remaining close to the centralized baseline. These findings demonstrate the practical feasibility of secure model-update exchange in FL on real IoT microcontrollers and provide hardware-grounded guidance for cipher selection under tight resource budgets.

Keywords: federated learning (FL); internet of things (IoT); intrusion detection; ESP32; edge computing; secure model update; embedded systems; data integrity

1. Introduction

The Internet of Things (IoT) has become a key part of modern digital infrastructure, supporting data collection and automation in healthcare, industry, smart buildings, energy systems, and transportation. Many IoT devices operate at the network edge and continuously generate sensitive data, including personal information and operational telemetry [1]. At the same time, edge devices are often built on resource-constrained microcontrollers with limited memory, limited computational power, and strict energy budgets. These constraints make it difficult to apply heavyweight security mechanisms while maintaining reliable real-time operation.

Federated learning (FL) is widely adopted as a privacy-aware learning paradigm for IoT environments, where raw data remain on edge devices and only model updates are exchanged with the global server. This design reduces direct data exposure and enables collaborative training in

 doi:10.20944/preprints202605.0029.v1

decentralized settings. However, the communication stage remains a critical attack surface. During transmission, model updates may be intercepted, allowing adversaries to infer sensitive information about local training data from gradients or parameters, even though raw data never leave devices. In addition, updates sent over insecure channels may be tampered with in transit before reaching the server, degrading global model integrity upon aggregation. As illustrated in Figure 1, each client performs local training and transmits its update to the global server; if an update is altered during transmission, the manipulated information can be incorporated into the aggregation and then propagated to clients in the next round. Therefore, securing model updates in transit is essential for trustworthy FL deployment in IoT systems. This issue is particularly important in federated intrusion-detection settings, where altered or intercepted updates may reduce the reliability of attack detection in IoT monitoring systems.

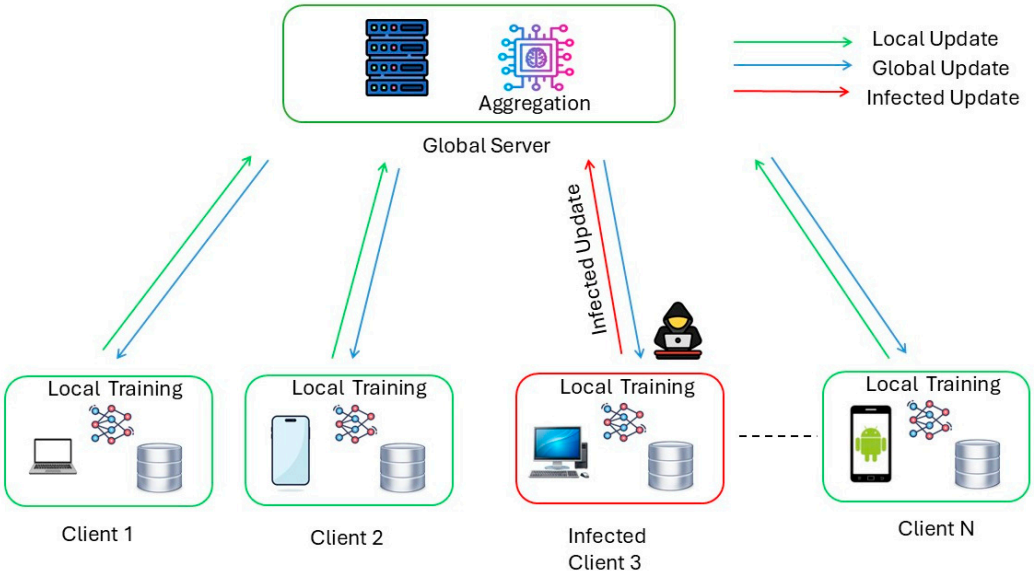


Figure 1. Federated learning communication and aggregation flow.

A common direction is to use cryptographic protection during aggregation (e.g., secure aggregation) so that the server cannot inspect individual client updates [2]. While such methods provide strong privacy guarantees in principle, they typically incur non-trivial overhead when applied repeatedly over many FL rounds. This overhead can be especially problematic for microcontroller-class clients that must combine local training, networking, and security operations within tight RAM and energy limits. However, for MCU-class IoT devices, there remains a need for lighter communication-level protection mechanisms whose practical overhead can be validated under real embedded deployment conditions.

For this reason, lightweight cryptography is an attractive alternative for constrained IoT devices. Lightweight block ciphers are designed to reduce code size, memory footprint, and computation cost, while still providing confidentiality for transmitted data [3]. Although these ciphers have been studied for embedded communication, their behavior within an iterative FL workflow remains poorly understood. In FL, encryption is performed at each communication round, and the resulting overhead interacts with local training time, wireless transmission latency, and overall system stability. Therefore, system-level evidence is needed to guide cipher selection for real IoT deployments. This need is especially relevant for intrusion-detection applications, where secure and reliable update exchange is required to preserve practical monitoring capability on resource-constrained IoT nodes. Most prior studies evaluate lightweight cryptography either in isolation from federated learning or without hardware-level validation on MCU-class devices, leaving the practical cost of secure model-update exchange under real embedded constraints insufficiently explored.

 doi:10.20944/preprints202605.0029.v1

To address this gap, this paper presents an end-to-end, hardware-validated framework for secure model-update exchange in a federated-learning-based intrusion-detection workflow on ESP32-class IoT microcontrollers. In the proposed workflow, each client performs on-device training, encrypts the resulting model updates using lightweight block ciphers and applies HMAC-based integrity protection before transmitting the protected updates to the server for aggregation. The study focuses on quantifying the practical cost of secure update exchange under real hardware constraints.

The main contributions of this work can be summarized as follows:

- An end-to-end secure model-update exchange framework for federated learning is developed for resource-constrained ESP32-based IoT devices.
- The framework jointly integrates lightweight block-cipher encryption and HMAC-based integrity protection to secure model updates during transmission.
- The proposed workflow is fully implemented on real ESP32 microcontrollers, combining on-device training, protected update transmission, and wireless communication within a single embedded pipeline.
- A comprehensive hardware-level evaluation quantifies latency, throughput, memory footprint, and energy consumption, enabling joint analysis of security overhead and practical deployment efficiency.

The remainder of this paper is organized as follows. Section 2 reviews related work on federated learning for IoT and lightweight cryptography for resource-constrained devices. Section 3 presents the proposed methodology, including the system architecture and the secure model-update exchange protocol. Section 4 describes the experimental setup and evaluation metrics. Section 5 reports the experimental results. Section 6 discusses the findings, trade-offs, and limitations. Section 7 concludes the paper.

2. Related Work

The accelerated adoption of federated learning in IoT environments has provided substantial benefits, including decentralized data analytics and enhanced privacy protection. Nevertheless, this change introduces critical challenges, including susceptibility to inference attacks and data poisoning. To mitigate these issues, recent research has explored integrating cryptographic mechanisms, such as homomorphic encryption, secure multiparty computation, and differential privacy, into federated learning frameworks. However, traditional cryptographic solutions are often computationally prohibitive on resource-constrained IoT hardware, prompting growing interest in lightweight cryptographic alternatives tailored for edge environments.

2.1. Federated Learning for IoT Environment

In recent years, FL has become a central paradigm for IoT, as it enables model training without sharing data. Nguyen et al. [4] demonstrated the practical viability of a self-learning FL-IDS on more than 30 devices (95.6% detection rate, ~257 ms latency, 0% false positives), although mainly in SOHO scenarios. Ferrag et al. [5] and Campos et al. [6] further showed that, for IoT-IDS, FL performance is highly sensitive to non-IID data distributions and aggregator choice (FedAvg/Fed+), while their findings remain largely evaluative.

To improve performance under non-IID conditions, more advanced schemes have been proposed. Danquah et al. [7] achieved 99.93% on Mirai with reduced computational complexity, but evaluated only a narrow attack class. Sun et al. [8] improved minority-attack detection on CICIoT2023 and increased poisoning resilience, albeit with additional server-side overhead. Rey et al. [9] showed that FL can approach centralized learning on N-BaIoT, yet simple averaging remains vulnerable to adversarial clients. Liu et al. [10] improved non-IID accuracy/efficiency via prototype-based transfer, although the method is sensitive to prototype quality. Latif et al. [11] incorporated recursive self-distillation into an FL-based malware detection framework for the Internet of Vehicles, demonstrating improved generalization under non-IID conditions, reduced communication

 doi:10.20944/preprints202605.0029.v1

overhead through iterative model compression, and mitigation of catastrophic forgetting across FL rounds.

The privacy/security track strengthens FL through cryptography and verifiable protocols. Niu et al. [12] proposed lightweight verifiable aggregation (secret sharing + blinding), reducing overhead but not eliminating cryptographic costs. Jin et al. [13] accelerated HE-based FL through selective encryption; however, HE remains computationally expensive for resource-constrained IoT nodes. Jalali and Chen [14] improved channel security and training privacy, but large-scale validation remains limited. Huang et al. [15] reduced authentication cost, though their model depends on a trusted authority.

From a practical standpoint, Campolo et al. [16] and dos Santos et al. [17] highlighted the importance of middleware/protocol design (MQTT, OMA LwM2M, Smart Campus) for real-world FL-IoT deployment, though without focusing on peak IDS performance. Zakariyya et al. [18] and Albanbay et al. [19] confirmed the core trade-off between resource efficiency and accuracy: more complex models generally yield higher detection performance. In contrast, lightweight models provide a better accuracy–cost balance on the edge platform.

2.2. Lightweight Cryptography in Resource-Constrained IoT Devices

For resource-constrained IoT devices, the main research stream focuses on designing and experimentally validating lightweight ciphers on MCUs. Mouad and Salim [20] on ESP8266/ESP32 reported higher speed and throughput than AES-128-ECB and Speck-128-ECB, with low memory usage, but with limited portability to 8/16-bit platforms. Ibrahim and Agbinya [21] proposed an SPN-based cipher with strong execution-time/throughput results and baseline resistance to linear/differential cryptanalysis, although additional validation against advanced attacks is still required. Nagesh et al. [22], evaluated on ATmega328/FPGA, achieved competitive randomness, timing, and power results, while noting the need for broader validation. Witwit et al. [23] reported acceleration over SPECK/SIMON/PRESENT on ESP32; however, evaluation was limited to a narrow baseline set and a single hardware platform. Wulandari et al. [24] implemented PRESENT on ATmega328P for LoRa and confirmed correctness with an acceptable memory footprint, but without a deep field-level, attack-oriented assessment. Băneasă et al. [25] presented a reduced-cost AES-128 implementation for constrained IoT with lower memory/computation requirements. At the same time, Ni et al [26] demonstrated a functional Feistel-like cipher in an end-to-end pipeline with reported execution around 3ms (4 rounds), albeit with limited cross-algorithm benchmarking.

A separate direction explores alternative lightweight primitives for application-specific use cases. Clemente-Lopez et al. [27] proposed chaos-based encryption for wearable healthcare, showing real-time feasibility in controlled settings. Jiteurtragool et al. [28] developed a Parabola-chaotic keyed-hash + SRAM-PUF scheme for ESP32 authentication with strong statistical properties, but without covering IDS/FL tasks. Ulla et al. [29] investigated STROBE/libdisco + PUF/ESP32, yet without fully standardized quantitative evaluation. Bansod [30] introduced an ultra-lightweight Feistel cipher with a low GE threshold, though with limited modern device-level validation.

In the FL context, lightweight cryptography is used to improve privacy and robustness. Chen et al.[31], Fan et al. [32], and Awan et al. [33] showed that combining lightweight encryption, reputation mechanisms, and communication optimization can reduce communication/computation costs but increases protocol complexity. Latif and Djenouri [34] proposed a hierarchical two-tier secure FL framework employing X25519 key exchange, ChaCha20-Poly1305 authenticated encryption, and HKDF-BLAKE2b key derivation for edge-cloud environments, achieving up to 90% reduction in cloud communication overhead while maintaining competitive model accuracy and 100% resistance to model poisoning. Zhang et al. [35] and Wang et al. [36] enhanced PPFL against poisoning/non-IID effects through encrypted filtering/selection; results are promising but largely validated on simulations and computer-vision datasets. Hu et al. [37] applied compressive sensing as lightweight encryption+compression and achieved communication-cost reductions with acceptable accuracy, although transferability to IoT-IDS scenarios requires further validation.

 doi:10.20944/preprints202605.0029.v1

As summarized in Table 1, most existing lightweight cryptography studies evaluate ciphers in isolation on a single hardware platform, without integration into end-to-end FL pipelines or integrity verification of transmitted model updates. Notably, none of the reviewed works combine real MCU-class hardware deployment with both confidentiality and integrity protection within a complete federated learning workflow.

Table 1. Comparison of secure federated learning approaches for IoT environments.

Reference	Security Mechanism	Hardware	Integrity	Key Management	MCU-Class Hardware	FL Evaluation
Jin et al.[13]	HE (CKKS) + selective encryption	CPU/GPU (Intel i7 + Tesla T4)	✗	Centralized Key Authority	✗	✓
Jalali & Chen[14]	CKKS + iTLS	Python simulation	Partial	Trusted Authority	✗	✓
Niu et al.[12]	LSSS + Blinding + CRT	GPU (RTX 4060)	Partial	Diffie–Hellman key agreement	✗	✓
Wei et al.[38]	Mask reuse + DP	Real IoT devices (unspecified)	✗	Secret sharing	✗	✓
Chen et al.[31]	Lightweight enc. + reputation	Simulated	✗	Pre-shared	✗	Partial
Zhang et al. [35]	Sym. HE (SHE) + anomaly detection	Desktop/GPU (i7 + RTX 4060)	✓	Trusted Authority (TA)	✗	✓
Ours	SPECK/SIMON/PRESENT + HMAC-SHA256	ESP32 (MCU)	✓	ECDH + HKDF	✓	✓

Survey studies by Amrita et al. [39] and Gunathilake et al. [40], Soto-Cruz et al. [41] and comparative analysis by Chinbat et al. [42] converge on the same conclusion: lightweight cryptography is essential for constrained IoT, but there is no universally best algorithm—selection is always platform-, workload-, threat-model-, and overhead-dependent.

2.3. Research Gap

Existing research on securing FL in IoT environments generally follows two distinct directions: either computationally intensive cryptographic mechanisms are applied, which are unsuitable for microcontroller-class devices, or lightweight ciphers are evaluated in isolation from a complete FL pipeline. As a result, there is a lack of end-to-end, hardware-validated analysis of how combined confidentiality and integrity protection affects system-level performance, such as latency, energy consumption, and memory usage, on real IoT devices. This gap highlights the need to integrate lightweight cryptography directly into the full federated learning workflow and evaluate it under realistic MCU constraints. To address this gap, the present study develops and experimentally validates an end-to-end secure model-update exchange framework on real ESP32-class IoT devices, with a joint evaluation of cryptographic overhead and learning utility under embedded-resource constraints.

3. Proposed Methodology

A secure federated intrusion detection framework for IoT environments was developed to enable collaborative model training without exposing raw local traffic data. The framework was designed for resource-constrained edge devices and implemented with lightweight cryptographic protection to secure the exchange of model updates.

 doi:10.20944/preprints202605.0029.v1

The proposed workflow consists of five stages. First, an IoT traffic dataset was selected and preprocessed through cleaning, normalization, and feature preparation. Second, client-side IDS models were initialized across the participating nodes. Third, local training was conducted independently for each client using private data partitions. Fourth, the resulting model updates were protected through lightweight cryptographic processing before transmission. Fifth, the protected updates were incorporated into the federated aggregation cycle to iteratively refine the global model across communication rounds.

To reflect practical deployment constraints, the framework was executed on ESP32-based nodes. Communication rounds were repeated according to the predefined training configuration, while convergence behavior was evaluated using accuracy and loss across rounds. Security and efficiency were jointly evaluated to analyze the trade-off between cryptographic protection and system overhead. The evaluation included per-byte latency, end-to-end operation time, memory usage (RAM/ROM), and energy consumption measured during encryption and decryption.

By combining federated training with lightweight cryptographic safeguards, the framework was designed to preserve data locality, strengthen the confidentiality and integrity of updates, and maintain deployability in low-power IoT scenarios.

3.1. System Architecture and Components

The proposed secure federated learning architecture comprises a global server and multiple microcontroller-based IoT clients (Client 1 … Client N), as illustrated in Figure 2. The global server is responsible for round coordination, secure update handling, parameter aggregation, and redistribution of the updated global model. Each client performs local IDS training on private traffic data and participates in iterative model refinement without sharing raw data.

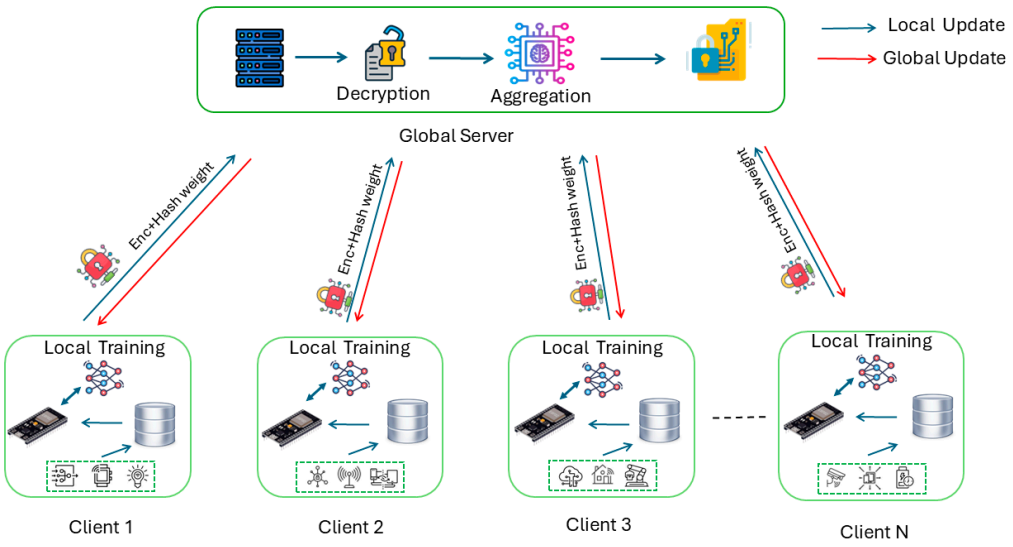


Figure 2. Secure FL architecture with encrypted and hash-verified local/global model updates.

In each communication round, the global server is responsible for distributing models, securely collecting model updates and aggregating them. In contrast, the client nodes perform local training on private datasets and participate in protected model-update exchange. Secure processing is applied in both communication directions, namely in the uplink transmission of local updates and in the downlink redistribution of the updated global model. In this way, the architecture integrates federated learning functionality with bidirectional communication protection while preserving data locality and practical deployability on constrained IoT hardware.

From a functional perspective, each client node includes four modules: local data storage, local training engine, cryptographic processing unit (hashing/encryption/decryption), and communication interface. The server-side includes: a federated round controller, a secure update validation unit, and

 doi:10.20944/preprints202605.0029.v1

an aggregation engine. Overall, the architecture provides confidentiality and integrity of exchanged model parameters while preserving data locality and maintaining practical deployability on constrained IoT hardware.

The detailed message flow, key-establishment procedure, and per-round security operations are described in Section 3.2.

3.2. A Secure Transmission Protocol for Federated Learning Model Updates on ESP32 Devices

A classical centralized federated learning architecture is considered, consisting of a single aggregating server and multiple client devices. The server S possesses sufficient computational resources and coordinates the training process by initiating FL rounds, distributing the current global model parameters, and aggregating the received updates. The clients C_i are resource-constrained ESP32-based nodes that store local data and perform on-device training without transmitting raw data to the server.

Figure 3 illustrates the operations performed during a single federated learning round, including local training, secure update transmission, integrity verification, and global model aggregation. Each round t consists of two phases: (1) the server distributes the current model parameters w_t along with round metadata to the clients; (2) each client C_i performs local training on its dataset D_i for E epochs and generates a model update $\Delta w_i^{(t)}$, which is then transmitted to the server S. Upon receiving a set of updates, the server performs aggregation using the FedAvg algorithm and produces an updated global model w_{t+1}.

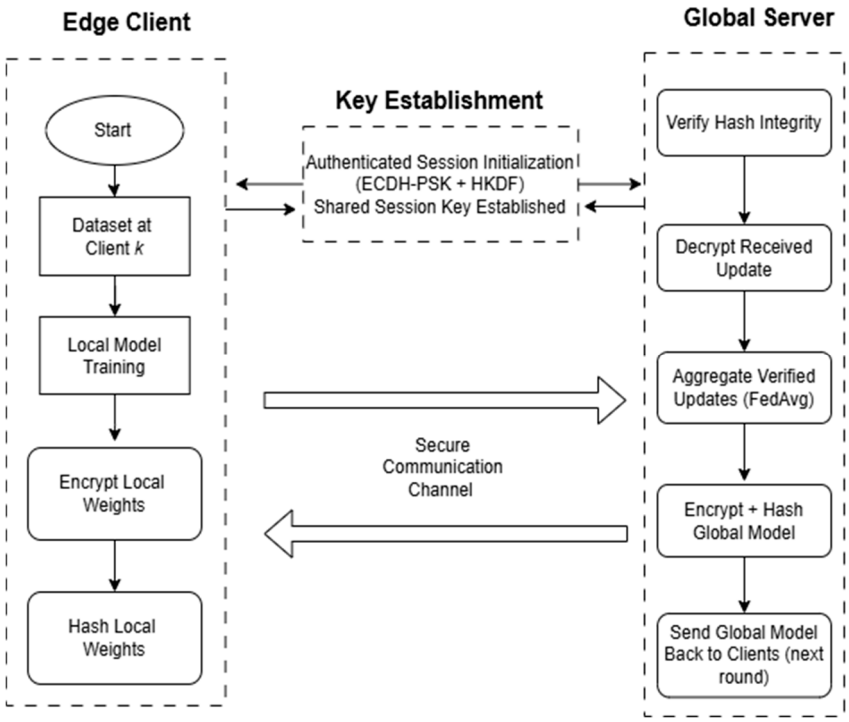


Figure 3. Secure federated learning workflow in a single communication round.

The proposed protocol is designed to ensure the confidentiality and integrity of model update exchanges in federated learning when transmitted over potentially insecure communication channels. In particular, it is required that, even if network traffic is intercepted, an adversary cannot reconstruct the content of the local update $\Delta w_i^{(t)}$ or extract any sensitive information from it. At the same time, the server must be able to unambiguously verify that the received message has not been modified during transmission and was indeed generated by the corresponding client C_i. Furthermore, the protocol must prevent replay attacks, in which a previously intercepted update is resent in a different round and mistakenly accepted by the server as valid.

 doi:10.20944/preprints202605.0029.v1

It is assumed that the adversary can eavesdrop on the communication channel and attempt to modify or substitute messages (eavesdropping/MITM threat model), but does not have physical access to the device to extract secrets (i.e., key material is assumed to be securely stored on the client side). Attacks aimed at compromising the correctness of the learning process by malicious clients (e.g., poisoning or Byzantine attacks, gradient manipulation) as well as robust aggregation techniques are considered a separate research direction and are beyond the scope of this section.

Device Registration and Session Key Establishment (ECDH + PSK)

Before exchanging model updates in federated learning, a device registration and session key establishment procedure is performed. This procedure ensures that the session is securely bound to a specific ESP32-based client and protects the communication against eavesdropping and message substitution attacks (MITM). The protocol is implemented as a lightweight handshake that combines: (i) elliptic curve–based key exchange to derive a shared secret, and (ii) device authentication based on a pre-shared master key and a unique device identifier known to the server.

On the client side, cryptographically secure random values are generated using an entropy source and a CTR-DRBG (CSPRNG). These random values form the *client_random* parameter and are used to generate an ephemeral ECDH key pair on the P-256 curve (secp256r1). The client's public key PK_C is serialized into a network-compatible format using elliptic curve point encoding/decoding procedures (ECP point encode/decode), ensuring reliable transmission of key material over the communication channel under microcontroller resource constraints.

For device registration, a pre-authentication mechanism is employed using a unique device identifier, DEVICE_ID, and a master key, MASTER_KEY, known to the server. On the client side, a device-specific derived key is computed as follows:

$$PSK_{device} = HMAC(MASTER_{KEY}, DEVICE_{ID}) \quad (1)$$

This derived key serves as a *root of trust* for authenticating the handshake process. The client sends a message to the server containing *DEVICE_ID*, *client_random*, and PK_C, along with an HMAC tag computed over the handshake data using PSK_{device}. The server performs the same derivation of PSK_{device} based on the *MASTER_KEY* and the received *DEVICE_ID*, verifies the HMAC tag, and thereby confirms that a registered device generated the message and has not been tampered with during transmission.

After successful initial verification, the server generates its own random parameter *server_random* and an ephemeral ECDH key pair on the P-256 curve. The server serializes its public key PK_S and sends it to the client along with *server_random* and an HMAC tag computed over the accumulated handshake transcript. During the exchange, both parties incrementally build a *transcript* (i.e., the sequence of exchanged handshake messages and fields) and compute its hash as follows:

$$transcript\_hash = SHA256(handshake\_transcript), \quad (2)$$

This mechanism binds the subsequent key derivation process to a specific session and its parameters (including *client_random, server_random,* PK_C, PK_S, and auxiliary fields). The shared secret is then computed using ECDH as:

$$Z = ECDH(SK_C, PK_S) = ECDH(SK_S, PK_C), \quad (3)$$

which is subsequently converted into a byte representation (MPI/bignum → bytes) for use in the key derivation function.

The final step involves deriving working session keys using HKDF-SHA256, where the shared secret Z serves as the input keying material, and the salt/context incorporates the *transcript_hash* and session-specific random parameters:

$$K\_session = HKDF\_SHA256(Z, salt, info), \quad (4)$$

where K_session denotes the session key material derived from HKDF-SHA256. The derived keying material is then deterministically split into purpose-specific keys, such as K^{enc} for encryption and K^{mac} for integrity protection.

To mutually confirm that both parties have derived identical keying material and that the handshake has not been tampered with, the final *ServerFinished* and *ClientFinished* messages are

 doi:10.20944/preprints202605.0029.v1

exchanged. These messages are computed as HMAC values over the *transcript_hash* using keys derived from HKDF. Successful verification of the *Finished* messages indicates that both the client and the server have agreed on a shared secret bound to the specific session and can securely proceed to protect application-layer messages symmetrically.

After the handshake is completed, all subsequent communication within the FL process is performed exclusively using symmetric cryptographic primitives (lightweight encryption and integrity protection), which is critical for ESP32 devices with limited computational resources, memory, and energy consumption.

Following the registration and handshake procedure (ECDH + mutual verification of *Finished* messages), the client C_i and the server S obtain a shared session keying material *KEY* (the output of HKDF), bound to the specific session via *transcript_hash* and session-specific random parameters. At this stage, asymmetric cryptography is no longer used; further protection of FL communication is achieved solely through symmetric primitives. From *KEY*, operational keys are deterministically derived, including at least an encryption key K^{enc} and an integrity key K^{mac} (e.g., via partitioning of the HKDF output or additional HKDF-Expand steps with distinct context labels).

In each round t, the client performs local training on dataset D_i for E epochs and produces an update $\Delta w_i^{(t)}$. The update is then transformed into a payload $P_i^{(t)}$ via parameter serialization. Since updates are of variable length and transmitted over the network, a lightweight block cipher (e.g., SPECK, SIMON, or PRESENT) is employed in CTR mode, which supports encryption of arbitrary-length serialized model updates using a nonce/IV. For each message, a unique per-message parameter $N_i^{(t)}$(nonce/IV) is generated using a CSPRNG (CTR-DRBG with entropy input), after which the ciphertext is computed as:

$$C_i^t = Enc_{K^{enc}}(P_i^t, N_i^t) \tag{5}$$

The use of $N_i^{(t)}$ensures encryption uniqueness and prevents ciphertext repetition for identical data, while also enabling efficient encryption of large messages without increasing memory requirements.

To protect against message substitution and ensure authenticity, an integrity tag is additionally computed using a keyed hash function (HMAC). The tag is calculated not only over the ciphertext but also over critical header fields, including the client identifier and the round number. This design enhances resistance to MITM attacks and prevents the reuse of intercepted messages across different rounds:

$$Tag_i^t = HMAC_{K^{mac}}(ID_i \parallel t \parallel N_i^t \parallel C_i^t) \tag{6}$$

Thus, in each round, the client transmits a protected application message of the form$\{ID_i, t, N_i^t, C_i^t, Tag_i^t\}$ .The protocol overhead mainly consists of the nonce and HMAC tag (along with a few header bytes) and is typically significantly smaller than the size of the model updates. Therefore, it does not impose a substantial burden on network transmission while ensuring verifiable integrity and confidentiality.

On the server side, incoming messages are processed in a strict order to minimize the risk of incorporating invalid data into the training process. First, message *freshness* is verified by checking the expected round number t and detecting potential replay (e.g., using a monotonic round policy and/or enforcing uniqueness of $N_i^{(t)}$ within a defined window). Next, the server computes the expected tag $\widehat{Tag}_i^{(t)}$ and compares it with the received $Tag_i^{(t)}$; in case of mismatch, the message is immediately discarded as tampered or corrupted. Only after successful integrity verification does the server proceed with decryption of $C_i^{(t)}$ using K^{enc} and $N_i^{(t)}$, reconstruction of the payload $P_i^{(t)}$, and decoding of the update $\Delta w_i^{(t)}$. The recovered updates are then used in the aggregation procedure (FedAvg) to produce the updated global model w_{t+1}.

The proposed scheme enables secure exchange of FL updates on ESP32 devices without requiring computationally expensive asymmetric operations in each round, thereby allowing a quantitative evaluation of the "cost of security" on resource-constrained microcontrollers. In subsequent sections, the implementation overhead of SPECK, SIMON, and PRESENT is analyzed

 doi:10.20944/preprints202605.0029.v1

with respect to execution time, throughput, energy consumption, and memory usage, and the impact of secure communication on the robustness and performance of federated learning is examined.

From a practical perspective, the protected client message in each round t follows a fixed structure $\{ID_i, t, N_i^t, C_i^t, Tag_i^t\}$. The fields ID_i and t uniquely identify the sender and the round context, $N_i^{(t)}$ ensures encryption uniqueness, $C_i^{(t)}$ contains the encrypted payload (serialized update), and $Tag_i^{(t)}$ guarantees integrity and authenticity. When using HMAC-SHA256, the tag length is 32 bytes, while the overall overhead per message—primarily determined by the nonce/IV, tag, and a few header bytes—typically amounts to only tens of bytes. This is significantly smaller than the size of the transmitted model updates and does not result in a substantial increase in network traffic.

To mitigate replay attacks, the server enforces freshness verification of incoming messages. First, the round number t is validated: updates are accepted only for the current expected round (or within a strictly limited window in the presence of network delays). Second, the server ensures the uniqueness of $N_i^{(t)}$ for each client within an active session (or, at a minimum, maintains the last accepted t for each ID_i and rejects messages with lower or equal values). This combination effectively prevents the reuse of previously intercepted packets and the acceptance of outdated updates during aggregation.

The key material is generated once during the handshake phase and subsequently used as a session key until session termination or key renewal. Rekeying is recommended upon device restart, session timeout, or after a predefined number of FL rounds to limit the amount of data protected under a single key set, thereby improving long-term security. Within the defined threat model, the protocol ensures confidentiality and integrity of updates transmitted over the communication channel and protects against message substitution and replay attacks.

4. Experimental Setup and Evaluation Metrics

4.1. Experimental Setup

The experimental evaluation was conducted on a practical edge-computing testbed designed for secure federated learning in IoT environments. Table 2 summarizes the hardware specifications of the devices used in the experiments. ESP32-S3-N16R8V boards served as distributed FL client nodes responsible for local training, cryptographic processing, and wireless transmission. A Raspberry Pi 4 Model B was used as the aggregation server for round coordination, integrity verification, decryption, and FedAvg aggregation. The hardware specifications of the client and server platforms are summarized in Table 2.

Table 2. Hardware specifications of experimental devices.

Parameter	ESP32-S3-N16R8V (Client)	Raspberry Pi 4B (Server)
Processor	Xtensa LX7, dual-core	BCM2711, quad-core Cortex-A72
Clock frequency	240 MHz	1.5 GHz
Architecture	32-bit	64-bit ARM
SRAM	512 KB	—
PSRAM	8 MB	—
System RAM	—	4 GB LPDDR4
Flash storage	16 MB	32 GB microSD
Wireless	Wi-Fi 802.11 b/g/n	Wi-Fi 802.11ac
Supply voltage	3.3 V logic (via USB 5 V)	5 V USB-C

To quantify the overhead introduced by secure model-update exchange, a separate hardware profiling setup was employed using the NI ELVIS II+ instrumentation platform together with a

 doi:10.20944/preprints202605.0029.v1

LabVIEW-based acquisition interface. During cryptographic execution and protected transmission of model updates, synchronized voltage and current signals were continuously recorded from the ESP32 platform. These measurements were used to derive instantaneous power and total energy consumption associated with the evaluated cryptographic operations. In parallel, execution time and memory-related indicators were collected directly from the ESP32 device in order to assess latency, throughput, and resource overhead under the proposed secure FL workflow.

As shown in Figure 4, the ESP32 board was mounted on the NI ELVIS II+ prototyping platform and connected to the corresponding measurement terminals, enabling controlled power supply and synchronized acquisition of electrical signals during runtime.

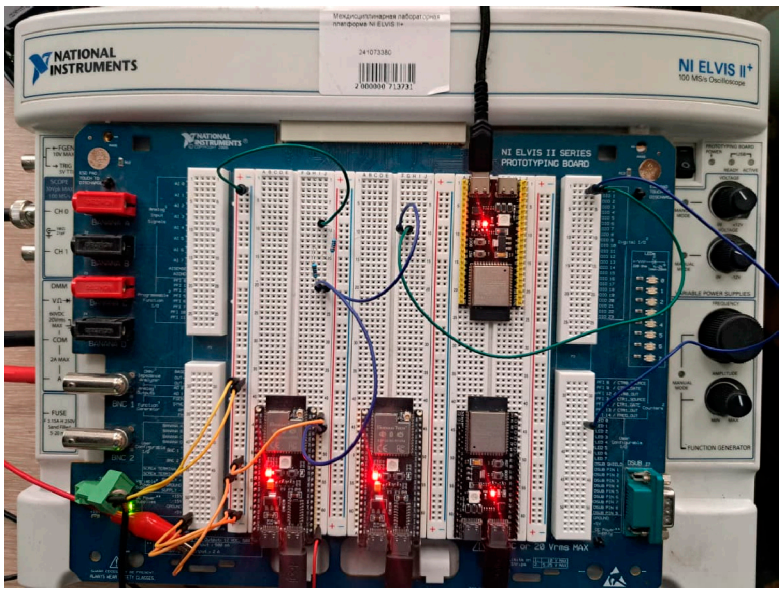


Figure 4. Experimental hardware setup based on the NI ELVIS II+ platform with ESP32 device connection for power and energy measurements.

Figure 5 presents the LabVIEW-based monitoring and control interface used to configure acquisition parameters and record voltage–current traces in real time. This measurement environment enabled high-resolution, repeatable profiling of the electrical and computational behavior of the ESP32-based client nodes.

Overall, the experimental setup reflects a hardware-executed client-side FL workflow on ESP32 devices, including local training, update generation, cryptographic protection, and communication. At the same time, the Raspberry Pi server coordinates aggregation and secure update processing. This design supports a system-level assessment of the trade-off between communication security and deployment efficiency in practical IoT federated learning scenarios.

To clarify the end-to-end implementation and support reproducibility, the complete firmware and server-side code used in this study are publicly available at: https://github.com/SU-developer-team/esp32-federated-learning. The repository includes ESP32-S3 firmware for three federated clients, centralized baseline firmware, the Python federated server used to receive, verify, decrypt, aggregate, and log model updates, the ECDH-PSK authentication library, lightweight cipher benchmark firmware, PlatformIO build configuration, and LabVIEW files used for power and energy measurements. The training firmware uses embedded dataset partitions stored in header files and is built for the ESP32-S3-N16R8V board with PSRAM enabled. Therefore, the reported experiments are supported by publicly available implementation resources covering local training, secure model-update exchange, server-side aggregation, and hardware-level measurement.

 doi:10.20944/preprints202605.0029.v1

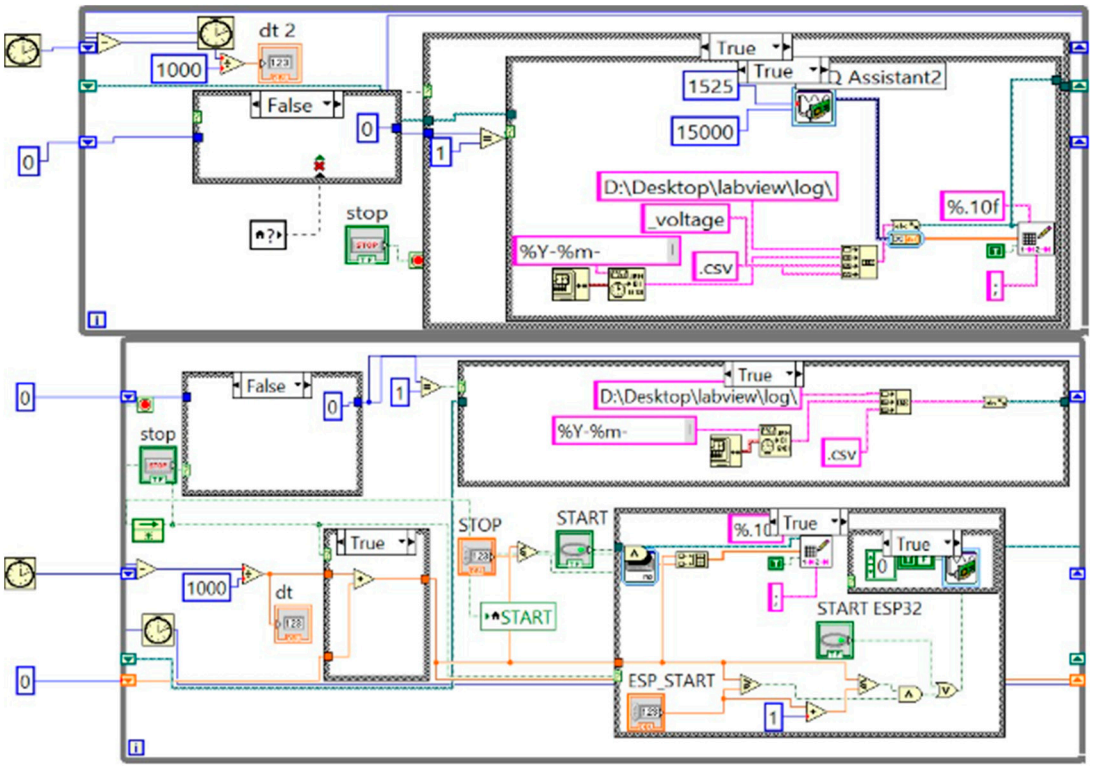


Figure 5. LabVIEW-based monitoring and control interface for real-time voltage and current acquisition during cryptographic operations.

4.2. Dataset and Model Configuration

This study employs the CICIoT2023 dataset, a large-scale benchmark for IoT intrusion detection, collected in a realistic testbed comprising 105 IoT devices. The dataset contains benign traffic and 33 attack types, which are organized into seven higher-level categories: DDoS, DoS, Recon, Web-based, Brute Force, Spoofing, and Mirai. The original release is provided in both PCAP and CSV formats; in this work, the CSV representation was used because it offers directly usable traffic-derived statistical attributes for the learning pipeline.

The dataset includes 47 extracted attributes, covering flow-level duration information, protocol and flag indicators, packet-length statistics, and temporal characteristics. Following the standard preprocessing logic for the dataset, the timestamp field is not used as an input variable, as it primarily preserves ordering information rather than capturing discriminative traffic behavior. Therefore, the federated learning pipeline operates on the cleaned feature matrix derived from the original CICIoT2023 records.

Before model training, the raw dataset underwent a multi-stage preprocessing pipeline. The original CICIoT2023 release consists of 313 separate CSV files, which were merged into a single unified dataset. All rows containing missing values (NaN) were removed, and infinite values were replaced by the maximum finite value within the corresponding feature. Duplicate and near-duplicate records were removed to reduce redundancy and mitigate overfitting. Feature normalization was performed in two steps: the Rate feature was first transformed using a logarithmic function to address its heavily skewed distribution, after which min–max scaling was applied to all features to map values into the [0, 1] range. Class labels were encoded as integer indices compatible with sparse categorical cross-entropy loss.

For the hardware-validated experiments, a controlled subset of 5,000 records was sampled from the preprocessed CICIoT2023 dataset. The subset included 16 selected traffic classes covering benign and representative attack categories. This subset size was selected to match the memory, processing, and runtime constraints of the ESP32-based clients. Since the objective of this study is hardware-validated feasibility rather than large-scale offline IDS benchmarking, the dataset size was chosen to

 doi:10.20944/preprints202605.0029.v1

allow local training, cryptographic processing, and communication to be executed within the limited resources of the microcontroller platform. Before splitting, the selected records were randomized to reduce ordering bias. The dataset was then divided into training and test subsets using an 80:20 split, resulting in 4,000 training samples and 1,000 test samples. The training portion was distributed equally across the three federated clients in an IID manner, so that each client received an identically sized partition with a similar class distribution. The test set was used exclusively for evaluating the final global model. Therefore, the reported classification results should be interpreted as hardware-constrained feasibility results under a controlled 16-class IID setting, rather than as a comprehensive benchmark over the full CICIoT2023 dataset.

The experimental configuration is defined as follows. The number of clients is set to $K = 3$. Training is performed over $R = 20$ global communication rounds. In each round, all clients participate ($C = 1.0$) and perform local training for $E = 1$ epoch. The batch size is set to $B = 32$, and the learning rate is $\eta = 0.001$. This configuration allows evaluation of the proposed framework under realistic resource-constrained IoT conditions.

To perform intrusion detection, a lightweight deep neural network (DNN) based on a multilayer perceptron (MLP) architecture was employed. The model is designed with consideration of the computational and memory constraints of ESP32 devices, providing a balance between detection performance and efficiency. The detailed architecture and training configuration are summarized in Table 3.

Table 3. DNN model architecture and training configuration.

Hyperparameters	Value
Model	DNN
Hidden layers	64, 64, 32
Activation function	ReLU
Output activation	Softmax
Optimizer	Adam
Loss function	Sparse categorical cross-entropy
Learning rate	0.001
Batch size	32
Local epochs	1
Global rounds	20
Validation split	0.20

4.3. Evaluation Metrics

To quantitatively evaluate the overhead introduced by the proposed secure federated learning (FL) framework on resource-constrained ESP32 nodes, we first analyze the resource and computational metrics of the cryptographic layer, followed by an assessment of the intrusion detection system (IDS) model performance.

RAM usage is defined as the minimum amount of dynamic memory required to execute a specific algorithm implementation. It is evaluated using a baseline-difference method: first, a minimal firmware is compiled and flashed onto the same hardware platform; then, the firmware with the integrated cryptographic module is built under identical configuration settings. The RAM value is extracted from the compiler's memory report, and the algorithm-specific consumption is calculated as the difference between the measured value with the algorithm and the baseline measurement.

ROM/FLASH memory represents the size of the program code (program memory). It is evaluated similarly using the baseline-difference approach: the FLASH size after integrating the algorithm is compared with the baseline firmware's FLASH size under identical build conditions.

Encryption/decryption latency is defined as the average time required to process one byte of data on the target device. Each algorithm is executed *N* times with a fixed payload size, and the total

 doi:10.20944/preprints202605.0029.v1

execution time is measured using a built-in timing function. The per-byte latency is then computed as follows:

$$latency\ [s/byte] = \frac{executionTime\ [s]}{N \cdot dataSize\ [bytes]} \quad (7)$$

Throughput is defined as the average number of bytes processed per second and is calculated using the same time measurements:

$$throughput\ [bytes/s] = \frac{N \cdot dataSize\ [bytes]}{executionTime\ [s]} \quad (8)$$

To assess the electrical 'cost' of cryptographic operations, current and power are measured using the NI ELVIS II+ platform and a LabVIEW-based data acquisition system. Voltage and current are recorded synchronously whilst the cryptographic operations are being performed, and instantaneous power is calculated using the formula:

$$Power\ [W] = Voltage\ [V] \cdot Current\ [A] \quad (9)$$

Power consumption is estimated based on the idle state. For each cipher, the average power consumption in idle mode (P_baseline) and in operation (P_active) is measured. The energy cost of the operation is calculated as:

$$E = (P_{active} - P_{baseline}) \cdot t \quad (10)$$

where t is the measured execution time of a single cryptographic operation (encryption or decryption) on the ESP32 for a given payload size and experimental conditions.

Standard classification metrics are used to evaluate the intrusion detection model's performance within the FL framework. For the multi-class CICIoT2023 task, Precision, Recall, and F1-score were computed using a one-vs-rest strategy for each class and then macro-averaged across the 16 classes. Therefore, TP, FP, TN, and FN in Equations (11)–(14) refer to class-wise counts; the analysis also includes confusion matrices for an interpretable analysis of inter-class errors.

Classification accuracy is defined as the proportion of correctly classified instances:

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} \quad (11)$$

Precision indicates the proportion of true positives among all predicted positives:

$$P = \frac{TP}{TP + FP} \quad (12)$$

Recall reflects the proportion of true positives among all actual positives:

$$R = \frac{TP}{TP + FN} \quad (13)$$

The F1 score is the harmonic mean of Precision and Recall:

$$\boldsymbol{F1 = 2 \times \frac{P \times R}{P + R}} \quad (14)$$

All quality metrics are calculated using the final global model obtained after 20 communication rounds in each experiment. Additionally, error matrices are generated to analyze the model's behavior across attack classes and identify typical cases of cross-confusion.

5. Results

This section presents a comprehensive experimental evaluation of the proposed lightweight cryptography–enhanced federated learning framework deployed on ESP32-based IoT nodes. The primary objective is to quantify the computational and energy overhead introduced by secure model-update exchange and to assess its impact on model convergence and intrusion-detection performance. To provide a holistic analysis, the results are organized into four main dimensions:

 doi:10.20944/preprints202605.0029.v1

1. Memory footprint, including RAM and ROM utilization;
2. Cryptographic processing efficiency, measured by latency and throughput;
3. Hardware-level energy consumption during encryption and decryption;
4. Classification performance of the federated IDS model.

This multi-dimensional evaluation enables a joint analysis of security enhancement and system-level efficiency, thereby supporting informed design decisions for secure and resource-efficient federated IoT deployments.

5.1. Hardware-Level Performance Evaluation of Lightweight Cryptography

Within the proposed secure model-update exchange framework, the choice of a lightweight block cipher directly affects the resource cost of protecting serialized updates on the client device. Therefore, this subsection evaluates SPECK, SIMON, and PRESENT on the ESP32 platform with respect to memory usage, latency, throughput, and energy consumption. This analysis provides the hardware-level basis for selecting the most suitable encryption component for resource-constrained federated IoT deployments.

Due to the substantial differences in computational efficiency among SPECK, SIMON, and PRESENT, different iteration counts were used to obtain stable and reproducible timing and electrical measurements. PRESENT was evaluated over 5,000 iterations, SIMON over 500,000 iterations, and SPECK over 1,250,000 iterations, with each operation processing a fixed block size of 8 bytes.

This approach enables comparable measurement durations across experiments, which is particularly important for accurately acquiring the power, voltage, and current traces shown in the figures. The resulting profiles reflect the full execution cycle of each algorithm under realistic operating conditions.

To ensure a fair comparison, all performance metrics, including latency, throughput, and energy consumption, were computed using the average execution time across repeated runs and normalized per operation or per byte. This normalization eliminates the influence of the total number of iterations, enabling objective comparison of cryptographic efficiency across different algorithms. This structured evaluation enables a joint analysis of security enhancement and system-level overhead, thereby supporting evidence-based selection of lightweight cryptographic mechanisms for resource-constrained federated IoT deployments.

Figure 6 presents the RAM and ROM requirements of the evaluated lightweight cipher implementations on the ESP32 platform. These metrics are important for resource-constrained IoT devices because cryptographic protection must be integrated with local training, communication, and federated learning logic within the microcontroller's limited memory.

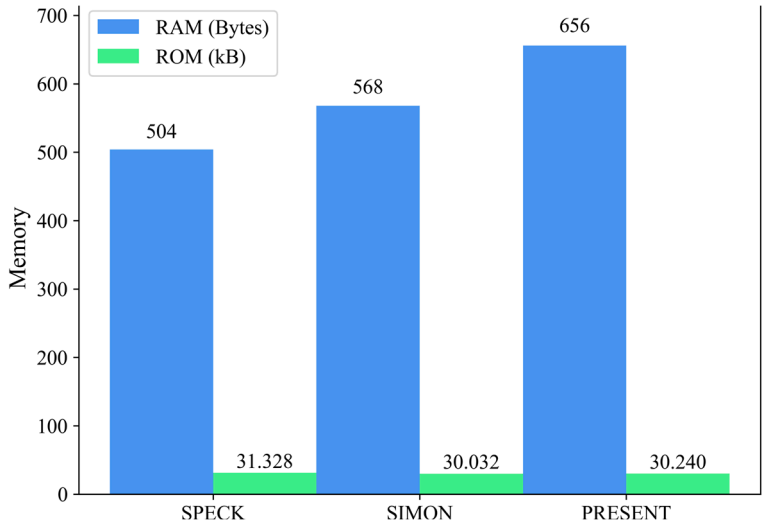


Figure 6. RAM and ROM memory usage of lightweight ciphers on ESP32.

 doi:10.20944/preprints202605.0029.v1

The RAM usage was 504 bytes for SPECK, 568 bytes for SIMON, and 656 bytes for PRESENT. Thus, SPECK showed the lowest RAM requirement, whereas PRESENT exhibited the highest RAM footprint among the three implementations.

The ROM usage values were closer together. SIMON required 30.032 kB of ROM, PRESENT 30.240 kB, and SPECK 31.328 kB. Overall, the difference in ROM usage remained small, while the variation in RAM usage was more noticeable.

The comparison indicates that SPECK is the most memory-efficient implementation in terms of RAM usage, while PRESENT shows the largest RAM footprint. However, the absolute RAM usage remains relatively low for all evaluated ciphers on the ESP32 platform. ROM usage varies only slightly across the implementations, suggesting that code-size overhead is comparable, whereas RAM consumption provides a clearer distinction among the algorithms.

The execution-time performance of the evaluated ciphers is shown in Figure 7, which reports the measured encryption and decryption latency per byte on the ESP32 platform.

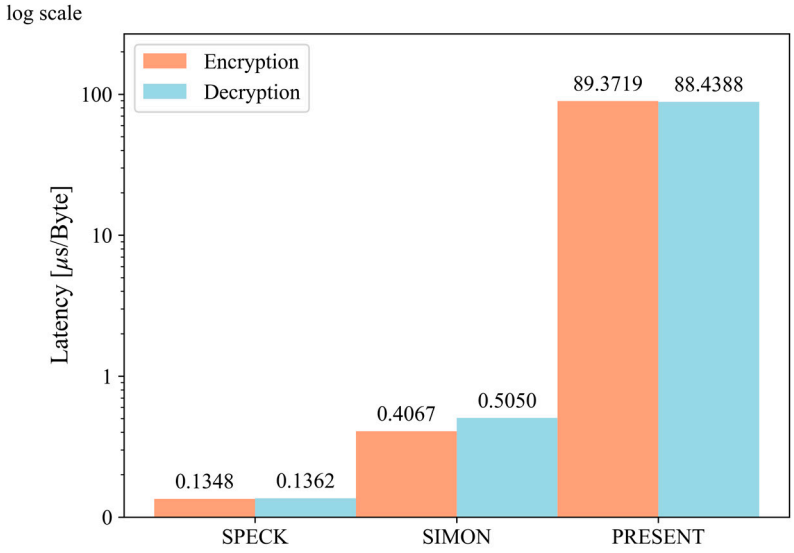


Figure 7. Encryption and decryption latency per byte on ESP32 (μs/B).

As shown in Figure 7, SPECK achieved the lowest latency among the three ciphers, with 0.1348 μs/B for encryption and 0.1362 μs/B for decryption. SIMON showed intermediate performance, reaching 0.4067 μs/B for encryption and 0.5050 μs/B for decryption. In contrast, PRESENT exhibited substantially higher latency values, namely 89.3719 μs/B for encryption and 88.4388 μs/B for decryption.

For each cipher, the encryption and decryption latencies were of similar magnitude, although SIMON showed a somewhat larger difference between the two operations. The largest overall separation was observed between SPECK and PRESENT.

The electrical behavior of the ESP32 during cryptographic processing is illustrated in Figure 8, which shows measured power-consumption traces for encryption and decryption.

As shown in Figure 8(a), the ESP32's baseline power consumption before encryption is approximately 0.21 W. When encryption is initiated (around 5s), power increases to approximately 0.33 W, depending on the selected cipher. PRESENT reaches approximately 0.33 W, SIMON approximately 0.32 W, and SPECK approximately 0.315 W. After completion of the encryption process, power returns to the baseline level.

Figure 8(b) presents the corresponding measurements for decryption. The baseline power level remains approximately 0.21 W before execution. During active decryption, power increases to approximately 0.33 W. PRESENT exhibits the highest sustained power level, followed by SIMON and SPECK. After decryption is completed, the power consumption returns to the idle baseline. Across

 doi:10.20944/preprints202605.0029.v1

both encryption and decryption phases, the application of cryptographic algorithms increases instantaneous power consumption from approximately 0.21 W in the idle state to approximately 0.33 W during active processing.

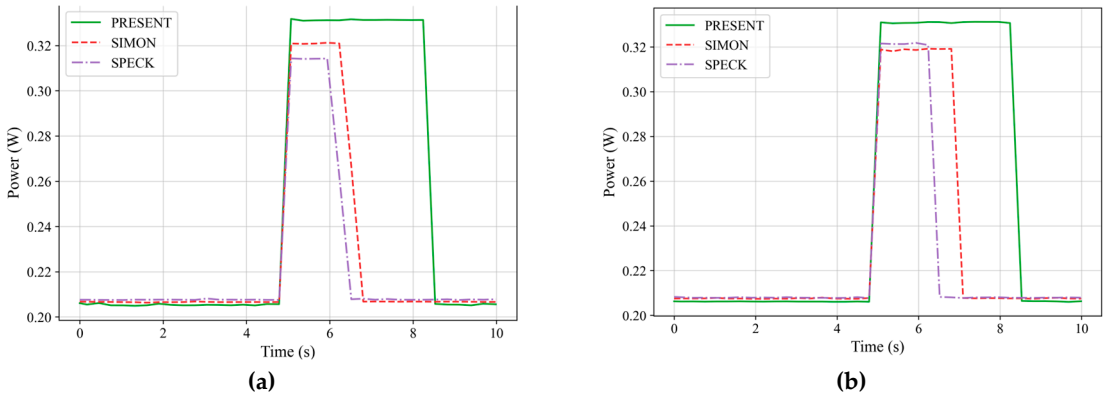


Figure 8. Power consumption during cryptographic processing on the ESP32 platform: (a) power consumption during encryption; (b) power consumption during decryption.

A more detailed view of the electrical measurements is provided in Figure 9, which shows the corresponding voltage and current traces during encryption and decryption.

As shown in Figure 9(a), the supply voltage remains stable at approximately 4.8 V throughout the entire encryption measurement interval. No significant voltage fluctuations are observed during idle or active cryptographic processing. Before encryption, the baseline current consumption is approximately 0.045–0.046 A. When encryption begins (around 5s), the current increases depending on the selected cipher. PRESENT reaches approximately 0.069–0.070 A, SIMON approximately 0.066–0.067 A, and SPECK approximately 0.064–0.065 A. After the encryption process is completed, the current returns to the baseline level.

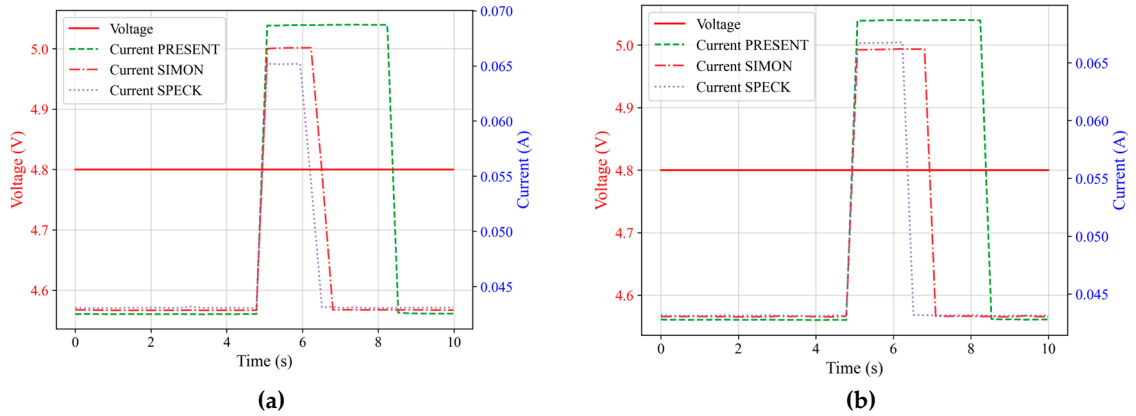


Figure 9. Voltage and current measurements during cryptographic processing on ESP32: (a) Voltage and current during encryption; (b) Voltage and current during decryption.

Figure 9(b) presents the corresponding measurements during decryption. The supply voltage again remains stable at approximately 4.8 V. The baseline current is approximately 0.045–0.046 A before execution. During active decryption, the current increases to approximately 0.069 A for PRESENT, 0.066 A for SIMON, and 0.065 A for SPECK. After completion of decryption, the current state returns to the idle level. Across both encryption and decryption phases, voltage remains constant, while current increases during active cryptographic processing and returns to baseline after completion.

Table 4 presents the measured throughput (in bytes per second) and total energy consumption (in millijoules) for encryption and decryption operations of the evaluated lightweight block ciphers. For encryption, SPECK achieves a throughput of 8,683,179 bytes/s with a total energy consumption

 doi:10.20944/preprints202605.0029.v1

of 138.3 mJ. SIMON demonstrates a throughput of 1,958,763 bytes/s and an energy consumption of 184.9 mJ. PRESENT shows a significantly lower throughput of 11,395 bytes/s, with total energy consumption of 446.2 mJ. For decryption, SPECK reaches 7,411,612 bytes/s with 134.1 mJ energy consumption. SIMON achieves 2,421,105 bytes/s and 241.3 mJ, while PRESENT records 11,188 bytes/s with 466.8 mJ. Across both encryption and decryption operations, measurable differences are observed in both throughput and total energy values among the evaluated ciphers.

Table 4. Throughput and energy consumption of lightweight block ciphers on ESP32.

Cipher	Operation	Throughput (bytes/s)	Energy (mJ)
SPECK	Encryption	8 683 179	138.3
	Decryption	7 411 612	134.1
SIMON	Encryption	1 958 763	184.9
	Decryption	2 421 105	241.3
PRESENT	Encryption	11 395	446.2
	Decryption	11 188	466.8

Overall, the throughput and energy measurements highlight the importance of hardware-level validation when selecting lightweight cryptography for federated IoT systems. Although all three algorithms are lightweight by design, their behavior on the ESP32 platform differs substantially. In the evaluated setting, SPECK offers the strongest balance between speed and energy consumption, SIMON shows acceptable intermediate performance, and PRESENT incurs the highest runtime cost.

5.2. Federated Learning Performance Evaluation

To assess the learning effectiveness of the proposed federated learning framework with secure model-update exchange (implemented using the SPECK), the final global model was evaluated after 20 communication rounds across three ESP32-based edge devices. The quantitative comparison between centralized and federated training paradigms is presented in Table 5. The centralized model achieved an overall accuracy of 86.80%, with macro-averaged precision of 0.88, recall of 0.87, and F1-score of 0.86. The federated learning model achieved 85.43% accuracy, with a precision of 0.87, a recall of 0.85, and an F1-score of 0.84.

These results show that the federated model remains close to the centralized baseline, with an accuracy difference of 1.37 percentage points. This moderate performance gap is expected because the federated setting trains the model across distributed client partitions rather than using all training data in a single centralized process. Nevertheless, the obtained results indicate that the proposed secure federated learning pipeline preserves competitive intrusion detection performance while maintaining data locality and protected model-update exchange.

Table 5. Performance comparison between centralized and federated learning models.

Training Paradigm	Accuracy (%)	Precision	Recall	F1-score
Centralized Learning	86.80	0.88	0.87	0.86
Federated Learning	85.43	0.87	0.85	0.84

Figures 10 and 11 present the confusion matrices corresponding to the centralized and federated training paradigms, respectively. As shown in Figure 10, the centralized model achieves strong detection performance for several attack categories, including DDoS ICMP flood, DDoS TCP flood, DDoS PSHACK, and Mirai. However, misclassification is still observed in specific classes, particularly DDoS SYN flood and DDoS Synonymous IP flood, where precision–recall imbalance and cross-class confusion affect performance. Moderate overlap among MITM, Reconnaissance, and Vulnerability Scan categories also reflects the similarity of probing-based traffic patterns.

 doi:10.20944/preprints202605.0029.v1

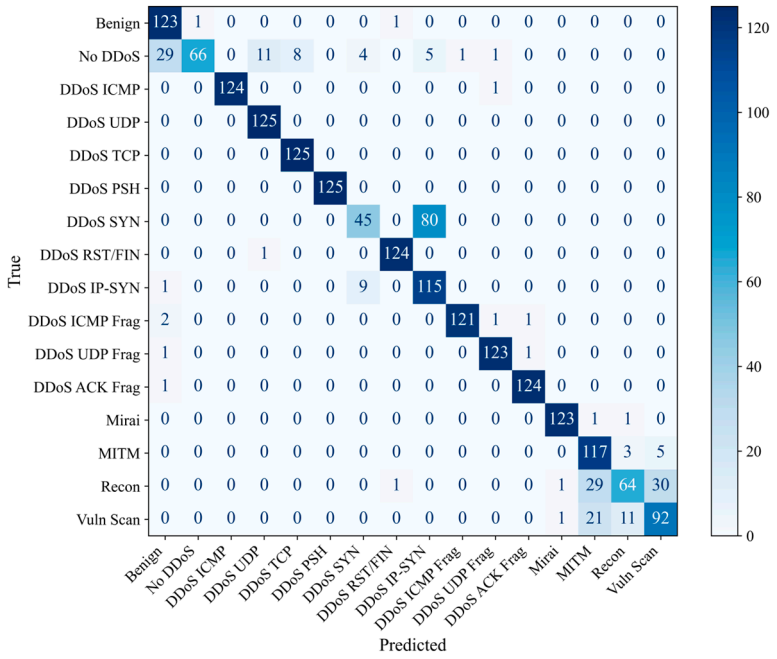


Figure 10. Confusion matrix of the centralized training model.

Figure 11 shows that the federated model preserves a clear diagonal structure, indicating stable class-wise discrimination after distributed training. Although some class-level confusion remains, the overall pattern demonstrates that the federated model maintains effective intrusion detection capability under the distributed training setting. This observation is consistent with the slightly lower but still comparable macro-averaged metrics reported in Table 5.

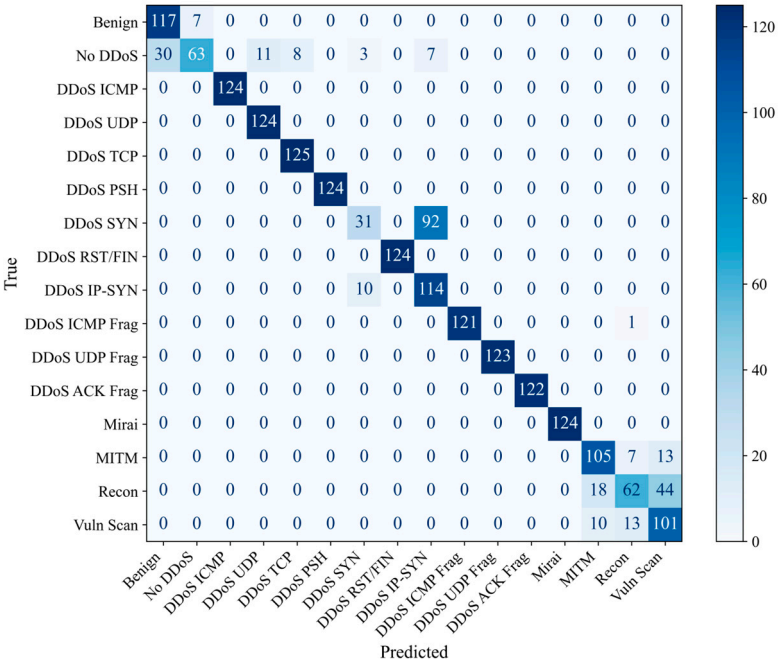
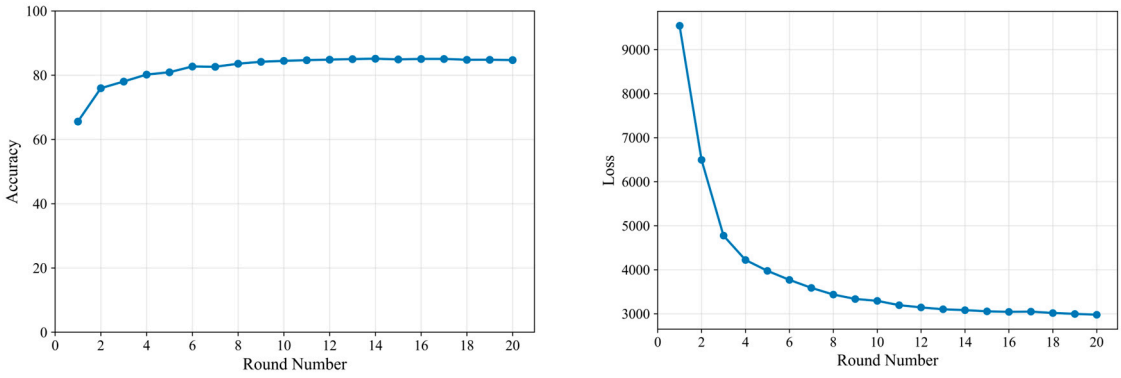


Figure 11. Confusion matrix of the federated learning model after the final communication round.

As shown in Figure 12, the global model demonstrates stable convergence during the secure federated training process. The test accuracy increases progressively across communication rounds, while the test loss decreases, indicating that the protected model-update exchange does not disrupt the FedAvg optimization process. Although minor fluctuations are observed in several intermediate

 doi:10.20944/preprints202605.0029.v1

rounds, the overall trend confirms that the proposed framework maintains stable learning behavior under the evaluated ESP32-based deployment.



(a) (b)

Figure 12. Global model convergence across 20 communication rounds: (a) accuracy; (b) loss.

These convergence results support the final classification metrics reported in Table 5. The secure federated model remains slightly below the centralized baseline, but it achieves competitive intrusion detection performance while preserving data locality and protected model-update exchange. This demonstrates that the integration of lightweight cryptographic protection does not prevent the global model from learning effectively across federated communication rounds.

6. Discussion

The results demonstrate that federated learning with secure model-update exchange is feasible on ESP32-class IoT devices, but its practical efficiency strongly depends on the choice of a lightweight cipher. While all evaluated ciphers can be integrated into the proposed secure update-exchange framework, their latency, throughput, energy, and memory characteristics differ to varying degrees. At the learning level, the protected federated training process maintained stable convergence and achieved classification performance close to the centralized baseline.

The most critical finding is that cipher selection must account for the target processor architecture. SPECK achieved throughput exceeding 8.6 MB/s on the ESP32 and a latency of approximately 0.13 μs/B, whereas PRESENT reached only about 0.011 MB/s and required approximately 89 μs/B. This large gap is mainly explained by how each cipher maps to the ESP32 processor architecture. SPECK relies on word-oriented arithmetic and logical operations, such as modular addition, rotation, and XOR, which are efficiently executed on 32-bit microcontrollers. SIMON also uses lightweight bitwise operations, but its round structure leads to a higher execution cost than SPECK under the evaluated implementation. PRESENT, although lightweight in terms of cryptographic design, is less efficient in software on a general-purpose MCU because its substitution–permutation structure requires repeated 4-bit S-box processing and bit-level permutations. These operations translate into many shift, mask, and rearrangement instructions on the ESP32, explaining its substantially higher latency and lower throughput. The memory results are less pronounced than the runtime differences, but they still show that SPECK has the lowest RAM footprint, whereas PRESENT requires the highest RAM usage among the evaluated implementations. Overall, these findings confirm that a cipher classified as lightweight in the cryptographic literature does not necessarily provide the best execution efficiency on a given embedded platform. Therefore, empirical hardware validation is essential before selecting cryptographic primitives for constrained federated IoT deployments.

The energy results reflect the combined effect of electrical load and execution efficiency. At the hardware level, cryptographic processing increases CPU activity, memory access, and internal

 doi:10.20944/preprints202605.0029.v1

switching in the microcontroller, which raises current draw while the supply voltage remains nearly constant. The power and current traces therefore show the ESP32 moving from an idle state to a higher active operating state during encryption and decryption. PRESENT produces the highest active power and current because its S-box substitutions and bit-level permutations require more masking, shifting, and data rearrangement in software. SPECK, in contrast, relies on word-level ARX operations that map efficiently to the 32-bit ESP32 core, reducing instruction overhead and active electrical load. Combined with the normalized latency and throughput results, this explains why SPECK achieves the lowest energy consumption. In contrast, PRESENT incurs the highest energy cost due to both higher active current/power and slower execution.

Beyond hardware cost, an equally important question is whether the added protection affects the learning process itself. The results in Table 5 show that the secure federated model remains close to the centralized baseline, with an accuracy difference of 1.37 percentage points (85.43% vs. 86.80%). Although the federated setting does not outperform centralized training, Figure 12 shows stable global-model convergence across 20 communication rounds, with accuracy increasing and loss decreasing over time. This moderate gap is expected because federated training operates over distributed client partitions rather than over fully centralized data. Nevertheless, the results indicate that encrypted update exchange and integrity verification do not prevent effective learning under the evaluated ESP32-based deployment.

From a security perspective, using HMAC-SHA256 over the encrypted payload allows the server to verify integrity and authenticity before decryption and aggregation. Because the authentication tag has a fixed length, the added communication overhead remains predictable for each protected update. This mechanism strengthens the reliability of model-update exchange against in-transit tampering, although it does not address malicious behavior by legitimate participating clients.

Compared with homomorphic encryption-based FL schemes such as FedML-HE [13], the proposed framework targets a lower-cost security layer that is more compatible with microcontroller-class clients. Blockchain-based approaches such as ShieldDFL [43] can provide stronger trust and auditability guarantees, but they also introduce coordination latency and infrastructure requirements that are difficult to accommodate on constrained hardware. Differential privacy approaches are communication-efficient and can strengthen privacy, yet they do not directly ensure the integrity of updates in transit. In this context, the proposed framework occupies a practical middle ground by combining confidentiality and integrity protection with hardware-validated feasibility on ESP32-based IoT devices.

The first limitation concerns the experimental scale. The evaluation involved three ESP32-based clients and 20 federated learning rounds, which reflects a small-scale deployment. This configuration was selected because local training, cryptographic processing, communication, and aggregation were executed on physical microcontroller hardware rather than in simulation, resulting in substantial runtime cost. Consequently, the present results demonstrate feasibility and stable learning behavior under realistic embedded conditions, but they do not yet establish scalability to larger federated systems. Broader experiments with more clients, longer training schedules, and additional communication rounds remain necessary.

Additional limitations also remain. The evaluation was conducted under controlled wireless conditions, whereas real-world RF interference and channel instability may affect communication latency and reliability. The framework currently relies on session keys established via ECDH, and the scalability of this key-management procedure in larger deployments requires further study. Although CICIoT2023 is comprehensive, it reflects a specific IoT traffic profile, and generalization to other application domains should be validated empirically. Finally, HMAC-SHA256-based verification provides integrity and authenticity for transmitted updates but does not provide non-repudiation or explicit source attribution in the presence of compromised legitimate clients. Future work should therefore consider larger deployments, additional datasets, and stronger trust mechanisms, including digital signatures or robust aggregation methods.

 doi:10.20944/preprints202605.0029.v1

7. Conclusion

This paper presented a hardware-validated approach for integrating lightweight cryptographic protection into federated learning pipelines on resource-constrained IoT devices. The proposed framework combines on-device encryption of model updates using lightweight block ciphers with HMAC-SHA256-based integrity verification, enabling secure model-update exchange while preserving practical feasibility on microcontroller-class hardware.

Experimental results on ESP32 microcontrollers demonstrated that the choice of cryptographic algorithm has a significant impact on overall system performance. In particular, SPECK achieved the best efficiency in terms of latency, throughput, and energy consumption, whereas PRESENT showed substantial performance limitations due to its poor software efficiency on a general-purpose 32-bit microcontroller. These findings highlight the importance of matching cryptographic design to the target hardware platform when developing secure federated IoT systems.

At the learning level, the proposed secure federated framework maintained stable convergence across communication rounds and achieved classification performance close to the centralized baseline. Although the federated model remained slightly below centralized training in the final evaluation, the results confirm that the integration of lightweight cryptographic protection does not prevent effective learning under the evaluated ESP32-based deployment. From a practical perspective, the study demonstrates that secure federated learning on real microcontroller-based IoT devices is feasible and can be implemented with acceptable overhead when the selected cryptographic primitive is well aligned with the hardware architecture.

Future work will focus on larger-scale validation of the proposed framework, including more clients, additional communication rounds, and evaluation on a broader range of IoT datasets and hardware platforms.

Author Contributions: Conceptualization, Y.T., A.A. and N.A.; methodology, Y.T. and Z.A; software, Y.T.; validation, Y.T., J.A. and N.A.; formal analysis, Y.T. and S.L.; investigation, Y.T. and D.D.; resources, N.A. and D.D.; data curation, Y.T. and D.O; writing—original draft preparation, Y.T.; writing—review and editing, Y.T., N.A., D.D. and S.L.; visualization, Y.T. and Z.A; supervision, N.A., A.A. and D.D.; project administration, N.A.; funding acquisition, D.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Committee of Science of the Ministry of Science and Higher Education of the Republic of Kazakhstan (Grant No. BR24993166)

Data Availability Statement: The CICIoT2023 dataset used in this study is publicly available at https://www.unb.ca/cic/datasets/iotdataset-2023.html. The source code of the proposed framework is publicly available at https://github.com/SU-developer-team/esp32-federated-learning (accessed on 9 April 2026). Additional data supporting the findings of this study are available from the corresponding authors upon reasonable request.

Acknowledgments: During the preparation of this work, the authors used GrammarlyGO Version 9.95.0, an AI writing assistant, in order to improve the flow, style, and grammar of the manuscript. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
FL	Federated Learning
RAM	Random Access Memory
ROM	Read-Only Memory
IDS	Intrusion Detection System

 doi:10.20944/preprints202605.0029.v1

SoC	System on Chip
USB	Universal Serial Bus
GPIO	General Purpose Input/Output
SRAM	Static Random Access Memory
PSRAM	Pseudo Static Random Access Memory
SHA	Secure Hash Algorithm
CPU	Central Processing Unit
GPU	Graphics Processing Unit
Wi-Fi	Wireless Fidelity

References

1. Dubey, K.; Dubey, R.; Panedy, S.; Kumar, S. A Review of IoT Security: Machine Learning and Deep Learning Perspective. *Procedia Comput. Sci.* **2024**, *235*, 335–346, doi:10.1016/j.procs.2024.04.034.
2. Tiburski, R.T.; Moratelli, C.R.; Johann, S.F.; Neves, M.V.; Matos, E. de; Amaral, L.A.; Hessel, F. Lightweight Security Architecture Based on Embedded Virtualization and Trust Mechanisms for IoT Edge Devices. *IEEE Commun. Mag.* **2019**, *57*, 67–73, doi:10.1109/MCOM.2018.1701047.
3. Shen, S.; Zhang, K.; Zhou, Y.; Ci, S. Security in Edge-Assisted Internet of Things: Challenges and Solutions. *Sci. China Inf. Sci.* **2020**, *63*, 220302, doi:10.1007/s11432-019-2906-y.
4. Nguyen, T.D.; Marchal, S.; Miettinen, M.; Fereidooni, H.; Asokan, N.; Sadeghi, A.-R. DÏoT: A Federated Self-Learning Anomaly Detection System for IoT. In Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS); IEEE: Dallas, TX, USA, July 2019; pp. 756–767.
5. Ferrag, M.A.; Friha, O.; Hamouda, D.; Maglaras, L.; Janicke, H. Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning. *IEEE Access* **2022**, *10*, 40281–40306, doi:10.1109/ACCESS.2022.3165809.
6. Campos, E.M.; Saura, P.F.; González-Vidal, A.; Hernández-Ramos, J.L.; Bernabé, J.B.; Baldini, G.; Skarmeta, A. Evaluating Federated Learning for Intrusion Detection in Internet of Things: Review and Challenges. *Comput. Netw.* **2022**, *203*, 108661, doi:10.1016/j.comnet.2021.108661.
7. Danquah, L.K.G.; Appiah, S.Y.; Mantey, V.A.; Danlard, I.; Akowuah, E.K. Computationally Efficient Deep Federated Learning with Optimized Feature Selection for IoT Botnet Attack Detection. *Intell. Syst. Appl.* **2025**, *25*, 200462, doi:10.1016/j.iswa.2024.200462.
8. Sun, S.; Sharma, P.; Nwodo, K.; Stavrou, A.; Wang, H. FedMADE: Robust Federated Learning for Intrusion Detection in IoT Networks Using a Dynamic Aggregation Method 2024.
9. Rey, V.; Sánchez Sánchez, P.M.; Huertas Celdrán, A.; Bovet, G. Federated Learning for Malware Detection in IoT Devices. *Comput. Netw.* **2022**, *204*, 108693, doi:10.1016/j.comnet.2021.108693.
10. Liu, S.; Xie, R.; Miao, Y.; Peng, J.; Leng, T.; Liu, Z.; Raymond Choo, K.-K. Efficient and Secure Federated Knowledge Transfer Under Non-IID Settings in IoT. *IEEE Internet Things J.* **2025**, *12*, 24644–24655, doi:10.1109/JIOT.2025.3555646.
11. Latif, S.; Louadj, R.; Djenouri, D. Federated Learning Meets Recursive Self-Distillation: A Scalable Malware Detection Framework for IoVs. In Proceedings of the 2025 IEEE 101st Vehicular Technology Conference (VTC2025-Spring); IEEE: Oslo, Norway, June 17 2025; pp. 1–5.
12. Niu, S.; Zhang, Y.; Wang, W.; Hu, L.; Nan, X. Lightweight Verifiable Federated Learning Based on Matrix Theory. *Neurocomputing* **2025**, *647*, 130579, doi:10.1016/j.neucom.2025.130579.
13. Jin, W.; Yao, Y.; Han, S.; Gu, J.; Joe-Wong, C.; Ravi, S.; Avestimehr, S.; He, C. FedML-HE: An Efficient Homomorphic-Encryption-Based Privacy-Preserving Federated Learning System 2024.
14. Jalali, N.A.; Chen, H. Federated Learning Security and Privacy-Preserving Algorithm and Experiments Research Under Internet of Things Critical Infrastructure. *Tsinghua Sci. Technol.* **2024**, *29*, 400–414, doi:10.26599/TST.2023.9010007.
15. Huang, Y.; Xu, G.; Wang, Q.; Song, X.; Wang, X. Efficient and Privacy-Preserving Authentication for Federated Learning in Industrial Internet of Things Data Sharing Application. *IEEE Internet Things J.* **2025**, *12*, 11652–11663, doi:10.1109/JIOT.2024.3516515.

 doi:10.20944/preprints202605.0029.v1

16. Campolo, C.; Genovese, G.; Singh, G.; Molinaro, A. Scalable and Interoperable Edge-Based Federated Learning in IoT Contexts. *Comput. Netw.* **2023**, *223*, 109576, doi:10.1016/j.comnet.2023.109576.
17. Dos Santos, L.E.B.; Ramos, R.G.S.; Gomes, R.D.; Lins, P.R.; Dos Santos, A.L. Middleware for Smart Campus applications based on Federated Learning. In Proceedings of the Brazilian Symposium on Multimedia and the Web; ACM: Curitiba Brazil, November 7 2022; pp. 324–328.
18. Zakariyya, I.; Kalutarage, H.; Al-Kadri, M.O. Resource Efficient Federated Deep Learning for IoT Security Monitoring. In *Attacks and Defenses for the Internet-of-Things*; Li, W., Furnell, S., Meng, W., Eds.; Lecture Notes in Computer Science; Springer Nature Switzerland: Cham, 2022; Vol. 13745, pp. 122–142 ISBN 978-3-031-21310-6.
19. Albanbay, N.; Tursynbek, Y.; Graffi, K.; Uskenbayeva, R.; Kalpeyeva, Z.; Abilkaiyr, Z.; Ayapov, Y. Federated Learning-Based Intrusion Detection in IoT Networks: Performance Evaluation and Data Scaling Study. *J. Sens. Actuator Netw.* **2025**, *14*, 78, doi:10.3390/jsan14040078.
20. Mouad Alyas Al_Azzawi, R.; Salim Mahmood Al-Dabbagh, S. A LIGHTWEIGHT ENCRYPTION ALGORITHM TO SECURE IOT DEVICES. *MINAR Int. J. Appl. Sci. Technol.* **2023**, *05*, 37–62, doi:10.47832/2717-8234.16.3.
21. Ibrahim, N.; Agbinya, J. Design of a Lightweight Cryptographic Scheme for Resource-Constrained Internet of Things Devices. *Appl. Sci.* **2023**, *13*, 4398, doi:10.3390/app13074398.
22. Nagesh, H.R.; Poojari, A.; Kumar, V.G.K. Design, Implementation and Analysis of HIBRI Cipher on IoT Platforms. *J. Inst. Eng. India Ser. B* **2024**, *105*, 1–13, doi:10.1007/s40031-024-01067-2.
23. Witwit, A.J.H.; Fanfakh, A.; Idrees, A.K. SUL32C: Secure Ultra-Lightweight 32-Bit Cipher for IoT Devices. *J. Supercomput.* **2025**, *81*, 1042, doi:10.1007/s11227-025-07448-8.
24. Wulandari, A.; Ramadhani, E.H.; Fathurahman, M.; Hikmaturokhman, A. Implementation Analysis of the PRESENT Algorithm on ATmega328P for Securing RFM95W LoRa Communications. In Proceedings of the 2024 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT); IEEE: Mataram, Indonesia, November 28 2024; pp. 615–621.
25. Băneasă, A.; Donca, R.; Besoiu, S.; Buleandră, D. Lightweight Implementation of the AES Encryption Algorithm for IoT Applications Constrained by Memory and Processing Power. In Proceedings of the 2024 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR); IEEE: Cluj-Napoca, Romania, May 16 2024; pp. 1–6.
26. Ni, L.C.; Ali, S.; Aziz, A.N.A.A.; Rashid, R.A. Implementation of Proposed Cryptography Algorithm on ESP32-Based IoT System. In Proceedings of the 2024 IEEE International Conference on Advanced Telecommunication and Networking Technologies (ATNT); IEEE: Johor Bahru, Malaysia, September 9 2024; pp. 1–4.
27. Clemente-Lopez, D.; Rangel-Magdaleno, J.D.J.; Muñoz-Pacheco, J.M. A Lightweight Chaos-Based Encryption Scheme for IoT Healthcare Systems. *Internet Things* **2024**, *25*, 101032, doi:10.1016/j.iot.2023.101032.
28. Jiteurtragool, N.; Samkunta, J.; Ketthong, P. Lightweight Parabola Chaotic Keyed Hash Using SRAM-PUF for IoT Authentication. *Int. J. Adv. Comput. Sci. Appl.* **2025**, *16*, doi:10.14569/IJACSA.2025.0160273.
29. Ulla, M.M.; Preethi; Khan, Md.S.; L, S.; J, S.B. Lightweight Strobe Security Libdisco Scheme for IoT-Based Sensor Data. In Proceedings of the 2024 8th International Conference on Computational System and Information Technology for Sustainable Solutions (CSITSS); IEEE: Bengaluru, India, November 7 2024; pp. 1–6.
30. Bansod, G. A New Ultra Lightweight Encryption Design for Security at Node Level. *Int. J. Secur. Its Appl.* **2016**, *10*, 111–128, doi:10.14257/ijsia.2016.10.12.10.
31. Chen, Y.; Liu, L.; Ping, Y.; Atiquzzaman, M.; Mumtaz, S.; Zhang, Z.; Guizani, M.; Tian, Z. A Privacy-Preserving Federated Learning Framework With Lightweight and Fair in IoT. *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 5843–5858, doi:10.1109/TNSM.2024.3418786.
32. Fan, M.; Ji, K.; Zhang, Z.; Yu, H.; Sun, G. Lightweight Privacy and Security Computing for Blockchained Federated Learning in IoT. *IEEE Internet Things J.* **2023**, *10*, 16048–16060, doi:10.1109/JIOT.2023.3267112.

 doi:10.20944/preprints202605.0029.v1

33. Awan, K.A.; Din, I.U.; Almogren, A.; Rodrigues, J.J.P.C. Privacy-Preserving Big Data Security for IoT With Federated Learning and Cryptography. *IEEE Access* **2023**, *11*, 120918–120934, doi:10.1109/ACCESS.2023.3328310.
34. Latif, S.; Djenouri, D. A Two-Tier Secure Federated Learning Framework with Lightweight Cryptography for Edge-Cloud Collaboration. In Proceedings of the 2025 IEEE 35th International Telecommunication Networks and Applications Conference (ITNAC); IEEE: Christchurch, New Zealand, November 26 2025; pp. 1–6.
35. Zhang, C.; Zhang, X.; Yang, X.; Liu, B.; Zhang, Y.; Zhou, R. Poisoning Attacks Resilient Privacy-Preserving Federated Learning Scheme Based on Lightweight Homomorphic Encryption. *Inf. Fusion* **2025**, *121*, 103131, doi:10.1016/j.inffus.2025.103131.
36. Wang, X.; Liu, Z.; Huang, B. Robust and Privacy-Preserving Federated Learning Scheme Based on Ciphertext-Selected Users. *Comput. Netw.* **2025**, *259*, 111072, doi:10.1016/j.comnet.2025.111072.
37. Hu, G.; Hu, Y.; Wu, T.; Zhang, Y.; Yuan, S. Lightweight Distributed Deep Learning on Compressive Measurements for Internet of Things. *Eng. Appl. Artif. Intell.* **2025**, *139*, 109581, doi:10.1016/j.engappai.2024.109581.
38. Wei, Z.; Pei, Q.; Zhang, N.; Liu, X.; Wu, C.; Taherkordi, A. Lightweight Federated Learning for Large-Scale IoT Devices With Privacy Guarantee. *IEEE Internet Things J.* **2023**, *10*, 3179–3191, doi:10.1109/JIOT.2021.3127886.
39. Amrita; Ekwueme, C.P.; Adam, I.H.; Dwivedi, A. Lightweight Cryptography for Internet of Things: A Review. *EAI Endorsed Trans. Internet Things* **2024**, *10*, doi:10.4108/eetiot.5565.
40. Gunathilake, N.A.; Al-Dubai, A.; Buchana, W.J. Recent Advances and Trends in Lightweight Cryptography for IoT Security. In Proceedings of the 2020 16th International Conference on Network and Service Management (CNSM); IEEE: Izmir, Turkey, November 2 2020; pp. 1–5.
41. Soto-Cruz, J.; Ruiz-Ibarra, E.; Vázquez-Castillo, J.; Espinoza-Ruiz, A.; Castillo-Atoche, A.; Mass-Sanchez, J. A Survey of Efficient Lightweight Cryptography for Power-Constrained Microcontrollers.
42. Chinbat, T.; Madanian, S.; Airehrour, D.; Hassandoust, F. Machine Learning Cryptography Methods for IoT in Healthcare. *BMC Med. Inform. Decis. Mak.* **2024**, *24*, 153, doi:10.1186/s12911-024-02548-6.
43. Cai, Y.; Du, X.; Zhang, C.; Li, M. ShieldDFL: A Blockchain-Based Federated Learning Framework With Dual Privacy Protection and Reputation-Driven Consensus. *IEEE Access* **2025**, *13*, 103931–103943, doi:10.1109/ACCESS.2025.3576261.