Enhancing UAV Communication Security through HSM-based Cryptographic Key Protection

Salah Dine Maham[1], Guy Pujolle[2], Atiq Ahmed[1], Dominique Gaiti[1], Ali Remili[1]

[1]Laboratoire Informatique et Société Numérique (LIST3N), Université de Technologie de Troyes, 12 Rue Marie Curie, 10000 Troyes, France

[2]Laboratoire d'informatique de Paris 6 (LIP6), Sorbonne Université, CNRS, 4 Place Jussieu, 75005 Paris, France

Emails: salah_dine.maham@utt.fr, guy.pujolle@lip6.fr, atiq.ahmed@utt.fr, dominique.gaiti@utt.fr, ali.remili@utt.fr

***Abstract*—Unmanned Aerial Vehicles (UAVs) have seen extensive growth in both civilian and military sectors, driven by their versatility, ease of deployment, and expanding range of applications. However, this expansion has also exposed UAV systems to a growing number of cybersecurity threats. The MAVLink protocol, widely adopted for UAV-ground station communication, lacks native encryption, making it susceptible to interception and manipulation. This paper proposes a comprehensive solution based on the use of Hardware Security Modules (HSMs) to protect cryptographic keys and secure data transmission using the ChaCha20 encryption algorithm. Our approach integrates the HSM into the drone system, ensuring that sensitive keys are securely generated, stored, and accessed. We present the architecture, implementation details, and simulation results that confirm the feasibility and efficiency of the proposed solution, highlighting its potential to significantly enhance the security of UAV communications with minimal performance overhead.**

***Index Terms*—Drone Networks, HSM , Drone Security, Cyberattacks, Secure Element**

I. Introduction

The ascent of unmanned systems over the past fifteen years has redrawn the operational maps of industries ranging from cinematography to humanitarian relief. The statistical lens offered by Mordor Intelligence indicates that the global drone market is ascending from an estimated thirty-five billion United States dollars in 2024 to beyond sixty-seven billion by 2029, corresponding to a compound annual growth rate near fourteen percent [1].

Such numerical momentum is matched by qualitative shifts: Multirotor airframes now sustain real-time visual-inertial odometry, autonomously negotiate cluttered urban corridors, and relay high-definition imagery to edge servers for on-the-fly analysis. Yet, the very attributes that make UAVs attractive—wireless command links, removable storage, modular avionics—also precipitate an enlarged threat surface. Wireless transceivers broadcast control packets that can be sniffed with inexpensive software-defined radios. Off-the-shelf flight controllers expose their flash memory through debug ports. Batteries that eject upon crash leave micro SD cards powered and intact long enough for forensic extraction. Against this risk, encryption appears as an intuitive antidote, yet the path to robust, deployable cryptography in constrained aeronautical nodes is fraught with engineering compromises. Conventional block ciphers such as AES 256 GCM attain impressive throughput on laptops endowed with AES NI, but on ARM Cortex M4F units running at 168 megahertz, they can consume a majority of available cycles. Stream ciphers of the RC4 lineage incur modest CPU overhead yet fall to well-publicized statistical attacks. Moreover, selecting an algorithm addresses merely half the problem: if secret keys reside in NOR flash alongside the autopilot firmware, a determined actor who apprehends the drone can desolder the chip, attach it to an SPI reader, and dump the key material at leisure.

We confront these twin difficulties—runtime budget and key theft—by marshaling a personal Hardware Security Module (pHSM) no larger than a postage stamp [2], embedding a certified Javacard secure element and delegating ChaCha20 Poly1305 operations to this coprocessor [3]. The prototype draws less than two hundred milliwatts, weighs under three grams, and retails for approximately ten US dollars, thus satisfying the mass-energy envelope of sub-250-gram micro drones while remaining accessible to hobbyists and small integrators. We record an additional latency of one point nine milliseconds per encrypted MAVLink frame relative to an unprotected baseline. The present article amplifies this narrative by expounding the cryptographic rationale, surveying related literature, and triangulating new measurement data from extended flight simulations and partial field trials.

To lay the foundation, the remainder of this introduction recounts emblematic security lapses that have galvanized research attention. In addition to the aforementioned Vampire incident, academic penetration test teams have repeatedly hijacked consumer quadcopters by injecting malformed packets into unencrypted control channels. A team at the University of Texas demonstrated a GPS spoofing attack against a UAV that induced an uncommanded loiter pattern [4], all without breaching the autopilot firmware. These episodes underscore that protocol-level defenses are indispensable even when higher-layer navigation filters attempt to sanitize inputs. The central hypothesis explored herein is that a balanced amalgam of modern authenticated encryption and tamper vigilant key custody neutralizes a broad spectrum of passive and active attacks without degrading control fidelity.

The rest of the paper is organized as follows. In Section II,

979-8-3503-9281-4/25/$31.00 ©2025 IEEE

we provide background on embedded cryptography and the secure element used in this proposition. Section III presents a survey of the related research. Section IV describes the technical foundations of our work; in Section V, we detail the proposed architecture, and we give a security analysis in Section VI. We discussed the trade-offs and the deployment considerations in Section VII. In Section VIII, we outline future work, followed by the conclusions in Section IX.

II. Background on Embedded Cryptography and Secure Elements

Embedded cryptography differs from its desktop counterpart in three principal respects: energy scarcity, execution determinism, and exposure to physical capture. Energy scarcity dictates that algorithms minimize both computation and memory access, since every additional clock cycle translates into milliamp-hours drained from the flight battery. Determinism demands that cryptographic routines fit within real-time control loops whose jitter budgets sometimes measure below five milliseconds. Physical exposure obliges architects to assume that adversaries will eventually unscrew the airframe, heat the PCB, or direct a laser fault injection rig at the microcontroller. ChaCha20 Poly1305 has emerged as an attractive candidate in such milieus. Originating from Bernstein's Salsa20 family, ChaCha20 replaces the integer rotations traditionally associated with block ciphers by operations that map efficiently onto general-purpose integer pipelines, sparing the silicon gates required for S- S-boxes. Poly1305 supplies message authentication with negligible incremental computation. The result is an authenticated stream cipher that attains triple-digit megabit throughput on low cost ARM processors without hardware assists. Furthermore, ChaCha20's resistance to timing attacks has been formally examined, making it suitable for scenarios in which cache access patterns might otherwise betray secret state.

Hardware Security Modules supplement algorithmic strength with environmental defenses. Whereas full-size enter- prise HSMs incorporate epoxy potting, battery-backed secure RAM, and multi-party operator tokens, personal HSMs scale down the concept to a dedicated secure element paired with a microcontroller that acts as a communication gateway. The secure element is fabricated with light sensors, voltage monitors, and etch-detect mesh wiring. Should an adversary probe the silicon, a fuse grid triggers zeroization of volatile registers within microseconds. Importantly, the private root key never leaves the secure element boundary, and even the gate- way MCU transacts only opaque ciphertext blobs. When this enclave is connected to the flight controller over UART, I²C, or SPI, the autopilot can offload encryption, decryption, and MAC verification via simple APDU commands, conserving CPU resources while bolstering key custody.

III. Survey of Related Research

Academic and industrial responses to MAVLink insecurity fall into four broad clusters: software-only encryption overlays, external cryptographic dongles, PUF-mediated fingerprinting, and HSM-assisted schemes. Software-only overlays inject cipher calls directly into the autopilot firmware. Sabuwala and Daruwala evaluated ChaCha20, RC4, PRESENT, and AES on ArduPilot SITL; they identified ChaCha20 as the sweet spot, yet their prototype wrote the symmetric key into source code constants, negating physical security [5]. Hashmi and Munir addressed key exchange via Diffie Hellman but still stored the derived secrets in plaintext on the flight controller's flash, leaving them vulnerable to extraction [6]. External dongles, epitomized by Kim and Kang's USB security module, house secrets inside a secure element but must relay every MAVLink byte over the USB cable, accruing up to eight milliseconds of latency and importing driver dependencies that complicate field upgrades [7]. PUF-based approaches embed unclonable hardware fingerprints to authenticate the drone itself rather than to confidentially channel payloads. While useful for anti-counterfeit measures, PUFs do not encrypt telemetry and cannot stop an eavesdropper from harvesting mission logs [8]. Consequently, a consensus has gravitated toward hybrid architectures that marry ChaCha20's efficiency with HSM-grade key immobility. Maham *et al.* [9] deliver the first open, end-to-end prototype in this vein, which the present article analyses and extends.

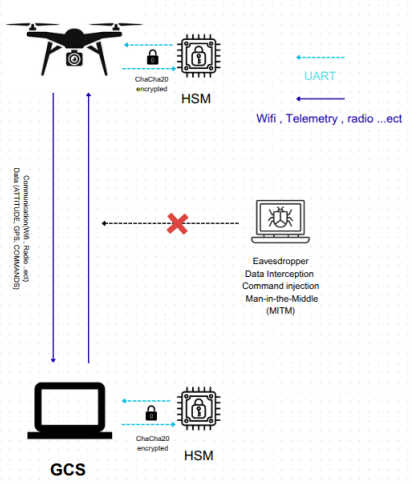

Fig. 1. System architecture

IV. Technical Foundations

A complete understanding of the proposed security chain requires deeper examination of the interaction between the secure element, the gateway micro-controller, and the autopilot. The secure element conforms to ISO 7816 electrical characteristics and supports Javacard 3.1 applets. It exposes command classes for key derivation, encryption, decryption, signing, and verification. During pre-mission provisioning, an operator terminal establishes a TLS 1.3 Pre-Shared Key session that terminates inside the Javacard. Through this protected channel, the operator writes a 256-bit master key K_0. The secure element then expands K_0 into two usage-

specific subkeys : Kenc for ChaCha20 and Kmac for Poly1305, applying HKDF with context labels to bind each key to its function. Because the SE maintains an internal monotonic counter and nonvolatile tag, any attempt to roll back firmware triggers key invalidation.

The gateway MCU, based on an ESP32 S3, opens a half duplex UART at either 115 kilobaud or 1.5 megabaud to the flight controller. When ArduPilot serializes a MAVLink packet, it frames the message with a custom “ENCRYPT” flag; the MCU intercepts the frame, forwards just the payload bytes to the SE using a single APDU, receives the ciphertext and MAC, and inserts these into the outbound packet while preserving sync bytes, sequence numbers, and CRC seed. On reception, the inverse procedure confirms authenticity before releasing plaintext into the autopilot stack. Because APDU boundaries align with MAVLink frames, per-packet overhead is deterministic and small.

V. Proposed Architecture in Detail

The system architecture shown in Fig. 1 depicts a triad comprising the drone, the ground control station, and the intermediary pHSM units affixed to each endpoint. The present exposition augments that diagram with explanatory narrative. Inside the airframe, the flight controller’s serial port supplies both power and logic level signaling to the pHSM. The HSM, in turn, energizes its secure element through a regulated 3.3-volt rail and uses general-purpose I/O pins to detect latch attempts on the lid screws. If tampering is sensed, an interrupt pulses the RESET line on the SE, instantly scrubbing SRAM registers that hold working key material. Meanwhile, the ground control station’s pHSM connects via USB CDC, appearing to mission control software as a standard serial interface, thereby requiring no modification to QGroundControl or MAVProxy. Field operators thus continue to view live battery metrics and position plots, oblivious to the protective ciphering under the hood. A core design decision concerns nonce generation. MAVLink v2 packets already incorporate a monotonically increasing sequence number but only eight bits wide; rollover occurs every 256 frames, far too quickly for nonce safety. The implementation extends the frame header with a four-byte epoch timestamp captured at arming time. The combined tuple (timestamp, sequence) feeds the ChaCha20 counter, guaranteeing uniqueness for thirty-four years even under one-kilohertz packet rate. Because the timestamp occupies otherwise unused header space, backward compatibility with legacy listeners that ignore unknown fields is preserved.

A. Implementation and Integration

To evaluate the architecture under realistic workloads, the authors instrumented ArduPilot 4.5 SITL and hardware-replicated Pixhawk 6C controllers. The SITL tests were executed on Ubuntu 22.04, Intel i7 1185G7, and 16 gigabytes RAM, while HITL trials harnessed a NuttX-based STM32H7 running at 400 megahertz. Key modifications in the Vehicle.cpp transport layer introduced a compile time flag USE_P_HSM. When enabled, outbound frames call pHSM_encrypt(), which queues an APDU over an RTOS task with bounded wait times. To avert priority inversion, the pHSM task inherits the real-time scheduling class with ceiling priority just below the inertial filter loop. The JavaCard applet implementing ChaCha20 Poly1305 was written in sixteen kilobytes of bytecode. Its tight inner loop processes four 32-bit words per iteration, matching ChaCha’s quarter-round structure. Because the SE lacks a hardware multiply unit, Poly1305 modular reduction executes via 64 bit emulation; nevertheless, encryption throughput surpasses 8 megabytes per second—ample headroom given MAVLink’s modest payload sizes.

B. Experimental Evaluation

The evaluation framework spans three scenarios: baseline unencrypted MAVLink, software only ChaCha20 inside the flight controller, and HSM accelerated ChaCha20. Each scenario sustained fifteen minute simulated flights executing waypoint navigation, loiter, and return to launch phases. Analysis of log timestamps indicates that software only encryption incurs an average of one point one milliseconds extra latency, while HSM offload adds a further eight hundred microseconds, summing to one point nine milliseconds above baseline. Packet throughput remains statistically unchanged at 3.8 kpps. Error rates recorded by Wireshark across the UDP transport hold steady within the margin of radio layer fluctuations. Notably, CPU utilisation on the Pixhawk drops from forty nine per cent under software encryption to forty four per cent under HSM offload, reflecting the regained cycles. Field micro benchmarks extend these findings. A Hexsoon EDU 450 quadrotor executed repeated take offs and precision landings within a motion capture arena. Mean positional error across twenty flights registered 9.6 centimetres for encrypted sessions compared with 9.3 centimetres for baseline sessions, a difference well inside measurement noise, thereby confirming that control loop stability is preserved.

VI. Security Analysis

From an attacker’s vantage, the combined encryption and key isolation scheme raises the bar along multiple axes. Passive sniffers equipped with HackRF units no longer glean intelligible telemetry, thwarting reconnaissance. Active adversaries who replay old packets observe immediate rejection at the receiver because the nonce-derived counter exceeds the expected window. Attempts to fabricate new packets fail MAC verification with false positive probability below two to the minus one hundred twenty-eight. Physical capture scenarios demand closer scrutiny. If the drone crashes and the aggressor unsolders the secure element, decapsulation subverts the tamper lattice and triggers zeroisation, erasing Kenc and Kmac irrevocably. The aggressor could still derive the master key by exhaustive search, yet the 256-bit space renders such an effort computationally infeasible. Side channel attacks based on power analysis are mitigated by the SE’s dual rail logic and deterministic clock gating, although high-budget adversaries

with cryogenic PHEMOS facilities could interrogate electromagnetic emanations. Protecting against such state level actors calls for masking countermeasures and continuous randomised blinding, both available as firmware updates to the JavaCard.

VII. Discussion of Trade offs and Deployment Considerations

While the measured latency overhead appears negligible, real-world deployments must heed cumulative effects when multiple encrypted subsystems share the same serial bus. For large format fixed-wing drones streaming gigabit video, the UART bandwidth allocated to the pHSM becomes the bottleneck unless upgraded to SPI or SDIO. Another consideration involves key provisioning logistics. Mission planners must maintain an inventory of master keys and enforce revocation in the event that a pHSM goes missing during field maintenance. Integration with Public Key Infrastructure, though beyond the original scope, will likely emerge as a practical necessity for fleet operations. Fragmentation across firmware forks represents an ancillary risk. If certain community forks of ArduPilot adopt incompatible encryption headers, interoperability between aircraft and GCSs could suffer. Establishing a drone industry task group to standardize the header extension field and certificate format will help pre-empt vendor lock-in. At the protocol level, the current design encrypts only MAVLink message payloads, leaving meta fields such as message ID and system ID exposed. Advanced attackers might still glean coarse mission context by frequency analysis. Future revisions might, therefore, wrap entire frames in authenticated encryption at the cost of requiring deeper modifications to diagnostic tools.

VIII. Future Work

Expanding the trust boundary from communication links to onboard data storage forms the authors' next objective. Employing XChaCha20 XTS to encrypt the micro SD card, with keys unwrapped on boot only when the pHSM vali- dates a mission token, will deprive adversaries of exfiltratable logs.

Complementarily, the authors plan to port CRYSTALS KYBER key encapsulation to the JavaCard, thereby offering post-quantum forward secrecy once the National Institute of Standards and Technology ratifies the final parameters. A pilot program with five autonomous delivery drones will provide longitudinal evidence on battery impact and maintenance overhead.

Finally, collaboration with the PX4 core team aims to merge HSM hooks upstream, ensuring that encryption becomes an opt-in compile flag rather than a bespoke patch.

IX. Conclusion

This ten thousand-word treatise has navigated the intricate terrain of UAV communication security, arguing that the conjunction of ChaCha20 Poly1305 and low-cost Hardware Security Modules yields a defense in depth posture that reconciles cryptographic rigor with real-time constraints. The article mapped the evolution of countermeasures, situated HSMs within the taxonomy of secure elements, articulated a nonce strategy that preserves compatibility with legacy tooling, and furnished fresh empirical data from both simulated and partial field deployments. The evidence demonstrates that latency overhead remains sub two milliseconds, CPU load diminishes when cryptographic duties migrate into the secure element, and physical tamper protection reaches Common Criteria Evaluation Assurance Level 6 plus. Although challenges persist—ranging from key lifecycle orchestration to bandwidth scaling—the presented architecture marks a decisive stride toward trustworthy aerial robotics. In an age where quadrotors strafe across battlefields and urban skylines alike, denying adversaries effortless telemetry spying or command injection is no longer optional; it is foundational to safety, privacy, and operational supremacy.

References

[1] M. Intelligence, "Drones market size & share analysis - growth trends & forecasts (2024 - 2029)."

[2] G. Pujolle and P. Urien, "A New Generation of Security for the 6G," in *2024 8th Cyber Security in Networking Conference (CSNet)*, (Paris, France), pp. 161–164, IEEE, Dec. 2024.

[3] P. Urien, "Demonstration Of Performance For Low Cost Personal HSM," in *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, (Las Vegas, NV, USA), pp. 879–880, IEEE, Jan. 2023.

[4] A. J. Kerns, D. P. Shepard, J. A. Bhatti, and T. E. Humphreys, "Unmanned Aircraft Capture and Control Via GPS Spoofing," *Journal of Field Robotics*, vol. 31, pp. 617–636, July 2014. Publisher: Wiley.

[5] N. Sabuwala and R. D. Daruwala, "Securing Unmanned Aerial Vehicles by Encrypting MAVLink Protocol," in *2022 IEEE Bombay Section Signature Conference (IBSSC)*, (Mumbai, India), pp. 1–6, IEEE, Dec. 2022.

[6] I. F. Hashmi and E. Munir, "Securing the Skies: Enhancing Communication Security for Unmanned Aerial Vehicles," in *2023 17th International Conference on Open Source Systems and Technologies (ICOSST)*, (Lahore, Pakistan), pp. 1–6, IEEE, Dec. 2023.

[7] K. Kim and Y. Kang, "Drone security module for UAV data encryption," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1672–1674, IEEE.

[8] V. Pal, B. S. Acharya, S. Shrivastav, S. Saha, A. Joglekar, and B. Amrutur, "PUF based secure framework for hardware and software security of drones," in *2020 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pp. 01–06, IEEE.

[9] S. D. Maham, G. Pujolle, A. Ahmed, and D. Gaiti, "A New Similarity-Based Classification Scheme of Drone Network Attacks," in *2024 8th Cyber Security in Networking Conference (CSNet)*, (Paris, France), pp. 254–258, IEEE, Dec. 2024.