DESIGN AND EVALUATION OF A LIGHTWEIGHT ADAPTIVE ENCRYPTION FRAMEWORK FOR ESP32-BASED IOT CYBER-PHYSICAL SYSTEMS

Mateusz Jakusz | Department of Computing | SETU Carlow, Ireland | 2026

SETU Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

INTRODUCTION

IoT devices are now embedded in cyber-physical systems (CPS) where telemetry directly influences physical processes. As deployments scale, securing constrained microcontrollers becomes critical.

The ESP32-P4 is a widely-used MCU with limited resources. Applying the same cipher to every message wastes resources or under-protects data.

This research proposes a **payload-adaptive algorithm selection policy** that selects the best AEAD scheme per message size, always maintaining authentication.

RESEARCH QUESTIONS

RQ1 How do AES-CTR, AES-GCM, ChaCha20-Poly1305 and ASCON-AEAD128 differ in encryption/decryption latency, memory overhead and processing behaviour across payload sizes?

RQ2 How do the evaluated schemes differ in security properties — authenticated encryption, tamper detection and replay resistance?

RQ3 Can a payload-adaptive policy achieve better security-performance balance than any single static scheme on an ESP32-P4 workload?

LITERATURE REVIEW

Hassija et al. (2019) and **Zhou et al. (2019)** identify IoT heterogeneity and static configurations as core security challenges.

Bormann et al. (2014) define constrained-node networks. Mehrotra et al. (2022) confirm AES-based schemes balance security and performance on low-power devices.

NIST (2023) standardised **ASCON-AEAD128** as the primary Lightweight Cryptography standard — permutation-based, 128-bit key/nonce/tag, minimal gate count. Ideal for constrained IoT hardware.

Nir & Langley (2018) standardised ChaCha20-Poly1305 (RFC 8439). Software-friendly design avoids look-up tables and operates on 32-bit additions — no hardware AES required.

METHODOLOGY

Quantitative experimental design — two ESP32-P4 nodes via MQTT through AWS IoT Core (eu-west-2). Four static runs + adaptive simulation.

- **Sender**: generates payloads 256–2048 B (linear sweep), encrypts, publishes binary frame to esp32/data + JSON metadata to esp32/meta
- **Receiver**: decrypts, verifies AEAD tags, checks replay via monotonic msg_id, publishes metrics to esp32/meta_rx
- **Each run**: 1,500 messages (6,000 total). Logs saved to S3 → analysed with Python (pandas/matplotlib).

SYSTEM ARCHITECTURE

ESP32-P4 Sender	AWS IoT Core	ESP32-P4 Receiver	Python Analysis
Encrypt + Publish	MQTT Broker/S3	Decrypt + Verify	Figures + Stats

ALGORITHMS EVALUATED

Algorithm	Type	Key	OH	Tag
AES-CTR	No auth	128	17 B	—
AES-GCM	AEAD / mbedTLS HW	128	29 B	16 B
ChaCha20-Poly1305	AEAD / RFC 8439	256	29 B	16 B
ASCON-AEAD128	AEAD / NIST 2023	128	33 B	16 B

ADAPTIVE POLICY LOGIC

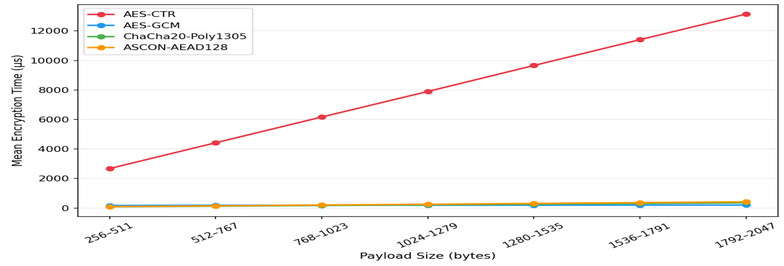
< 512 B	512–1023 B	≥ 1024 B
ASCON-AEAD128	*ChaCha20-Poly1305*	*AES-GCM*
saves 77 µs vs GCM	saves 41 µs vs GCM	optimal for large

6,000	100%	0	6,000
Messages	Delivery Rate	Auth Failures	Matched IDs

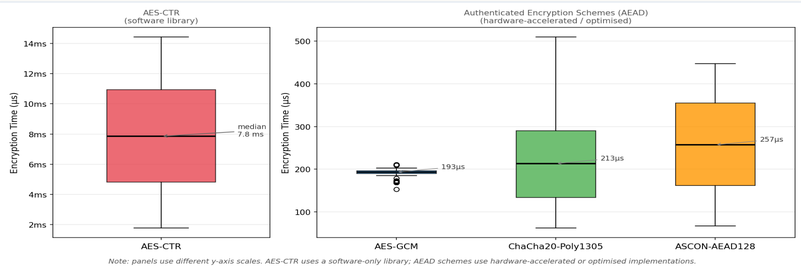
KEY RESULTS

7,900 µs	191 µs	216 µs	258 µs
AES-CTR mean enc	AES-GCM mean enc	ChaCha20 mean enc	ASCON mean enc

ENCRYPTION LATENCY VS PAYLOAD SIZE



LATENCY DISTRIBUTION



CONCLUSIONS & NEXT STEPS

✓ All three AEAD schemes outperformed AES-CTR by 31–41× — explained by hardware-accelerated mbedTLS (GCM) vs software-only Arduino Crypto library (CTR).

✓ Adaptive policy saves 19–77 µs per message on short/medium payloads while maintaining authenticated encryption — AES-CTR excluded from policy path (no auth tag).

✓ 100% delivery across 6,000 messages. Zero authentication failures across all 4,500 AEAD messages confirms correct implementation on ESP32-P4.

→ **Future work:** Live adaptive hardware run, deliberate tamper/replay injection tests, multi-device deployment, energy consumption profiling.

setu.ie

bloobsky@gmail.com |
github.com/bloobsky/esp32-encryption |
SETU Carlow 2026