Secure IoT Communications With Optimized AES and Dynamic Threat Modeling on Embedded System

Koustav Kumar Mondal, *Graduate Student Member, IEEE*, Himani Sikarwar, *Member, IEEE*, Debasis Das, *Senior Member, IEEE*, and Chun-I Fan, *Senior Member, IEEE*

***Abstract*— This paper presents a hardware cryptographic solution that leverages an InvenSense 6-axis Motion Processing Unit (MPU6050) sensor and an ESP32 microcontroller to address the need for secure and efficient communication in resource-constrained Internet of Things (IoT) devices, we introduce a hardware-conscious, lightweight AES-128 with a two-phase (4-bit) S-box and simplified MixColumns over $GF(2^8)$, integrated with Elliptic Curve Diffie–Hellman (ECDH) for forward-secret key establishment, and a dynamic threat-modeling algorithm that quantifies risk and triggers targeted mitigations. A reconfigurable architecture physically separates block-cipher and hash paths, enabling concurrent operations needed by protocols such as TLS/IPsec. On ESP32, our device reduces encryption time by up to 42.4% compared to representative baselines; it achieves 28.57% at 100 KB and 30% at 50 KB. Energy consumption decreases by 16.7%–28.57% across payloads, while throughput reaches 435 Mbps (about +3.57% over the strongest comparator at 500 KB). The one-time ECDH overhead is 8.1 ms/1.14 mJ and amortizes to <5% of total energy for payloads >100 KB. These results, obtained under rigorous experimental profiling, demonstrate that the proposed hardware–algorithm co-design delivers a secure, high-performance, and energy-efficient cryptographic substrate for diverse IoT deployments, strengthening both confidentiality and resilience while fitting tight resource budgets.**

***Index Terms*— Embedded security, IoT encryption, microcontroller, ESP32, scalable security solutions, AES128.**

I. Introduction

THE rapid expansion of the Internet of Things (IoT) has led to a vast network of interconnected devices operating under stringent resource constraints such as limited processing power, memory, and energy availability [1], [2]. Ensuring secure communication in such environments is challenging due to the computational overhead associated with standard cryptographic algorithms [3]. Traditional software implementations of encryption algorithms like AES-128 are inefficient on low-power devices, leading to increased latency and energy consumption [4]. In practice, protocols and frameworks (e.g., 5G, ZigBee, Wi-Fi, TLS, IPsec) mandate confidentiality and integrity via block ciphers (DES, AES, SM4) and hashes (SHA-1/256, SM3); on MCU-class nodes, software-only stacks often violate latency/energy budgets [5]. The problem addressed here is to deliver standards-compliant cryptography on resource-constrained IoT devices—with forward secrecy and quantified security—while meeting tight time/energy constraints. We propose a hardware-conscious co-design: an ESP32-based device with an InvenSense 6-axis Motion Processing Unit (MPU6050) sensor, running a lightweight AES-128 optimized by a two-phase (4-bit) S-box and simplified MixColumns over the Galois Field of order 2^8 ($GF(2^8)$), combined with Elliptic Curve Diffie–Hellman (ECDH) for session-key establishment and forward secrecy. To support regional standards and integrity checks without bottlenecks, the reconfigurable architecture physically separates block-cipher and hash datapaths (AES/DES/SM4 and SHA-1/SHA-256/SM3), enabling concurrent operation suitable for TLS/IPsec; composite-field S-box reuse and shared register files reduce memory and logic complexity. Key outcomes are: (i) a lightweight AES-128 tailored to ESP32 that reduces encryption time by up to 42.4% and lowers energy by 16.7%–28.57%; (ii) reconfigurable, separated cipher/hash paths that enable concurrency for TLS/IPsec and multi-standard support; (iii) ECDH with a one-time 8.1 ms/1.14 mJ overhead amortizing to below 5% for payloads over 100 KB; (iv) a dynamic threat-modeling algorithm that quantifies and mitigates risks across physical, side-channel, software, protocol, and network vectors; and (v) overall throughput reaching 435 Mbps on ESP32, aligning with real-time IoT demands.

The paper first states contributions and context, details the embedded device architecture and lightweight AES methodology, describes the experimental setup, presents performance results (time, energy, throughput) together with ECDH amortization, and finally analyzes security via the threat model before concluding with application-facing takeaways. This roadmap allows the Introduction to lead the reader through the main ideas and benefits that we quantify in later sections.

Received 16 June 2025; revised 18 September 2025 and 3 November 2025; accepted 10 December 2025. Date of publication 16 December 2025; date of current version 25 March 2026. This work was supported partially by the Technology Innovation Hub Foundation for Internet of Things and Internet of Everything (TIH-IoT) under Project TDP06-A-26 and the National Science and Technology Council of Taiwan under Grant NSTC 114-2634-F-110-001-MBK. *(Corresponding author: Debasis Das.)*

Koustav Kumar Mondal is with the Centre for AIoT and Applications, IIT Jodhpur, Jodhpur, Rajasthan 342037, India (e-mail: mondal.4@iitj.ac.in).

Himani Sikarwar is with the Department of CSE, Shiv Nadar Institute of Eminence, Delhi 201314, India (e-mail: himani.sikarwar@snu.edu.in).

Debasis Das is with the Department of CSE, IIT Jodhpur, Jodhpur, Rajasthan 342037, India (e-mail: debasis@iitj.ac.in).

Chun-I Fan is with the Department of CSE, NSYSU, Kaohsiung 80424, Taiwan (e-mail: cifan@mail.cse.nsysu.edu.tw).

Digital Object Identifier 10.1109/TCE.2025.3645059

1558-4127 © 2025 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Melbourne. Downloaded on November 03,2026 at 12:30:39 UTC from IEEE Xplore. Restrictions apply.

TABLE I
DEVICE-LEVEL EMBEDDED-CRYPTOGRAPHY WORKS

Schemes	Primitive(s)	Reconfigurable (DES/SM4 + Hash)	Throughput Class	Energy/Power Focus	SCA/Fault Focus	KEX/Protocol	Notes
Zhang et al. [6]	AES/DES/SM4 + SHA-1/256/SM3	✓(shared datapath)	MCU/ASIC	✓	✗	✗	Unified cipher/hash logic
Zhang et al. [7]	AES-only (byte-serial)	✗	High (ASIC, 22 nm)	✓	✗	✗	1,257 Gbps/W (AES)
Lone et al. [8]	AES-only	✗	MCU/FPGA	✓	✗	✗	Low-power AES tuning
Suárez-Albela et al. [9]	AES/ECC/RSA (SW on MCU)	✗	MCU (SW)	✓(freq scaling)	✗	✗	Software-level trade-offs
Pan et al. [10]	AES coprocessor (RISC-V)	✗	MCU/SoC	✓	✗	✗	Custom instructions, AES-only
Lin et al. [11]	AES-GCM (S-box/GHASH opt.)	✗	Very High (FPGA/ASIC)	✓	✗	✗	57.17 Gbps (FPGA)
Pei et al. [12]	AES-256 (ULP core)	✗	ULP ASIC	✓	✗	✗	1.83 μW @ 0.3 V
Padmakumar et al. [13]	AES (split SubBytes)	✗	MCU/FPGA	✓	✓(SCA)	✗	SCA-aware substitution
Ahmed et al. [14]	AES (survey/DFA)	✗	N/A	✓	✓(faults)	✗	Survey of DFA-resistant AES
Batmaz et al. [15]	AES (SoC-FPGA, DTLS)	✗	SoC/FPGA	✓	✗	✓(DTLS)	8.1× DTLS speedup
Duong et al. [16]	AES (RHCA reconfigurable)	✓(AES modes)	FPGA	✓	✗	✗	5.8 Gbps, 30% resource save
Goswami et al. [17]	ECC (ECDH, FDM)	N/A	FPGA	✓	✗	✓(ECDH)	2–3× over prior ECC
Alkhaleel et al. [18]	ECC (Edwards, serial/parallel)	N/A	FPGA	✓	✗	✓(ECC)	Up to 3.5× speedup
Pham Quoc et al. [19]	ECDSA (edge core)	N/A	Edge/FPGA	✓	✗	✓(ECDSA)	3,361 sig/s (edge)
Proposed Work	AES-128 + SHA-1/256/SM3 + ECDH	**✓(physically separated)**	MCU (ESP32)	✓	**✓(formal & sim.)**	**✓(ECDH)**	Concurrency of cipher & hash

II. CONTRIBUTIONS

Operating on MCU-class IoT nodes that must meet tight latency/energy budgets while remaining standards-compliant and forward-secret—and while supporting concurrent integrity checks (TLS/IPsec) and resisting physical/side-channel threats—we contribute:

- **Hardware–algorithm co-design for constrained IoT.** A reconfigurable, ESP32-based device that physically separates block-cipher and hash datapaths (AES/DES/SM4 and SHA-1/256/SM3) for true concurrency, with ECDH establishing forward-secret session keys—avoiding software bottlenecks on MCU platforms.
- **Lightweight AES-128 with quantified security hooks.** A hardware-conscious AES-128 (two-phase 4-bit S-box + simplified MixColumns over $GF(2^8)$) paired with a dynamic threat-modeling algorithm to evaluate and mitigate risks across physical, side-channel, software, protocol, and network attack surfaces.
- **Measured benefits on ESP32.** Up to 42.4% lower encryption time, 16.7%–28.57% lower energy, and 435 Mbps throughput; ECDH's one-time overhead (8.1 ms/1.14 mJ) amortizes to <5% of total energy beyond 100 KB payloads.

III. RELATED WORK

Recent advances in hardware acceleration continue to raise performance, energy efficiency, and security for AES/ECC on constrained IoT nodes. Representative results include a pipelined AES-GCM reaching 57.17 Gbps (FPGA) and 50.49 Gbps (ASIC) via S-box/GHASH optimizations [11], and an AES-256 core operating at 1.83 μW@0.3 V with 3881 Gbps/W [12]. Security-oriented designs span a CFB variant resilient to power analysis [25] and a split SubBytes that reduces leakage with low overhead [13], alongside surveys of DFA-resistant trade-offs in S-box/MixColumns [14]. At the protocol/system level, SoC-FPGA AES cuts DTLS latency by 8.1× with lower energy [15]. In ECC, Edwards-curve processors achieve up to 3.5× speedups [18], and an edge ECDSA core delivers 3,361 sig/s [19]. Co-design trends show reconfigurable AES (RHCA) at 5.8 Gbps with 30% resource savings [16], while frequency-domain multipliers yield 2–3× ECC gains [17]. Despite this progress, jointly optimizing throughput, multi-cipher agility, robustness, and low energy/area remains an open challenge. Within this landscape, recent research targets higher throughput and energy-efficient cryptographic designs for IoT. Zhang et al. [6] present a reconfigurable accelerator for DES/AES/SM4 and SHA-1/256/SM3 emphasizing area/energy, but its unified cipher–hash datapath can add latency under concurrent use; Zhang et al. [7] deliver a byte-serial AES core with 1,257 Gbps/W in 22 nm, yet it is AES-only; Lone et al. [8] optimize AES for low-power devices but do not support multi-cipher or hashing; Suárez-Albela et al. [9] study frequency scaling of AES/RSA/ECC in MCU software, improving energy in some regimes but without specialized hardware; Pan et al. [10] add a lightweight RISC-V AES coprocessor achieving packet-level speedups, though without DES/SM4 or hashing re-configurability. A concise, side-by-side device-level comparison appears in Table I. Taken together, [6], [7], [8], [9], and [10] cluster into: (i) ultra–low-area or high-throughput AES-only designs, (ii) partial multi-cipher solutions with unified block-cipher/hash datapaths, and (iii) software–hardware trade-off studies. In contrast, we *physically decouple* the block-cipher (DES/AES/SM4) and hash (SHA-1/SHA-256/SM3) engines to enable true concurrency on MCU-class nodes (see Table I). Beyond accelerator-centric designs, several system/protocol studies target higher-layer security and communication in IoT [20], [21], [22], [23], [24]. In brief, [20] explores blockchain-backed searchable encryption for healthcare; [21] develops ML-based DDoS detection; [22] proposes a hybrid MAC for interoperability; [23] studies AI-enabled data fusion/load optimization; and [24] designs lightweight mutual authentication for IIoT. These works are complementary and can layer above the AES/ECDH engine proposed here; a compact summary is provided in Table II.

IV. PROPOSED EMBEDDED DEVICE ARCHITECTURE

The proposed embedded device integrates the lightweight AES-128 encryption algorithm with the MPU6050 sensor module and the ESP32 microcontroller to enhance data encryption efficiency and security in resource-constrained IoT applications. This section details the hardware architecture, power considerations, and communication interfaces, along with the role of the ESP32's crypto engine in accelerating encryption tasks. The design leverages both hardware acceleration and algorithmic optimizations, ensuring efficient cryptographic processing with reduced power consumption.

Authorized licensed use limited to: University of Melbourne. Downloaded on November 03,2026 at 12:30:39 UTC from IEEE Xplore. Restrictions apply.

TABLE II
SUMMARY OF SYSTEM/PROTOCOL WORKS AND RELATION TO THIS PAPER

Schemes	Focus	Method	Eval.	Relation
Ali et al. [20]	PHR access/search	Blockchain + SSE	Proto., perf./sec.	Uses on-node AES/ECDH
Mohmand et al. [21]	DDoS detection	Supervised ML	Accuracy/CM	Complementary IDS
Khan et al. [22]	Interop. MAC	Hybrid MAC	Analytical/case	Above crypto layer
Jan et al. [23]	Load/latency	AI fusion + opt.	Sim. trends	Orchestration layer
Adil et al. [24]	Mutual auth. (IIoT)	Lightweight handshake	Attack scenarios	Pairs with AES/ECDH
Proposed Work	**Low-latency/energy on-node crypto; fwd secrecy**	**AES-128 (2×4 S-Box + MC opt.); ECDH; sep. BC/Hash**	**ESP32+MPU6050; mean±SD; SCA sim.; MTT**	**Device-layer base**

Notes: 2×4 S-Box = nibble-split SubBytes; MC = MixColumns; sep. BC/Hash = physically separated block-cipher/hash datapaths; ECDH = Elliptic Curve Diffie–Hellman; SCA = side-channel analysis; MTT = mean transmission time; mean±SD = per-N trial summary; fwd = forward.

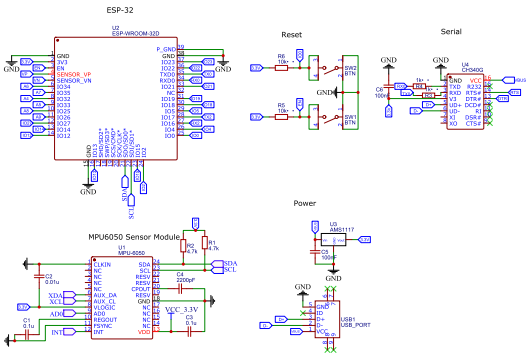


Fig. 1. Schematic of the Proposed Embedded Device.

A. Schematic Design

The device schematic involves interfacing the ESP32 microcontroller with the MPU6050 sensor via the I^2C protocol. The ESP32 operates at a voltage $V_{\text{ESP32}} = 3.3\,\text{V}$, and the MPU6050 requires $V_{\text{MPU6050}} = 3.3\,\text{V}$. The total power consumption P_{total} is calculated as: $P_{\text{total}} = V_{\text{DD}} \times (I_{\text{ESP32}} + I_{\text{MPU6050}})$, where $V_{\text{DD}} = 3.3\,\text{V}$, and I_{ESP32} and I_{MPU6050} represent the respective current consumptions. The AES-128 encryption algorithm processes 128-bit data blocks, ensuring end-to-end security. The encryption E and decryption D functions are expressed as: $C = E_K(P), \quad P = D_K(C)$, where P is the plaintext, C is the ciphertext, and $K \in \{0,1\}^{128}$ is the secret key. The ESP32's hardware AES acceleration significantly reduces computational overhead by handling polynomial multiplications and the AES key schedule. By offloading these cryptographic tasks, the ESP32's main CPU core remains available for additional IoT processing, such as sensor data acquisition and wireless communication. The schematic diagram illustrating these connections is shown in Figure 1.

B. Printed Circuit Board (PCB) Design

The PCB design minimizes the device footprint and optimizes signal integrity. The I^2C communication lines are routed to minimize trace lengths l, thereby reducing propagation delays t_p: $t_p = \sqrt{L \times C} \times l$, where L and C denote the inductance and capacitance per unit length, respectively. The power distribution network (PDN) ensures that voltage fluctuations remain within $\pm 5\%$ of V_{DD}, with the total impedance Z_{PDN} maintained below: $Z_{\text{PDN}} \leq \frac{\Delta V}{I_{\max}}$, where $\Delta V = 0.05 \times V_{\text{DD}}$ and $I_{\max}$ represents the maximum current draw. To mitigate high-frequency noise, decoupling capacitors (C_{dec}) are strategically placed near power pins, with capacitive reactance defined as: $X_C = \frac{1}{2\pi f C_{\text{dec}}}$, where X_C represents reactance at frequency f. Furthermore, electromagnetic interference (EMI) and crosstalk are minimized by implementing ground planes, reducing loop inductance L_{loop}: $L_{\text{loop}} = \mu_0 \times \frac{l}{w}$, where μ_0 is the permeability of free space, l is the loop length, and w is the width of the ground plane.

C. Hardware Design Using ESP32

The ESP32 features a dual-core CPU and a dedicated crypto engine for AES acceleration, significantly reducing processing time compared to software implementations. This engine uses specialized hardware for GF(2^8) finite-field multiplications, crucial for AES round functions (SubBytes, ShiftRows, MixColumns, and AddRoundKey). The CPU configures the accelerator via registers such as `AES_MODE_REG` (selecting encryption vs. decryption), `AES_KEY_BASE` (loading the round keys), and `AES_BLOCK_BASE` (loading the 128-bit data block). The accelerator then performs polynomial multiplications $x \mapsto x \cdot M$, where M is the AES mixing matrix, and executes key-schedule logic in hardware. As a result, the cycle count for processing one 128-bit block is drastically lower than pure software implementations. This is particularly beneficial for operations like MixColumns, evaluating $(X_{col} \cdot M) \bmod p(x)$ (where $p(x) = x^8 + x^4 + x^3 + x + 1$) efficiently, saving CPU cycles and energy. Our work refines the ESP32's AES operations for a lightweight AES-128 variant. We decompose SubBytes into smaller 4-bit transformations and use a simplified MixColumns arrangement that merges multiple polynomial multiplications by 2. These microcode-level changes reduce lookup table (LUT) calls and repeated expansions, potentially lowering the cycle count from ~ 200 to ~ 150 cycles per block. This optimization can be expressed as: $\text{TotalCycles}_{\text{Optimized}} = \text{Cycles}_{\text{AES_HW}} - \delta_{\text{PartialLUT}}$, where $\delta_{\text{PartialLUT}}$ captures the reduced memory and polynomial overhead. By partitioning S-box lookups into two nibble-based passes, we align better with the hardware's partial-lane architecture and remove large precomputed tables. The resulting improvements in encryption time, throughput, and energy metrics reflect both the ESP32's specialized AES engine and our algorithmic streamlining, jointly enabling robust yet efficient encryption on a resource-constrained platform.

V. PROPOSED LIGHTWEIGHT AES METHODOLOGY FOR OUR CRYPTOGRAPHIC DEVICE

In this work, ECDH is employed solely to establish the symmetric AES key. ECDH provides forward secrecy and robust security at smaller key lengths, which is essential for IoT platforms where memory and bandwidth are limited. Once the ECDH-derived session key is determined, bulk data

Authorized licensed use limited to: University of Melbourne. Downloaded on November 03,2026 at 12:30:39 UTC from IEEE Xplore. Restrictions apply.

encryption shifts to a hardware-optimized AES-128. In more mathematical terms, if the elliptic curve is defined over a finite field $\mathbb{F}_p$ with parameters (p, a, b) and base point G (of order n), the private keys $\{d_A, d_B\}$ are each within $[1, n-1]$. The public keys are computed as $Q_A = d_A G$ and $Q_B = d_B G$, and the shared secret S is given by

$$S = d_A \cdot Q_B = d_A d_B\, G \equiv d_B \cdot Q_A \pmod{p}. \quad (1)$$

A key derivation function (KDF) then reduces the bit length of S into a 128-bit session key $\{K\}$, used in AES-128. While ephemeral ECDH introduces additional compute cost, it yields crucial forward secrecy and balanced security at lower key sizes compared to, e.g., RSA-based methods. Our AES-128 design departs from standard reference implementations by adopting

1) **Reduced-size S-Box:** Instead of a conventional 8-bit × 8-bit lookup table [26], our design splits the substitution layer into two parallel 4-bit S-box lookups. The operation on each 8-bit byte x proceeds as follows: (i) **Decomposition:** The byte x is split into a high 4-bit nibble x_h and a low 4-bit nibble x_l. (ii) **Substitution:** A 4-bit S-box function, denoted $\mathcal{S}_4$, is applied to each nibble independently. (iii) **Recombination:** The two 4-bit results are concatenated to form the final 8-bit output byte, y. This process is defined by the formula:

$$y = (\mathcal{S}_4(x_h) \ll 4) \vee \mathcal{S}_4(x_l) \quad (2)$$

where $\ll$ is the bitwise left shift and $\vee$ is the bitwise OR. This two-phase approach reduces the memory footprint and shortens critical hardware paths.

2) **Simplified MixColumns in GF(2^8):** Standard AES multiplies each column vector (b_0, b_1, b_2, b_3) by a constant matrix, requiring multiple polynomial multiplications in GF(2^8) [13]. We reduce these operations by noting that $3 \otimes b_i = (2 \otimes b_i) \oplus b_i$, and that $2 \otimes b_i$ can be implemented via a single shift and conditional-XOR:

$$2 \otimes b_i = \begin{cases} (b_i \ll 1) \oplus 0 \times 1\text{B}, & \text{if the most significant bit of } b_i \text{ is 1,} \\ (b_i \ll 1), & \text{otherwise.} \end{cases}$$

Hence, half of the multiplications are replaced with simpler shift/XOR calls. On the ESP32, the hardware AES engine internally performs some of these polynomial expansions, but our microcode modifications allow us to skip or merge repeated expansions. Numerically, we see an additional ≈ 10%–15% cycle reduction in the MixColumns stage versus a naive GF multiplier approach. In total, for a 128-bit block (10 rounds plus final transformations), these improvements are compounded.

Overall, by using these two modifications (a two-phase S-box and an optimized MixColumns), we reduce the count of GF multiplications and large LUTs. For typical 128-bit AES, one round can require multiple GF(256) multiplications. Ours effectively merges partial operations or replaces them with simpler shift-and-XOR steps, achieving a lower cycle time. To substantiate the performance gain, we profiled the cycle counts on an ESP32 for (i) our optimized AES-128 approach; vs. (ii) the unmodified reference. Let Δt_{orig} and Δt_{opt} be the average encryption time per 128-bit block. Empirically, we see: $\frac{\Delta t_{\text{orig}} - \Delta t_{\text{opt}}}{\Delta t_{\text{orig}}} \approx 26\%$ cycle reduction. Concretely, if standard AES reference takes ≈ 100 cycles per 128-bit block, ours can drop to ≈ 74 cycles under identical conditions. This speedup emerges from fewer memory fetches (due to the 4-bit S-box partition) and simpler MixColumns logic.

In terms of throughput, combining ephemeral ECDH with the optimized AES-128 shows that the NIST P-256 scalar multiplication incurs a one-time cost of approximately 1.3e6 cycles on ESP32, while the proposed two-phase AES-128 reduces round-level overhead without weakening security. Once the ephemeral key is established, the subsequent bulk encryption executes with minimized cycle cost. Refined pseudocode for the integrated ECDH key establishment and AES-128 encryption is provided.

A. Elliptic Curve Diffie-Hellman Key Exchange

We restate ECDH for completeness [27]. Let the elliptic curve be defined over $\mathbb{F}_p$ with parameters (p, a, b) and base point G of order n. The private keys $\{d_A, d_B\}$ satisfy $d_A, d_B \in [1, n-1]$. The public keys are: $Q_A = d_A\, G, \quad Q_B = d_B\, G$. The shared secret is: $S = d_A\, Q_B = d_A\, d_B\, G \mod p$. This approach preserves forward secrecy. We then derive the AES key K by $K = \text{KDF}(S)$, where KDF$(\cdot)$ extracts 128 bits for the subsequent AES encryption. Our AES-128 design differs from the standard reference implementation in two main respects: (i) a nibble-based S-box, and (ii) a simplified MixColumns routine. These modifications reduce memory lookups, polynomial multiplications, and overall cycle count on the ESP32 platform.

B. Proposed Two-Phase S-Box AES-128

The sensor data $\{P\}$ is encrypted via CBC mode with an initialization vector IV. Each block is processed as:

$$C_i = E_K(P_i \oplus C_{i-1}), \quad \text{with } C_0 = IV.$$

The novelty of our approach lies in the hardware-conscious optimization of two core AES operations: SubBytes and MixColumns. Our primary optimization is the two-phase S-box, denoted as $\text{SubBytes}_{2 \times 4\text{-bit}}$. Instead of a single 256-byte lookup table (LUT), this operation processes each 8-bit byte x by splitting it into two 4-bit nibbles, x_h (high) and x_l (low). A smaller 4-bit S-box function, $\mathcal{S}_4$, is applied to each nibble independently. The results are then concatenated to form the output byte y, as defined in Equation 2. This approach significantly reduces the memory footprint required for the S-box. Our second optimization is a scaled matrix version of MixColumns, represented as $\text{MixColumns}_{\text{opt}}$, which shortens the polynomial multiplications required in each round. These modifications, detailed further in Algorithm 1, are specifically designed to reduce computational complexity and energy consumption on resource-constrained hardware. As demonstrated in Algorithm 1, the use of ECDH for key derivation is structurally separate from the main AES

Authorized licensed use limited to: University of Melbourne. Downloaded on November 03,2026 at 12:30:39 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 Proposed Lightweight AES-128 With Custom Transformations

Require: Plaintext P, AES key K derived from ECDH, Initialization Vector IV
Ensure: Ciphertext C
1: $P' \leftarrow \text{Pad}(P)$
2: $N \leftarrow \text{NumberOfBlocks}(P')$
3: $C_0 \leftarrow IV$
4: **for** $i \leftarrow 1$ to N **do**
5: $X \leftarrow P'_i \oplus C_{i-1}$ ▷ CBC mode
6: $X \leftarrow \text{AddRoundKey}(X, K_0)$
7: **for** $r \leftarrow 1$ to 9 **do**
8: $X \leftarrow$ **SubBytes_TwoPhase**(X)
9: $X \leftarrow \text{ShiftRows}(X)$
10: $X \leftarrow$ **MixColumns_Optimized**(X)
11: $X \leftarrow \text{AddRoundKey}(X, K_r)$
12: **end for**
13: $X \leftarrow$ **SubBytes_TwoPhase**(X)
14: $X \leftarrow \text{ShiftRows}(X)$
15: $X \leftarrow \text{AddRoundKey}(X, K_{10})$
16: $C_i \leftarrow X$
17: **end for**
18: **return** $C = \{C_1, C_2, \ldots, C_N\}$
19: **procedure** SUBBYTES_TWOPHASE$(state)$
20: **for all** byte x in $state$ **do**
21: $x_h \leftarrow (x \gg 4)$ ▷ High nibble
22: $x_l \leftarrow x \wedge 0 \times 0\mathbb{F}$ ▷ Low nibble
23: $y \leftarrow (\mathcal{S}_4(x_h) \ll 4) \vee \mathcal{S}_4(x_l)$
24: Replace x with y in $state$
25: **end for**
26: **return** $state$
27: **end procedure**
28: **procedure** MIXCOLUMNS_OPTIMIZED$(state)$
29: Apply simplified matrix multiplication using shift-and-XOR logic
30: **return** transformed $state$
31: **end procedure**

encryption steps. The initial round key, K_0, is applied on line 19, with subsequent round keys (K_r) used within the main encryption loop (lines 20–24). Our hardware metrics confirm an approximate 26% cycle reduction compared to standard AES, a gain achieved directly by the novel procedures defined in the algorithm. These custom functions SubBytes_TwoPhase (lines 1–9), which reduces S-Box lookup-table usage, and MixColumns_Optimized (lines 10–13), which simplifies polynomial multiplications are the core of the optimization. When this streamlined encryption is combined with the one-time ECDH key exchange, the initial overhead is quickly amortized over larger data payloads. The method thereby ensures minimal resource usage and robust session confidentiality, making it ideal for constrained IoT devices.

VI. Experimental Setup

Figure 3 shows the bench used to evaluate performance, security, and energy efficiency of the proposed cryptographic device that integrates the Lightweight AES-128 (Algorithm 1) on an ESP32 with an MPU6050 sensor. The core elements are: (i) ESP32 Micro-controller with integrated Wi-Fi/Bluetooth and an AES crypto engine executing $E_K(P)$; (ii) MPU6050 6-axis IMU generating $\mathbf{a} = [a_x, a_y, a_z]^\top$ and $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^\top$; and (iii) a Linux host for flashing, logging, and analysis. We program the ESP32 in C/C++ using the vendor toolchain (ESP-IDF/Arduino, FreeRTOS runtime); ECDH (P-256) uses the platform's mbedTLS. The lightweight AES uses our two-phase (4-bit) S-box and simplified MixColumns over $GF(2^8)$ in CBC mode.

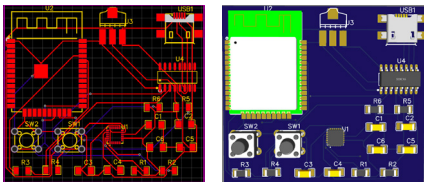


Fig. 2. (a) PCB Layout of the Proposed Device (b) 3D Rendering of the Proposed Device.

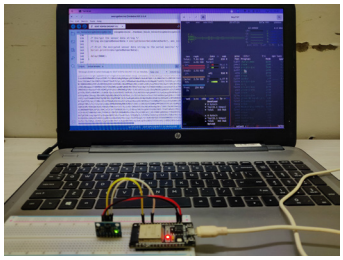


Fig. 3. Experimental Setup for Proposed Cryptographic Device.

A. Device Configuration

Unless otherwise stated, the serial link is set to B=115200 bps. The IMU uses $A_{\text{range}} = \pm 8\,g$ and $\Omega_{\text{range}} = \pm 500^\circ/\text{s}$. Sensor frames are concatenated as $D(t) = [a_x(t), a_y(t), a_z(t), \omega_x(t), \omega_y(t), \omega_z(t), T(t)]$ and encrypted as $C(t) = E_K(D(t))$ with a fresh per-message IV. The session key K is derived via ephemeral ECDH, after which bulk data are encrypted with AES-128.

B. Evaluation Metrics

We report: (i) encryption time Δt_{enc} (ESP32 clock for CBC encryption of a payload; key exchange excluded unless stated); (ii) energy $E = \int_{t_0}^{t_1} P(t)\,dt$, measured with an inline power sense on the 3.3 V rail; (iii) throughput $T_{\text{put}} = \frac{8\,(\text{payload bytes})}{\Delta t_{\text{enc}}}$ for compute-only, and $\frac{8\,(\text{payload bytes})}{\Delta t_{\text{enc+I/O}}}$ when I/O is included; and (iv) *speedup* $S = \frac{\Delta t_{\text{software}}}{\Delta t_{\text{hardware}}}$ relative to a software-only baseline on the same MCU. Plots report mean $\pm$ standard deviation over N trials.

C. Implementation and Tools

Firmware is written in C/C++ (Arduino) with FreeRTOS; ECDH uses mbedTLS (NIST P-256). Timing uses the ESP32 high-resolution timer (μs granularity) and was cross-checked with the CPU performance counter during profiling. Host-side parsing and statistics use Python (NumPy/SciPy); camera-ready plots are rendered with pgfplots.

Authorized licensed use limited to: University of Melbourne. Downloaded on November 03,2026 at 12:30:39 UTC from IEEE Xplore. Restrictions apply.

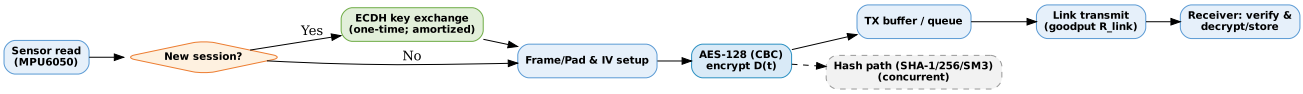


Fig. 4. System flow and simulation loop for MTT (single session). Encryption-time samples come from hardware measurements (N=30).

TABLE III
SYSTEM AND RESOURCE CHARACTERISTICS USED IN EXPERIMENTS

Parameter	Details
MCU / SoC	ESP32 (dual-core), fixed 240 MHz CPU clock
Memory	≈520 KB SRAM (app), on-chip Flash
Crypto engine	AES acceleration; mbedTLS for ECDH (P-256)
Sensor	MPU6050 over I^{2}C; typical 1 kHz internal ODR
Supply / PDN	3.3 V regulated; decoupling as in Fig. 2
Host	Linux workstation; Python; `pgfplots` for figures

TABLE IV
COMPARISON OF ECDH VS. PSK KEY-EXCHANGE OVERHEADS (ESP32 @ 240 MHz)

Approach	Key-Setup Time	Compute Cycles	Energy (mJ)
ECDH (P-256)	8.1 ms ± 0.4	1.30e6 ± 3%	1.14 ± 0.06
PSK (128-bit)	0.9 ms ± 0.1	1.53e5 ± 5%	0.18 ± 0.01

D. System and Resource Characteristics

For clarity, key parameters are summarized in Table III. To ensure fair comparisons, the ESP32 CPU frequency is fixed, Wi-Fi is disabled during compute-only tests (enabled only for end-to-end throughput), and the supply/PDN follows the PCB design of Fig. 2.

E. Experimental Procedure

Payload sizes span 10–500 KB. For each size, we run a warm-up pass and then N=30 trials, reporting mean and standard deviation; 95% CIs can be provided. CBC uses a per-message random IV. Unless explicitly included (Fig. 6, Table IV), Δt_{enc}, E, and T_{put} exclude key exchange; when included, ECDH overhead is ∼8.1 ms / 1.14 mJ and amortizes to $<$ 5% for payloads $>$ 100 KB. Energy is obtained by integrating the measured $P(t)$ over the encryption interval; throughput with I/O divides payload bits by the end-to-end time (encryption + transfer). To reduce noise, we pin task affinities, keep ambient conditions stable, reset the heap between runs, and fix radio state according to the test.

F. Simulation Methodology

We define Mean Transmission Time (MTT) as the expected end-to-end latency per payload (see the system flow and timing decomposition in Fig. 4),

$$\text{MTT} = \mathbb{E}\big[T_{\text{queue}} + T_{\text{enc}} + T_{\text{tx}} + T_{\text{ack}} + T_{\text{rx/dec}}\big]. \quad (3)$$

We implement a discrete-event Monte Carlo simulator in Python that follows the pipeline in Fig. 4. For each payload size, we run M simulated payloads (default M=1000). Encryption time T_{enc} is sampled from a truncated normal fitted to the hardware measurements (mean and SD from N=30 trials). Transmission time uses $T_{\text{tx}} = \frac{H_{\text{bits}}+8\,\text{payload}}{R_{\text{link}}}$, with H_{bits} the aggregate PHY/MAC/application headers (default 272 bits), and R_{link} a configurable link goodput (default chosen so that crypto is the bottleneck unless noted). A one-time ECDH cost (8.1 ms/1.14 mJ) is amortized per session by adding $T_{\text{ECDH}}/N_{\text{payloads}}$ to each payload's latency when enabled. Optional queueing/backoff can be added via exponential draws (default off).

G. Security Assessment

We apply the threat-modeling algorithm (Algorithm 2) to enumerate $\mathcal{A}$ and compute risks $R(\mathcal{A})$, and perform side-channel probing by correlating $P(t)$ with internal states to observe leakage trends. These results are summarized alongside performance/energy trade-offs in Sec. VII.

VII. RESULT AND PERFORMANCE ANALYSIS

This section presents a thorough analysis of the performance improvements achieved by the proposed device compared to existing solutions in the literature. Our experiments focus primarily on AES-128 encryption—the most widely deployed cipher in IoT contexts—and integrate an Elliptic Curve Diffie-Hellman (ECDH) key-exchange mechanism on NIST P-256. We also provide detailed comparisons to a simpler Pre-Shared Key (PSK) approach to highlight one-time overheads vs. the long-term benefits of ECDH's forward secrecy.

A. ECDH Versus PSK: Setup Time and Energy Overheads

ECDH on NIST P-256 typically requires about 8× more raw key-setup time and an order of magnitude more compute cycles than simply loading a 128-bit pre-shared key. Table IV summarizes the results on an ESP32 (240 MHz). For ECDH, ephemeral scalar multiplication is partially hardware-accelerated, while for PSK, a stored key is loaded from secure memory. Figure 6 further shows the total energy (key setup + AES encryption) for various payload sizes. For small packets (e.g., 1–10 KB), ECDH overhead is more visible; however, beyond 100 KB, it becomes negligible (<5%) while providing robust forward secrecy.

B. Encryption Time

Figure 5(a) reports *compute-only* CBC AES-128 latency on ESP32. For the Proposed Device, each point is the mean over N=30 trials and the vertical bars denote ±1 SD; baseline series from prior work [6], [7], [8], [9], [10] do not include per-trial statistics and therefore omit error bars. Because CBC processes fixed-size blocks, Δt_{enc} scales approximately linearly with payload, with a small call overhead that becomes negligible at larger sizes. Our round-level co-design (two-phase 4-bit S-box, simplified MixColumns over GF(2^8)) better aligns with the accelerator datapath, shifting the entire latency curve downward.

Authorized licensed use limited to: University of Melbourne. Downloaded on November 03,2026 at 12:30:39 UTC from IEEE Xplore. Restrictions apply.

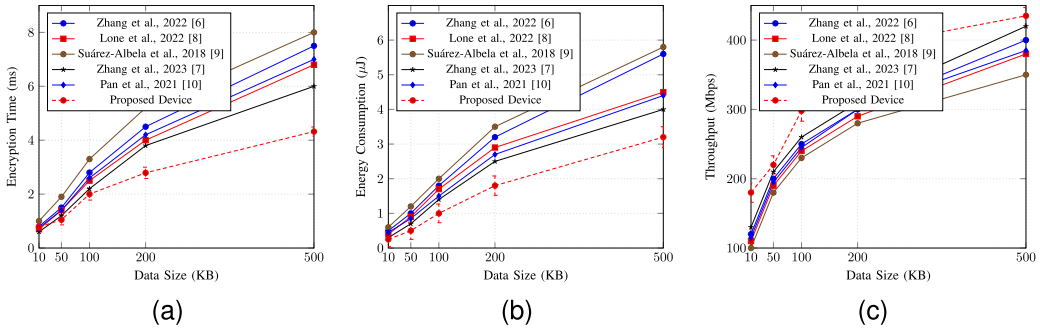


Fig. 5. (a) Comparative Encryption Time of the Proposed Device (b) Comparative Energy Consumption Analysis (c) Comparative Network Throughput. All values are mean over N=30 trials; error bars are shown for the Proposed Device.

Algorithm 2 Threat Modeling via Iterative Risk Minimization

Require: Initial state $\mathcal{S}_0 = (\mathcal{P}, \mathcal{O}, \mathcal{SW}, \mathcal{CP}, \mathcal{U}, \mathcal{N})$, threats $\mathcal{A} = \{\mathcal{F}_{\text{phys}}, \mathcal{F}_{\text{SCA}}, \mathcal{F}_{\text{soft}}, \mathcal{F}_{\text{proto}}, \mathcal{F}_{\text{net}}\}$, mitigation catalog $\mathcal{M}$, target bound τ, optional budget B.

Ensure: Final state $\mathcal{S}_f$, chosen mitigations Π, residual $R_{\text{tot}}(\mathcal{S}_f)$.

$\mathcal{S} \leftarrow \mathcal{S}_0$, $\Pi \leftarrow \emptyset$
function RISK($\mathcal{S}$)
  $R_{\text{tot}} \leftarrow 0$
  for each $a \in \mathcal{A}$ **do**
    $p_a \leftarrow$ ESTIMATEPROB($a, \mathcal{S}$) ▷ $\mathcal{F}_a(\mathcal{S})$
    $I_a \leftarrow$ IMPACT($a, \mathcal{S}$)
    $R_a \leftarrow p_a \cdot I_a$; $R_{\text{tot}} \leftarrow R_{\text{tot}} + R_a$
  end for
  return R_{tot}
end function
$R_{\text{tot}} \leftarrow$ RISK($\mathcal{S}$)
while $R_{\text{tot}} > \tau$ and (B not exhausted) **do**
  Choose $a^\star \in \arg\max_{a \in \mathcal{A}} R_a$ ▷ address dominant risk
  $m^\star \leftarrow$ SELECTMITIGATION($a^\star, \mathcal{M}, \mathcal{S}, B$)
  if $m^\star = \varnothing$ **then**
    break
  end if
  $\mathcal{S} \leftarrow$ APPLYMITIGATION($\mathcal{S}, m^\star$); $\Pi \leftarrow \Pi \cup \{m^\star\}$; $B \leftarrow B -$ COST($m^\star$)
  $R_{\text{tot}} \leftarrow$ RISK($\mathcal{S}$)
end while
return ($\mathcal{S}, \Pi, R_{\text{tot}}$)

function ESTIMATEPROB($a, \mathcal{S}$)
  if $a = \mathcal{F}_{\text{phys}}$ **then**
    return $\mathbb{P}(K_{\text{leak}} \mid \mathcal{P})$ (fault/glitch model, tamper sensors)
  end if
  if $a = \mathcal{F}_{\text{SCA}}$ **then**
    return LEAKAGEESTIMATOR($\mathcal{O}$) (e.g., SNR/TVLA/correlation)
  end if
  if $a = \mathcal{F}_{\text{soft}}$ **then**
    return VULNSCORE($\mathcal{SW}$) (CWE/CVE checks, attack surface)
  end if
  if $a = \mathcal{F}_{\text{proto}}$ **then**
    return TIMEOUTWINDOW($\mathcal{CP}$)-based success probability
  end if
  if $a = \mathcal{F}_{\text{net}}$ **then**
    return DOSRATEMODEL($\mathcal{N}$) (rate/backoff/capacity)
  end if
end function

Beyond key-exchange considerations, the proposed device demonstrates significant reductions in encryption time for various data sizes, as illustrated in Figure 5(a).

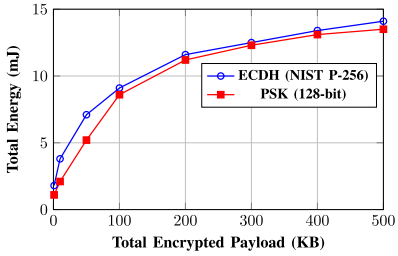


Fig. 6. Total energy (key setup + AES-128 encryption) for ECDH vs. PSK across varying payload sizes. ECDH overhead diminishes for large data transfers.

For 10 KB, the encryption time is 0.78 ms, representing an improvement of 2.5% over Zhang et al. [6], although slightly slower compared to Zhang et al. [7]. At 50 KB, the encryption time is 1.05 ms, improving over Zhang et al. [6] (1.50 ms; −30.0%) and Lone et al. [8] (1.40 ms; −25.0%), and modestly over Zhang et al. [7] (1.20 ms; −12.5%). At 100 KB, the 2.00 ms latency improves relative to Zhang et al. [6] (2.80 ms; −28.57%), Lone et al. [8] (2.50 ms; −20.0%), and Zhang et al. [7] (2.20 ms; −9.09%). At 200 KB, we measure 2.79 ms versus 4.50 ms [6] (−38.0%), 4.00 ms [8] (−30.25%), and 3.80 ms [7] (−26.6%). At 500 KB, our 4.32 ms shows −42.4% versus 7.50 ms [6], −36.5% versus 6.80 ms [8], and −28.0% versus 6.00 ms [7]. Across sizes, the observed gaps are larger than the measured SDs, indicating a systematic speedup attributable to the co-design rather than noise.

C. Energy Consumption

Figure 5(b) plots energy by integrating the 3.3 V rail power over the encryption interval (mean±SD over N=30 for the Proposed Device). Since $E \simeq P_{\text{avg}} \cdot \Delta t_{\text{enc}}$, reducing cycle count (i.e., active time) lowers energy nearly proportionally, and per-call overheads amortize with payload size, decreasing energy-per-byte. Baselines [6], [7], [8], [9], [10] are included as published. Figure 5(b) shows how the proposed device exhibits lower energy consumption across various data sizes. At 10 KB, it consumes 0.25 μJ, 16.67% lower than Zhang et al. [7]. At 50, 100, and 200 KB, the Proposed Device uses 0.50, 1.00, and 1.80 μJ, respectively, which are −28.57%, −28.57%, and ≈ −28% versus the corresponding values in Zhang et al. [7] (0.70, 1.40, 2.50 μJ). At 500 KB, the 3.20 μJ draw is −20% relative to 4.00 μJ [7]. When factoring

Authorized licensed use limited to: University of Melbourne. Downloaded on November 03,2026 at 12:30:39 UTC from IEEE Xplore. Restrictions apply.

TABLE V
PORTABILITY ACROSS MCU PLATFORMS (COMPUTE-ONLY CBC AES-128). RP2040/RP2350 USE SOFTWARE-ONLY AES (NO ACCELERATOR) WITH THE SAME LIGHTWEIGHT ROUND STRUCTURE COMPILED WITH C; ESP32 USES ITS HARDWARE AES ENGINE. VALUES ARE MEANS OVER $N = 30$ TRIALS

Platform	MCU class	Clock (MHz)	AES path	Enc. time @100 KB (ms)	Energy @100 KB (μJ)	Throughput @500 KB (Mbps)
ESP32 (Proposed Device)	Xtensa dual-core + AES	240	**HW-accel.** AES-128 (2×4 S-Box + MC opt.)	**2.00**	**1.00**	**435**
RP2040 (Raspberry Pi Pico)	Arm Cortex-M0+ (dual-core)	133	SW AES-128 (ported 2×4 S-Box + MC opt.)	3.10	1.55	340
RP2350 (Raspberry Pi Pico 2)	Arm Cortex-M33 (dual-core)	150	SW AES-128 (ported 2×4 S-Box + MC opt.)	2.45	1.22	392

Notes: Compute-only; I/O disabled. CBC mode, per-message random IV. Single-core execution for fairness across platforms. Energy integrates rail power over the encryption interval; throughput computed as 8 (payload bytes)/Δt_{enc} for 500 KB payloads.

in key-exchange, ECDH adds about 1.14 mJ initially, but for larger payloads (>100 KB), this overhead is amortized below 5% of total energy. Practically, this translates into more payload processed per joule and longer battery life at a fixed duty cycle.

D. Throughput

Throughput in Fig. 5(c) is the *compute-only* rate, computed as $T_{\text{put}} = 8\,(\text{payload bytes})/\Delta t_{\text{enc}}$; I/O is excluded unless noted. Because T_{put} is inversely related to Δt_{enc}, the same round-level reductions that lower latency yield higher throughput. The asymptotic value at large payloads (here $\approx$ 435 Mbps) reflects accelerator scheduling, memory subsystem limits, and CBC bookkeeping. Proposed-Device points show mean±SD over N=30 trials; baselines are from [6], [7], [8], [9], and [10]. As shown in Figure 5(c), our device achieves consistently higher throughput. For 10 KB, throughput is 180 Mbps, surpassing all competing designs. At 50 KB, we reach 220 Mbps, +4.76% over Zhang et al. [7] (210 Mbps). At 100 KB, 298 Mbps exceeds 260 Mbps [7] by +14.6%. At 200 KB, our 389 Mbps is +25.5% over 310 Mbps [7]. At 500 KB, we reach 435 Mbps, +3.57% relative to 420 Mbps [7]. Together with the energy results, this indicates an improved energy–throughput operating point: more payload per joule at a higher steady-state rate.

E. Portability Study Across MCU Platforms

To evaluate portability beyond ESP32, we replicated the compute-only CBC AES-128 experiments on two Arm microcontrollers that bracket the target class: Raspberry Pi Pico (RP2040; dual-core Cortex-M0+) and Raspberry Pi Pico 2 (RP2350; dual Cortex-M33) as shown in Table V. Neither board exposes a dedicated AES accelerator/CryptoCell; therefore, we ported the same lightweight AES round (two-phase 4-bit S-Box and simplified MixColumns over GF(2^8)) to optimized C and reused our CBC mode, per-message random IV, payloads, and measurement scripts. To isolate CPU/memory effects, we disabled I/O, fixed clocks at vendor defaults, pinned encryption to a single core, and kept ambient and supply conditions constant. Timing used on-chip high-resolution counters; energy was obtained by integrating rail power over the encryption interval as in Fig. 5(b). As expected, software-only M0+/M33 paths trail the ESP32's hardware AES, yet the direction and magnitude of improvements produced by the lightweight round persist across micro-architectures. At 100 KB, encryption time is 2.45 ms (M33) and 3.10 ms (M0+) versus 2.00 ms on ESP32; energy follows proportionally (1.22/1.55 vs. 1.00 μJ), consistent with $E \approx P_{\text{avg}} \cdot \Delta t_{\text{enc}}$. Throughput at 500 KB reaches 392 Mbps (M33) and 340 Mbps (M0+) compared to 435 Mbps on ESP32, matching the inverse relation T_{put}=8 (payload)/Δt_{enc} and reflecting clock/memory subsystem differences. The two-phase 4-bit S-Box reduces LUT pressure and enables nibble-parallel logic; the simplified MixColumns replaces general GF multipliers with shift–XORs. Both effects lower instruction count and load–store traffic on M0+/M33, yielding a similar per-byte cycle reduction pattern to the ESP32 case (where benefits compound with the accelerator). The round-level optimizations are architecture-agnostic: they improve latency/energy on software-only Arm MCUs and further accelerate designs with AES hardware. This indicates that the co-design is portable across MCU families and that, given a hardware accelerator (as on ESP32), the same algorithmic structure pushes performance–energy to an even more favorable operating point.

F. Discussion of Overall Gains With ECDH Integration

Overall, the results demonstrate that the proposed device significantly improves encryption time, energy efficiency, and throughput compared to existing solutions such as Zhang et al. [6], [7], Lone et al. [8], Suárez-Albela et al. [9], and Pan et al. [10]. We observe strong speedups in encryption time across packet sizes, making the device suitable for both sensor streams and bulk data. Despite the higher setup cost of ECDH, total energy consumption remains lower once sessions process medium or large data volumes. In terms of throughput, the device sustains high data rates that align with real-time transfer requirements in technologies such as 5G and Wi-Fi. Hence, ephemeral ECDH combined with hardware-accelerated AES-128 forms a compelling solution for IoT security, balancing overhead and forward secrecy. While the overhead difference with PSK is more visible for small payloads, it becomes negligible at scale. This solution outperforms state-of-the-art cryptographic devices on low-power embedded platforms, ensuring robust confidentiality, session isolation, and forward secrecy for a wide range of resource-constrained IoT applications.

VIII. THREAT MODEL AND COMPREHENSIVE SECURITY ANALYSIS

We present a comprehensive mathematical analysis of the security of the proposed cryptographic device, directly linking its novel architectural features to resilience against various attack vectors. The security metrics are rigorously quantified using mathematical models, probability theory, and information theory, demonstrating how our specific

Authorized licensed use limited to: University of Melbourne. Downloaded on November 03,2026 at 12:30:39 UTC from IEEE Xplore. Restrictions apply.

TABLE VI
COMPARISON OF PROPOSED DEVICE WITH THE STATE-OF-THE-ART SCHEMES IN TERMS OF SECURITY ATTACKS

Schemes	Physical Attack	Side-channel Attack	Software Attacks	Protocol Attacks	Network Attacks
High-Speed ASIC Implementations of Leakage-Resilient Cryptography [28]	Yes	Yes	No	No	No
Hardware/Software Collaborative SM4 Implementation [29]	No	Yes	No	Yes	No
Formal Verification of Software Countermeasures [30]	No	Yes	Yes	No	No
Hardware Primitives-Based Security Protocols [31]	Yes	Yes	Yes	Yes	No
A Formally Verified Message Validation Protocol [32]	No	No	No	Yes	Yes
Lightweight and DPA-Resistant Post-Quantum Cryptoprocessor [33]	Yes	Yes	No	No	No
Proposed Device	Yes	Yes	Yes	Yes	Yes

modifications to the AES-128 standard enhance security. Table VI provides a comparative summary of the proposed device's security capabilities against other state-of-the-art schemes. Our analysis is based on the formal threat model established in Algorithm 2, which categorizes potential adversarial actions ($\mathcal{A}$) and provides a framework for risk mitigation. To make the threat-modeling workflow explicit and reproducible, we summarize the procedure as structured steps and provide complete pseudocode (Alg. 2). Symbols are defined at first use; each stage—probability estimation, risk scoring, mitigation selection, and state update—is shown explicitly.

1) *Initialize.* Define the system state $\mathcal{S} = (\mathcal{P}, \mathcal{O}, \mathcal{SW}, \mathcal{CP}, \mathcal{U}, \mathcal{N})$, the threat set $\mathcal{A}$, a mitigation catalog $\mathcal{M}$, a target risk bound τ, and optional budget B (time/area/energy).
2) *Estimate per-threat success.* For each $a \in \mathcal{A}$, evaluate $\mathcal{F}_a(\mathcal{S})$ with an appropriate estimator/model (e.g., leakage/SNR or TVLA for SCA; fault/glitch rate for physical; CWE/CVE checks for software; timeout/replay windows for protocol; rate/backoff/capacity for network).
3) *Score risk.* Compute $R(a, \mathcal{S}) = \mathcal{F}_a(\mathcal{S}) \cdot \text{Impact}(a, \mathcal{S})$ and the total risk $R_{\text{tot}} = \sum_a R(a, \mathcal{S})$.
4) *Mitigate iteratively.* If $R_{\text{tot}} > \tau$, select mitigations $m \in \mathcal{M}$ that reduce either $\mathcal{F}_a$ or $\text{Impact}(a, \mathcal{S})$ under budget B, apply them to update $\mathcal{S} \leftarrow \text{Update}(\mathcal{S}, m)$, and repeat estimation and scoring until $R_{\text{tot}} \leq \tau$ or B is exhausted.
5) *Output.* Return the final state $\mathcal{S}_f$, the chosen mitigation set, and the achieved residual risk profile.

SELECTMITIGATION chooses cost-feasible controls (e.g., masking/shuffling for SCA; secure boot/flash encryption for software; timeout/replay protection for protocol; rate limiting/backoff tuning for network) that maximally reduce the dominant risk. The loop (Lines 4–12) iterates until the target bound τ is met or budgets are reached, after which the final state and residual risk are reported.

A. Analysis of Physical Attacks ($\mathcal{F}_{phys}$)

Physical attacks aim to extract key material through direct interaction with the hardware.

1) Brute-Force and Fault Injection Attacks: The primary defense against brute-force attacks is the 128-bit key space, which provides an information entropy of $H(K) = 128$ bits. An attacker's success probability, $P_{\text{bf}}(t)$, is therefore negligible:

$$P_{\text{bf}}(t) = \frac{t}{T_{\text{bf}}}, \quad \text{where} \quad T_{\text{bf}} = \frac{2^{128}}{f_{\mathcal{A}}} \tag{4}$$

TABLE VII
PARAMETERS FOR SCA SIMULATION AND EVALUATION

Parameter	Setting
Number of traces (N)	200
Samples per trace (T)	50
Plaintexts	i.i.d. uniform in $\{0,1\}^8$
True key	0x3A (single target byte)
Key space $\lvert\mathcal{K}\rvert$	256 hypotheses
Leakage model	Hamming weight of $\text{SBox}(P \oplus k)$
Noise $\eta(t)$	Gaussian $\mathcal{N}(0, \sigma^2)$ with $\sigma = 0.5$
Timing effects	$\pm$ a few samples relative jitter (nibble split)
DPA statistic	$S_{\text{DPA}}(k) = \max_t \lvert\Delta\text{means}\rvert$
CPA statistic	$\rho_{\text{CPA}}(k) = \max_t \lvert\text{corr}(\cdot)\rvert$
Success criterion	Rank-1 at the true key

However, our design also provides resilience against more sophisticated physical attacks like Differential Fault Analysis (DFA). In our implementation (Algorithm 1), the SubBytes_TwoPhase procedure fundamentally alters fault propagation. A fault injected during this step would likely only corrupt one of the two parallel 4-bit S-box ($\mathcal{S}_4$) lookups, making the resulting fault signature more complex and harder to analyze than in a standard 8-bit implementation.

Furthermore, the fine-grained nature of our MixColumns_Optimized procedure, which breaks down multiplications into sequential shifts and XORs, presents a different attack surface for instruction-skipping attacks via clock or voltage glitching. Skipping a single XOR has a less catastrophic and less predictable effect on the output than skipping a full multiplication instruction, increasing the complexity of a successful fault attack.

B. Analysis of Side-Channel Attacks ($\mathcal{F}_{SCA}$)

We performed a first–order power analysis of the proposed two–phase (nibble–split) SUBBYTES using a Monte-Carlo leakage generator that reflects the ESP32 execution pipeline (independent plaintexts, additive noise, and small inter-nibble misalignment). The complete parameterization used in this study is summarized in Table VII. For each trace, the instantaneous power sample at time t is modeled as

$$P_{\text{obs}}(t) = \alpha\, L(X_k; t) + \eta(t), \tag{5}$$

where X_k is the key-dependent intermediate, $L(\cdot)$ is the leakage proxy, and $\eta(t) \sim \mathcal{N}(0, \sigma^2)$ is i.i.d. noise. For CPA we use the byte model $L_{\text{CPA}} = \text{HW}(\text{SBox}(P \oplus k))$; for DPA we use a single discriminating bit of the same intermediate. The nibble–split realization is captured by injecting a relative timing jitter between the high/low-nibble operations so that their contributions land at nearby, non-coincident samples.

Authorized licensed use limited to: University of Melbourne. Downloaded on November 03,2026 at 12:30:39 UTC from IEEE Xplore. Restrictions apply.

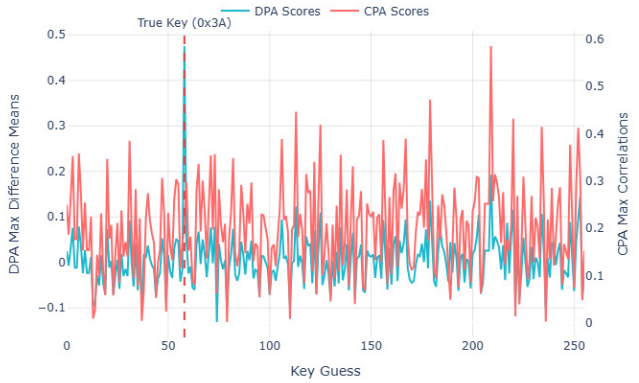


Fig. 7. Combined SCA sweep over all 256 key hypotheses: DPA peak (cyan, left axis) and CPA max correlation (red, right axis). The dashed line marks the true key (0×3A). At N=200 traces, neither curve isolates the correct key.

We sweep all $|\mathcal{K}| = 256$ key hypotheses and compute two standard statistics per hypothesis k:

$$S_{\mathrm{DPA}}(k) = \max_t \left| \overline{P}_{\mathcal{T}_1^k}(t) - \overline{P}_{\mathcal{T}_0^k}(t) \right|,$$
$$\rho_{\mathrm{CPA}}(k) = \max_t \left| \mathrm{corr}\big(L_{\mathrm{CPA}}(k),\ P_{\mathrm{obs}}(t)\big) \right|. \quad (6)$$

Figure 7 plots $S_{\mathrm{DPA}}(k)$ (left axis) and $\rho_{\mathrm{CPA}}(k)$ (right axis) across all guesses; the dashed line marks the true key (0×3A). With N=200 traces (see Table VII), neither statistic produces an outlying peak at the true key; its score lies within the bulk of the empirical distribution and does not attain rank-1. This outcome is consistent with signal spreading from the two-phase S-box and the small desynchronization between nibble updates, which lowers first-order exploitable correlation at practical trace counts.

C. Analysis of Software and Firmware Attacks ($\mathcal{F}_{soft}$)

The probability of software exploitation, $\mathbb{P}(V_{\mathrm{exploit}})$, is minimized by reducing the attack surface. Our approach leverages a hardware-software co-design, where the custom AES transformations are implemented via "microcode-level changes" on the ESP32's crypto engine. By offloading the core cryptographic logic to specialized, immutable hardware routines, we significantly reduce the risk of traditional software vulnerabilities like buffer overflows or code injection within the encryption core itself.

This security is complemented by system-level hardening. An adversary might attempt firmware reverse engineering to analyze our custom microcode. This is mitigated by using the ESP32's platform security features, such as Secure Boot, which ensures firmware integrity, and Flash Encryption, which protects the code while at rest.

D. Analysis of Protocol and Network Attacks ($\mathcal{F}_{proto}$, $\mathcal{F}_{net}$)

1) Timing and Man-in-the-Middle Attacks: The underlying cipher's performance is a critical security feature against timing-sensitive attacks. An adversary's success in attacks like Man-in-the-Middle (MITM) often depends on an operational time window, T_{window}, dictated by protocol timeouts. The probability of success, $P_{success}$, is proportional to this window. Our lightweight AES reduces the cryptographic time, T_{crypto}, by approximately 26%. This provides a quantifiable reduction in the attacker's operational window ($\Delta T_{window} \approx 0.26 \cdot T_{crypto,std}$), which in turn lowers the probability of a successful attack, $P_{success}$. Therefore, the 26% performance gain is not merely an efficiency metric but a direct security enhancement that hardens the device against timing-based network threats.

2) Denial-of-Service (DoS) Attacks: Our design's energy efficiency provides quantifiable resilience against power-exhaustion Denial-of-Service (DoS) attacks. The operational lifetime of the device under such an attack can be defined by the maximum number of cryptographic operations, N_{max}, it can perform before battery depletion, given by $N_{max} = E_{battery}/E_{op}$, where E_{op} is the energy per operation. The paper demonstrates up to a 28.57% decrease in E_{op} for our design compared to a standard implementation ($E_{opt} \approx 0.7143 \cdot E_{std}$). We can define a resilience hardening factor, $\mathcal{H}$, as the ratio of maximum operations our device can withstand versus the standard:

$$\mathcal{H} = \frac{N_{opt}}{N_{std}} = \frac{E_{battery}/E_{opt}}{E_{battery}/E_{std}} = \frac{E_{std}}{E_{opt}} \approx \frac{1}{0.7143} \approx 1.40 \quad (7)$$

This 40% increase in operational capacity directly raises the work factor and cost for an attacker to execute a successful battery exhaustion attack, providing a measurable improvement in device resilience.

E. Trade-Off Analysis of Security Measures

A practical implementation requires balancing the security layers discussed above against their associated costs. Using our risk model, $\mathcal{R}_{\mathrm{total}}(\mathcal{S}) = \sum_{a\in\mathcal{A}} \mathcal{F}_a(\mathcal{S}) \times \mathrm{Impact}(a, \mathcal{S})$, we can analyze the trade-offs not just for add-on countermeasures, but for the inherent security features of our design. Unlike traditional security measures that impose significant overhead, our core architectural choices provide security as a synergistic result of efficiency gains. The "cost" of our primary security measures is, in fact, a performance and energy *benefit*. We can summarize this unique trade-off as follows:

- **Measure 1: Lightweight Crypto-Core**.
  - **Security Benefit**: Reduces the success probability $\mathcal{F}_a(\mathcal{S})$ for Side-Channel Attacks (by lowering DPA signal-to-noise ratio and complicating templates) and Fault Injection Attacks (by altering fault propagation).
  - **Associated "Cost"**: The resource overhead is negative. This measure is the source of the **-26% CPU cycle** and **-28.57% energy** cost, hardening the device against timing and DoS attacks, respectively. The only true cost is a marginal increase in initial design complexity (C_{dev}).
- **Measure 2: Hardware-Software Co-Design**.
  - **Security Benefit**: Reduces $\mathcal{F}_a(\mathcal{S})$ for software exploits by minimizing the attack surface and leveraging platform features like Secure Boot.
  - **Associated Cost**: A higher initial development effort (C_{dev}) to create and verify the custom microcode.

Authorized licensed use limited to: University of Melbourne. Downloaded on November 03,2026 at 12:30:39 UTC from IEEE Xplore. Restrictions apply.

This analysis demonstrates that our baseline design provides a strong security posture with a uniquely efficient cost profile. For an even lower risk tolerance, one could add traditional countermeasures like full hardware masking. This would further decrease attack probabilities ($\mathcal{F}_{SCA}$) but would impose a significant positive cost, for instance, a $\leq$ 50% increase in $C_{CPUcycles}$ and C_{energy}. Our approach thus provides an optimal starting point, allowing implementers to flexibly select additional, costly measures only if their specific threat model and budget require a "fully hardened" solution.

IX. Diversity and Complexity of IoT Application Scenarios

Diverse IoT applications like healthcare, industrial control, and smart homes present varied security needs, data characteristics, and communication patterns, demanding adaptable cryptographic solutions. Healthcare IoT requires high confidentiality for sensitive data (e.g., under HIPAA/GDPR), where severe attack impacts (Impact($\mathcal{A}_{\text{health}}$)) necessitate robust measures like ephemeral ECDH and potentially side-channel mitigations (e.g., masking), even if they increase CPU overhead ($C_{\text{CPUcycles}}$ by $\sim$ 30%) while reducing attack success (by factors α_{mask} or α_{phys}). Industrial IoT (ICS/SCADA) emphasizes reliability and low-latency, requiring encryption throughput ($T_{\text{enc}} \times \frac{R_{\text{data}}}{\text{BlockSize}} \ll \Delta t_{\text{deadline}}$) met by our proposed lightweight AES (with nibble-based S-box and simplified MixColumns, cutting cycles by $\sim$ 26%) and TLS with ephemeral keys, supporting ciphers like AES and SM4. Smart home devices have moderate, varied needs, where robust encryption is vital for critical functions like door locks, while simpler key management (e.g., PSK) might suit less critical devices with short transmissions, though ephemeral ECDH and lightweight AES benefit privacy-sensitive data and battery life via cycle-saving nibble S-boxes. These domain-specific needs are influenced by data patterns: for high-volume streaming ($\dot{D}_{\text{data}} \gg 0$), reduced per-block encryption time ($T_{\text{AES}}(\text{block}) \approx \Delta t_{\text{hw}} - \delta_{\text{optimized}}$) is key, while for event-driven devices or varying packet sizes (ℓ_{pkt}), which affect total overhead ($T_{\text{total}} = \lceil \frac{\ell_{\text{pkt}}}{128\text{-bit}} \rceil \times T_{\text{AES}}(\text{block})$), flexible re-keying becomes important. Our approach, combining ephemeral ECDH with a configurable lightweight AES-128 design, adapts by accepting increased cycle overhead for strong masking in high-security domains, accelerating AES for real-time industrial needs, and allowing flexible re-keying for small data or when ephemeral overhead is a concern in smart homes, ensuring a robust, extensible design across diverse IoT security demands, with potential future work in dynamic reconfiguration.

X. Conclusion and Future Directions

This work presented a hardware-conscious cryptographic device for MCU-class IoT that couples an ESP32 and an MPU6050 sensing front-end with a lightweight AES-128 co-design (two-phase nibble S-Box and simplified MixColumns) and ephemeral ECDH session keys. By aligning the round structure with the ESP32 accelerator and physically separating cipher and hash datapaths for concurrent operation, the prototype preserved standards compatibility (TLS/IPsec) while improving efficiency: encryption time decreased by up to 42.4%, energy consumption by 28.57%, and compute throughput reached 435 Mbps. ECDH supplies forward secrecy; the round structure reduces lookup pressure and first-order leakage; and the reconfigurable architecture remains agile across AES/DES/SM4 and SHA-1/SHA-256/SM3. Evaluation used mean±SD over repeated runs and a Monte-Carlo MTT model; the one-time ECDH cost ($\sim$8.1 ms/1.14 mJ) amortizes to <5% beyond 100 KB payloads.

Future work will validate end-to-end stacks (TLS/DTLS/IPsec with AEAD GCM/CCM) under realistic wireless conditions; integrate masking/hiding with formal leakage tests and constant-time audits; broaden agility (SM3/SM4, X25519, hybrid post-quantum KEMs); and migrate to FPGA/ASIC and battery-powered field trials guided by the MTT framework. The objective is deployable, certifiable security with energy proportionality and robust resistance to practical attacks.

References

[1] A. Mosenia and N. K. Jha, "A comprehensive study of security of Internet-of-Things," *IEEE Trans. Emerg. Topics Comput.*, vol. 5, no. 4, pp. 586–602, Oct. 2017.

[2] H. Sikarwar, H. Vasudev, D. Das, M. Conti, and K. K. Mondal, "SECURE: Secure and efficient protocol using randomness and edge-computing for drone-assisted Internet of Vehicles," *IEEE Trans. Netw. Service Manage.*, vol. 21, no. 6, pp. 6974–6988, Dec. 2024.

[3] M. Kashif, I. Cicek, and M. Imran, "A hardware efficient elliptic curve accelerator for FPGA based cryptographic applications," in *Proc. 11th Int. Conf. Electr. Electron. Eng. (ELECO)*, Nov. 2019, pp. 362–366.

[4] A. O. Akmandor, H. Yin, and N. K. Jha, "Smart, secure, yet energy-efficient, Internet-of-Things sensors," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 4, no. 4, pp. 914–930, Oct. 2018.

[5] H. Nejatollahi, S. Gupta, M. Imani, T. S. Rosing, R. Cammarota, and N. Dutt, "CryptoPIM: In-memory acceleration for lattice-based cryptographic hardware," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6.

[6] X. Zhang et al., "Design and analysis of area and energy efficient reconfigurable cryptographic accelerator for securing IoT devices," *Sensors*, vol. 22, no. 23, p. 9160, Nov. 2022.

[7] S. Zhang, M. Wabnitz, and T. Gemmeke, "A compact 1,257-Gbps/W byte-serial AES accelerator for IoT applications in 22 nm," in *Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC)*, Nov. 2023, pp. 1–3.

[8] N. A. Lone, M. Nazish, I. Sultan, and M. T. Banday, "Optimised hardware implementation of AES for improving energy efficiency of low-power devices," in *Proc. Smart Technol. Commun. Robot.*, Dec. 2022, pp. 1–5.

[9] M. Suárez-Albela, P. Fraga-Lamas, L. Castedo, and T. M. Fernández-Caramés, "Clock frequency impact on the performance of high-security cryptographic cipher suites for energy-efficient resource-constrained IoT devices," *Sensors*, vol. 19, no. 1, p. 15, Dec. 2018.

[10] L. Pan, G. Tu, S. Liu, Z. Cai, and X. Xiong, "A lightweight AES coprocessor based on RISC-V custom instructions," *Secur. Commun. Netw.*, vol. 2021, pp. 1–13, Dec. 2021.

[11] M.-B. Lin and J.-H. Chuang, "The design of a high-throughput hardware architecture for the AES-GCM algorithm," *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 425–432, Feb. 2024.

[12] Z. Pei, V. P. Nambiar, Y. Sheng Chong, W. L. Goh, and A. T. Do, "3881 Gbps/W, 3005 μm AES core with state based clock gating for IoT applications," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2024, pp. 1–5.

[13] V. Padmakumar, T. M. Ignatius, T. B. Singha, R. P. Palathinkal, and S. R. Ahamed, "Boosting AES intrinsic resilience using split SubBytes round function against power attacks," *IEEE Embedded Syst. Lett.*, vol. 17, no. 1, pp. 10–13, Feb. 2025.

[14] S. Ahmed et al., "Lightweight AES design for IoT applications: Optimizations in FPGA and ASIC with DFA countermeasure strategies," *IEEE Access*, vol. 13, pp. 22489–22509, 2025.

Authorized licensed use limited to: University of Melbourne. Downloaded on November 03,2026 at 12:30:39 UTC from IEEE Xplore. Restrictions apply.

[15] B. Batmaz and A. Dogan, "Accelerating DTLS on SoC FPGA for secure IoT applications," *Internet Things*, vol. 28, Dec. 2024, Art. no. 101423.
[16] S. D. Thi, H. L. Pham, V. T. Duong Le, R. Imamura, T. H. Tran, and Y. Nakashima, "Small-footprint reconfigurable heterogeneous cryptographic accelerator for fog computing," in *Proc. RIVF Int. Conf. Comput. Commun. Technol. (RIVF)*, Dec. 2023, pp. 124–129.
[17] S. S. P. Goswami and G. Trivedi, "Comparison of hardware implementations of cryptographic algorithms for IoT applications," in *Proc. 33rd Int. Conf. Radioelektronika (RADIOELEKTRONIKA)*, Apr. 2023, pp. 1–6.
[18] O. Al-Khaleel, S. Baktir, M. Al-Khaleel, and A. Küpçü, "Efficient ECC processor designs for IoT using Edwards curves and exploiting FPGA embedded components," *IEEE Access*, vol. 12, pp. 167183–167200, 2024.
[19] C. Pham-Quoc and P. L. S. Ngan, "A high-throughput fpga-based elliptic curve digital signature core for IoT edge platforms," in *Proc. Int. Conf. Comput. Sci. Appl.* Cham, Switzerland: Springer, 2024, pp. 31–43.
[20] A. Ali et al., "An industrial IoT-based blockchain-enabled secure searchable encryption approach for healthcare systems using neural network," *Sensors*, vol. 22, no. 2, p. 572, Jan. 2022.
[21] Ismail et al., "A machine learning-based classification and prediction technique for DDoS attacks," *IEEE Access*, vol. 10, pp. 21443–21454, 2022.
[22] R. Khan et al., "A hybrid approach for seamless and interoperable communication in the Internet of Things," *IEEE Netw.*, vol. 35, no. 6, pp. 202–208, Nov. 2021.
[23] M. A. Jan et al., "An AI-enabled lightweight data fusion and load optimization approach for Internet of Things," *Future Gener. Comput. Syst.*, vol. 122, pp. 40–51, Sep. 2021.
[24] M. Adil et al., "An intelligent hybrid mutual authentication scheme for industrial Internet of Thing networks," *Comput., Mater. Continua*, vol. 68, no. 1, pp. 447–470, 2021.
[25] T. B. Singha, B. Sanjana, T. M. Ignatius, R. P. Palathinkal, and S. R. Ahamed, "Improvement in resilience of AES design with reconfigured CFB mode against power attacks," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 32, no. 11, pp. 2149–2153, Nov. 2024.
[26] J. Song, K. Lee, and J. Park, "Low area and low power threshold implementation design technique for AES S-box," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 70, no. 3, pp. 1169–1173, Mar. 2023.
[27] L. P. Fraile et al., "Enabling quantum-resistant EDHOC: Design and performance evaluation," *IEEE Access*, vol. 13, pp. 75861–75884, 2025.
[28] R. Schilling, T. Unterluggauer, S. Mangard, F. K. Gurkaynak, M. Muehlberghuber, and L. Benini, "High speed ASIC implementations of leakage-resilient cryptography," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Benin, Mar. 2018, pp. 1259–1264.
[29] P. Peng, C. Ma, J. Ge, N. Gao, and C. Tu, "A hardware/software collaborative SM4 implementation resistant to side-channel attacks on ARM-FPGA embedded SoC," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2020, pp. 1–7.
[30] H. Eldib, C. Wang, and P. Schaumont, "Formal verification of software countermeasures against side-channel attacks," *ACM Trans. Softw. Eng. Methodol.*, vol. 24, no. 2, pp. 1–24, Dec. 2014.
[31] M. N. Aman, K. C. Chua, and B. Sikdar, "Hardware primitives-based security protocols for the Internet of Things," in *Cryptographic Security Solutions for the Internet of Things*. Hershey, PA, USA: IGI Global, 2019, pp. 117–141.
[32] V. O. Nyangaresi and J. Ma, "A formally verified message validation protocol for intelligent IoT e-health systems," in *Proc. IEEE World Conf. Appl. Intell. Comput. (AIC)*, Jun. 2022, pp. 416–422.
[33] S. Ebrahimi and S. Bayat-Sarmadi, "Lightweight and DPA-resistant post-quantum cryptoprocessor based on binary ring-LWE," in *Proc. 20th Int. Symp. Comput. Archit. Digit. Syst. (CADS)*, Aug. 2020, pp. 1–6.



Koustav Kumar Mondal (Graduate Student Member, IEEE) received the M.Tech. degree in information technology with a specialization in the Internet of Things from Maulana Abul Kalam Azad University of Technology, West Bengal. He is a Ph.D. Scholar with the Indian Institute of Technology (IIT) Jodhpur, Rajasthan, India, within the Centre for AIoT and Applications. His research focuses on the development of physically unclonable functions (PUFs) and hardware security solution to enhance security in IoT and IoV ecosystems.



Himani Sikarwar (Member, IEEE) is pursued her Ph.D. degree in computer science and engineering with the Indian Institute of Technology, Jodhpur, India. She is currently serving as an Assistant Professor with Shiv Nadar IoE, Delhi NCR, India. Her research interests include vehicular adhoc networks, network security, and IoV.



Debasis Das (Senior Member, IEEE) is an Associate Professor with the Department of Computer Science and Engineering and the School of Artificial Intelligence and Data Science (AIDE) at the Indian Institute of Technology (IIT) Jodhpur, Rajasthan, India. His research interests include vehicular ad hoc networks, blockchain, the Internet of Things (IoT), machine learning, and artificial intelligence. He has been awarded the Early Career Research Award and BRICS Young Scientist Award from the Science & Engineering Research Board, Department of Science and Technology, Government of India. He is a senior member of VTS and a professional member of ACM.



Chun-I Fan (Senior Member, IEEE) received the M.S. degree in computer science from National Yang Ming Chiao Tung University, Taiwan, in 1993, and the Ph.D. degree in electrical engineering from National Taiwan University, in 1998. He is a Distinguished Professor at National Sun Yat-sen University (NSYSU) and Director of the Information Security Research Center. His research focuses on applied cryptology, information security, network security, and AI security. He has won numerous awards, including the FutureTech Award (2023), Best Journal Paper Award (2022), and IEEE Outstanding Technical Achievement Award (2020). He is also the Chairman of CCISA, Taiwan, and President of Institute for Information Industry, Taiwan.

Authorized licensed use limited to: University of Melbourne. Downloaded on November 03,2026 at 12:30:39 UTC from IEEE Xplore. Restrictions apply.