PQC-MAV: A Complete Post-Quantum Cryptographic Tunnel for UAV–GCS Communication

Burak Güneysu
Department of Computer Science
University of Applied Sciences
burak@example.edu

***Abstract*—Unmanned aerial vehicles (UAVs) depend on continuous, low-latency MAVLink telemetry between the flight controller and a ground-control station (GCS). The advent of cryptographically relevant quantum computers threatens every classical key-agreement scheme protecting this link, yet *no complete post-quantum cryptographic (PQC) tunnel for MAVLink traffic exists in the literature*.**

We present PQC-MAV, a *bump-in-the-wire* PQC tunnel that transparently encrypts bidirectional MAVLink traffic between a Raspberry Pi 5 drone companion computer and a Windows 10 GCS. PQC-MAV implements a full cryptographic stack: a KEM + SIG + HKDF handshake protocol, a custom AEAD framing layer with deterministic nonce construction and anti-replay protection, and a 72-suite cipher registry spanning three KEM families (ML-KEM, HQC, Classic McEliece), three signature families (ML-DSA, Falcon, SPHINCS$^+$), and three AEADs (AES-256-GCM, ChaCha20-Poly1305, Ascon-128a) across NIST security levels 1, 3, and 5.

We benchmark every cryptographic primitive in isolation (19,600 timed operations) and validate 71 of 72 suites through complete end-to-end tunnel runs on real hardware. KEM keygen times span *four orders of magnitude*—from 0.08 ms (ML-KEM-512) to 8,835 ms (Classic McEliece-8192128)—demonstrating that algorithm-family selection, not parameter tuning, is the decisive factor for UAV deployability.

To manage this heterogeneity at runtime, we introduce a telemetry-aware adaptive rekey policy that consumes battery voltage, SoC temperature, link quality, and armed state to select, switch, and gracefully degrade cipher suites during flight. We identify three Pareto-optimal suites and show that ML-KEM-based tunnels achieve a rekey overhead below 0.03% at 60-second intervals, while Classic McEliece tunnels incur up to 13.3% overhead—a 440× difference that makes runtime suite management essential.

***Index Terms*—post-quantum cryptography, UAV security, MAVLink, authenticated encryption, key encapsulation, cipher agility, NIST FIPS 203/204/205, ARM, embedded systems**

I. Introduction

Modern UAVs communicate with their ground-control stations via MAVLink 2.0 [8], a compact binary telemetry protocol that carries heartbeats, GPS coordinates, attitude quaternions, and actuator commands at rates up to 320 Hz. Intercepting or injecting MAVLink packets can hijack the vehicle, exfiltrate mission data, or trigger safety-critical failsafes. Protecting this link with authenticated encryption is therefore a first-order requirement for any operational deployment.

The NIST post-quantum cryptography (PQC) standardisation process [1] has produced three final standards—FIPS 203 (ML-KEM) [2], FIPS 204 (ML-DSA) [3], and FIPS 205 (SLH-DSA / SPHINCS$^+$) [4]—with additional candidates such as Falcon [5], HQC [6], and Classic McEliece [7] under evaluation. While PQC algorithms have been extensively benchmarked on x86 servers [13], [14] and Cortex-M microcontrollers [11], *no prior work has integrated them into a complete, functioning tunnel for real UAV traffic on ARM Cortex-A hardware*.

Challenges. Deploying PQC on a drone companion computer introduces three challenges absent from server-side TLS:

1) **Extreme performance heterogeneity.** Across the nine KEM algorithms in our registry, keygen times span from 0.08 ms to 8,835 ms—a ratio exceeding 10^5. Choosing the wrong algorithm can cause multi-second link blackouts during which no telemetry flows.
2) **Resource constraints.** A Raspberry Pi 5 operates within a 3.8 W power budget, 3,796 MB RAM, and thermal limits (80 °C throttle point). Heavy PQC operations spike CPU and temperature, risking throttle-induced packet loss.
3) **Safety-critical continuity.** MAVLink heartbeats must arrive within 5 s to prevent GCS failsafe triggers. Any rekey operation that blocks the data plane beyond this deadline is a safety violation.

Contributions. We make five contributions:

C1: We design and implement PQC-MAV, a *complete* bump-in-the-wire PQC tunnel for MAVLink with a formal handshake protocol (KEM + SIG + HKDF), a custom AEAD wire format with deterministic nonces and anti-replay, and a two-phase rekey mechanism (Section III).

C2: We construct a **72-suite cipher registry** (9 KEMs × 8 SIGs × 3 AEADs) covering NIST levels 1, 3, and 5 with three mathematically distinct KEM families (lattice, code-based, quasi-cyclic code) and three signature families (Section III).

C3: We benchmark 19,600 **cryptographic operations** and **71 end-to-end tunnel handshakes** on a Raspberry Pi 5, providing the most comprehensive PQC benchmark dataset for ARM Cortex-A76 drone platforms to date (Section V).

C4: We identify the **Pareto frontier** of NIST security level versus handshake latency and derive the *rekey overhead fraction* Φ, proving that only ML-KEM suites are viable for sub-minute rekey intervals (Section V).

C5: We present a **telemetry-aware adaptive rekey policy** that consumes real-time drone telemetry to select, switch, and gracefully degrade suites during flight, preventing the thermal and link failures that occur under naïve scheduling (Section IV).

Paper organisation. Section II defines the threat model. Section III presents the tunnel architecture. Section IV describes the adaptive policy. Section V reports the experimental evaluation. Section VI discusses findings and limitations. Section VII surveys related work. Section VIII concludes.

II. Threat Model

We consider a Dolev–Yao adversary [19] who controls the wireless channel between the drone and GCS:

- **Eavesdropping:** The adversary can record all ciphertexts traversing the link.
- **Injection/modification:** The adversary can inject, drop, replay, or reorder packets.
- **Quantum capability:** The adversary possesses a cryptographically relevant quantum computer (CRQC), rendering RSA, ECDH, and ECDSA insecure.

Trust assumptions:

- The drone and GCS share a *pre-shared key* (PSK) for mutual identity binding, deployed via a secure out-of-band channel before flight.
- Per-suite PQC signing key pairs are pre-generated and distributed offline. The GCS holds the signing key; the drone holds the corresponding public key.
- The companion computer's firmware and the liboqs library [10] are trusted.

Security goals:

G1: *Confidentiality*: No plaintext MAVLink data is recoverable from the encrypted channel, even by a quantum adversary.

G2: *Integrity*: Any modification to a ciphertext is detected and rejected (AEAD authentication).

G3: *Replay protection*: Duplicate or reordered packets are detected via sequence-number tracking with a sliding-window bitmap.

G4: *Forward secrecy*: Each handshake generates ephemeral KEM keys; compromise of long-term keys does not reveal past session keys.

G5: *Anti-downgrade*: The negotiated suite is embedded in the signed transcript; any mismatch is rejected.

III. System Architecture

A. Tunnel Overview

PQC-MAV is a *bump-in-the-wire* proxy: it sits transparently between the existing MAVLink endpoints (flight controller and GCS application) without requiring any modification to either. Figure 1 shows the end-to-end data path.

TABLE I: Algorithms in the PQC-MAV cipher suite registry

Type	Family	Variants	Levels
KEM	ML-KEM (FIPS 203)	512, 768, 1024	1,3,5
	Classic McEliece	348864, 460896, 8192128	1,3,5
	HQC	128, 192, 256	1,3,5
SIG	ML-DSA (FIPS 204)	44, 65, 87	1,3,5
	Falcon	512, 1024	1,5
	SPHINCS$^+$ (FIPS 205)	128s, 192s, 256s	1,3,5
AEAD	AES-256-GCM	—	—
	ChaCha20-Poly1305	—	—
	Ascon-128a	—	—

Suite count: L1: $3 \times 3 \times 3 = 27$; L3: $3 \times 2 \times 3 = 18$; L5: $3 \times 3 \times 3 = 27$. **Total: 72.**

On the drone, the Pixhawk flight controller emits MAVLink packets over serial to MAVProxy [9], which bridges them to UDP. The PQC proxy reads plaintext UDP datagrams, encrypts each one with the active AEAD session key, and transmits the ciphertext over the network. On the GCS side, a mirror PQC proxy decrypts and forwards to a second MAVProxy instance, which delivers plaintext UDP to QGroundControl. The path is fully bidirectional: commands from the GCS traverse the reverse direction.

Controller–follower model. The drone acts as the *controller*: it initiates all suite changes, manages the rekey schedule, and sends JSON-RPC commands (`start_proxy`, `prepare_rekey`, `stop`, `chronos_sync`) to the GCS over a TCP control channel (port 48080). The GCS is a passive *follower* that executes commands and reports telemetry. This design reflects the operational reality that the drone is the security-critical endpoint.

B. Cipher Suite Registry

PQC-MAV maintains an immutable registry of 72 cipher suites, constructed as the Cartesian product of NIST-level-matched KEM and SIG algorithms crossed with all three AEADs:

$$\mathcal{S} = \{(\text{KEM}_i, \text{SIG}_j, \text{AEAD}_k) \mid L(\text{KEM}_i) = L(\text{SIG}_j)\} \quad (1)$$

where $L(\cdot)$ returns the NIST security level. Table I enumerates the constituent algorithms. Suite identifiers follow the format `cs-{kem}-{aead}-{sig}`, *e.g.*, `cs-mlkem768-aesgcm-mldsa65`.

Each suite record carries wire-format header identifiers (1-byte KEM family/parameter IDs, 1-byte SIG family/parameter IDs), the HKDF label, OQS algorithm names for runtime dispatch, and a human-readable display name. The registry is immutable after construction; runtime filtering (*e.g.*, by maximum NIST level or allowed AEAD subset) produces views without modifying the source.

Algorithm diversity. The three KEM families rely on distinct mathematical assumptions: ML-KEM on module-LWE (lattices), HQC on quasi-cyclic codes, and Classic McEliece on Niederreiter (binary Goppa codes). This diversity enables a

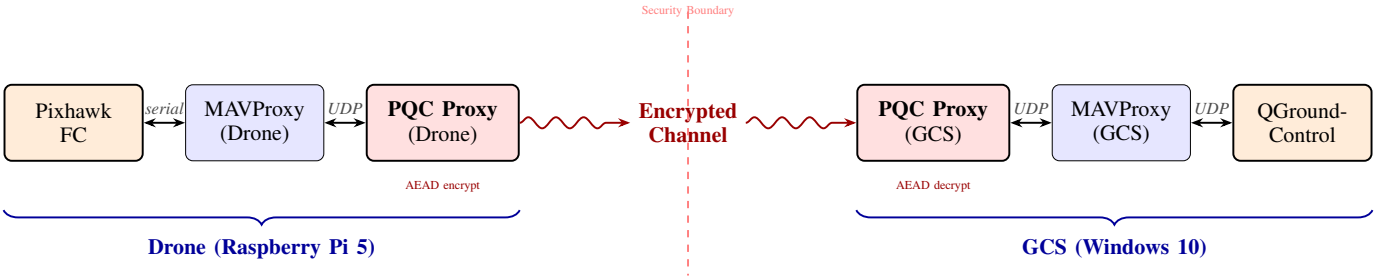

Fig. 1: PQC-MAV tunnel architecture. The PQC proxy encrypts/decrypts every UDP datagram using AEAD with session keys derived from a KEM + SIG + HKDF handshake. MAVProxy and the flight controller/GCS application are unmodified—the tunnel is fully transparent.

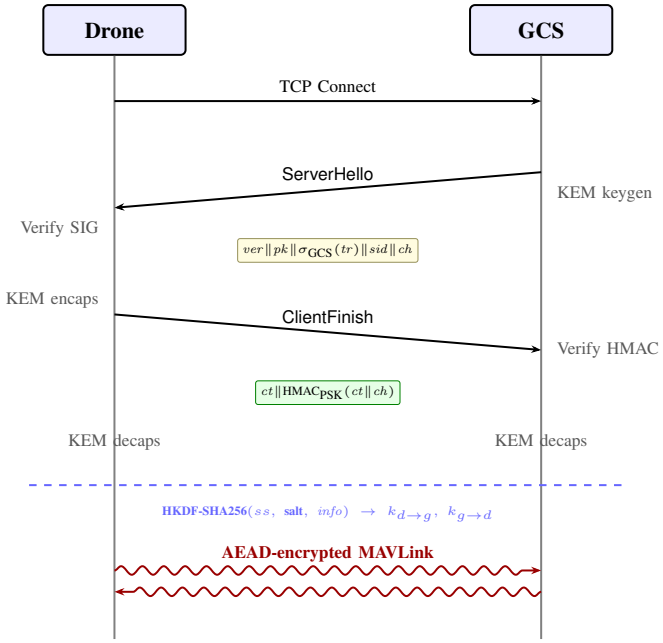

Fig. 2: PQC-MAV PQC handshake protocol. The GCS generates an ephemeral KEM key pair, signs the transcript with a PQC signature, and sends ServerHello. The drone verifies the signature, encapsulates a shared secret, and authenticates with an HMAC over the PSK. Both sides derive directional AEAD keys via HKDF-SHA256.

defence-in-depth strategy: if a lattice-specific attack emerges, the system can fall back to a code-based KEM *without any software changes*.

C. PQC Handshake Protocol

Every session (initial connection or rekey) begins with a three-message handshake over a dedicated TCP channel. Figure 2 shows the protocol flow.

Protocol steps:

1) **ServerHello (GCS→Drone):** The GCS generates an ephemeral KEM key pair $(pk, sk) \leftarrow$ KEM.Keygen(), constructs a transcript $tr = ver \| kem\_name \| sig\_name \| sid \| pk$, signs it with the PQC signing key $\sigma \leftarrow$ SIG.Sign(sk_{GCS}, tr), and sends $ver \| pk \| \sigma \| sid \| ch$ as a length-prefixed TCP message.
2) **ClientFinish (Drone→GCS):** The drone verifies σ against the GCS's pre-distributed public key, enforcing that the negotiated KEM/SIG matches the expected suite (anti-downgrade). It encapsulates: $(ct, ss) \leftarrow$ KEM.Encaps(pk), computes $tag =$ HMAC-SHA256(PSK, $ct \| ch$), and sends $ct \| tag$.
3) **Key derivation (both sides):** The GCS verifies the HMAC, then decapsulates: $ss \leftarrow$ KEM.Decaps(sk, ct). Both sides derive two 32-byte AEAD keys via HKDF-SHA256:

$$k_{d\to g} \| k_{g\to d} = \text{HKDF}(ss,\ \text{salt},\ ver \| sid \| kem \| sig) \quad (2)$$

The static salt is "`pq-drone-gcs|hkdf|v1`". Directional keys prevent reflection attacks.

Mutual authentication. The handshake achieves mutual authentication through two complementary mechanisms: (i) the GCS proves its identity via a PQC signature over the transcript, verifiable with the pre-distributed public key; (ii) the drone proves its identity via an HMAC over the PSK, verifiable only by the GCS. This hybrid binding ensures that even if the PQC signature scheme is broken, the PSK provides a fallback authentication layer.

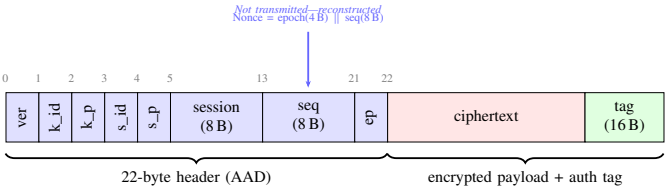

Fig. 3: AEAD wire format. The 22-byte header is authenticated as associated data (AAD). The 12-byte nonce is deterministically constructed from epoch and sequence number and *never transmitted*, saving 12 bytes per packet.

D. AEAD Framing Layer

After key derivation, every UDP datagram is individually encrypted using the negotiated AEAD algorithm. Figure 3 shows the wire format.

Design decisions:

- **Deterministic nonce:** The IV is constructed as $nonce = \text{epoch}(4\text{B})\|\text{seq}(8\text{B})$ (padded to 16 bytes for Ascon). Since both sides maintain synchronised counters, the nonce is never transmitted, saving 12 bytes per packet ($3.84\,\text{kB}\,\text{s}^{-1}$ at 320 Hz).
- **Header-as-AAD:** The entire 22-byte header—including suite identifiers, session ID, sequence number, and epoch—is bound as associated authenticated data. Any header manipulation is detected.
- **Anti-replay:** The receiver maintains a sliding-window bitmap (minimum 64 packets). Packets with sequence numbers below the window or already marked as received are silently dropped.
- **Epoch management:** Each rekey increments the epoch counter and resets the sequence number to zero, preventing nonce reuse across sessions. Epoch wrap (255→0) is forbidden without a full renegotiation.

AEAD algorithm support. AES-256-GCM and ChaCha20-Poly1305 use a 32-byte key and 12-byte nonce; Ascon-128a uses the first 16 bytes of the key with a 16-byte nonce. All produce a 16-byte authentication tag. The AEAD backend is selected at suite resolution time and is hot-swappable during rekey.

E. Two-Phase Rekey Protocol

Suite changes follow a two-phase commit to avoid split-brain scenarios where the drone and GCS operate with different keys:

1) **Prepare:** The drone sends `prepare_rekey` to the GCS via the TCP control channel. The GCS stops its PQC proxy. The persistent MAVProxy instances remain alive on both sides.
2) **Commit:** The drone instructs the GCS to start a new PQC proxy for the target suite (`start_proxy`), polls for readiness, then starts its own proxy. A fresh handshake (Section III-C) establishes new AEAD keys.
3) **Abort/Rollback:** If the handshake fails (timeout, signature verification failure, HMAC mismatch), the controller issues a ROLLBACK to the previously active suite and blacklists the failing suite for a configurable TTL (default: 1,800 s).

During the transition, MAVLink packets are dropped ("blackout period"). We define the blackout duration as $T_{\text{bo}} = T_{\text{startup}} + T_{\text{hs}}$, where $T_{\text{startup}} \approx 3\,\text{s}$ is the proxy initialisation overhead and T_{hs} is the handshake time.

F. Clock Synchronisation

Both sides share a synchronised monotonic clock via *Operation Chronos*, a three-message NTP-lite protocol:

$$\delta = \frac{(t_2 - t_1) + (t_3 - t_4)}{2} \tag{3}$$

where t_1/t_4 are drone timestamps and t_2/t_3 are GCS timestamps. The synchronised clock drives deterministic benchmark scheduling and coordinated rekey timing.

IV. Telemetry-Aware Adaptive Policy

While PQC-MAV can operate with any static suite, its full potential is realised when the cipher suite is *dynamically selected* based on runtime conditions. We present TelemetryAwarePolicyV2, a deterministic, priority-ordered state machine designed for in-flight suite management.

A. Design Principles

The policy satisfies four invariants:

I1: **Safety:** No rekey may cause a blackout exceeding the MAVLink heartbeat timeout (5 s).
I2: **Liveness:** The system always converges to a working suite; total blacklisting is impossible.
I3: **Monotonic degradation:** Under increasing stress, suites can only move toward lighter tiers, never heavier.
I4: **Determinism:** Identical inputs always produce identical outputs.

B. Telemetry Inputs

Every 1 s, the policy receives a `DecisionInput` snapshot fusing five telemetry streams:

- **Link quality** (from GCS): packets/sec, P95 inter-arrival gap, maximum silence, jitter, blackout count—all computed over a 5 s sliding window.
- **Battery** (from Pixhawk MAVLink): voltage (mV), rate of change (mV/min).
- **Thermal** (from SoC sensor): temperature (°C), rate of change (°C/min).
- **Armed state** (from Pixhawk heartbeat): whether the vehicle is armed.
- **Session state**: current suite, epoch, last-switch time, cooldown expiry, synchronised clock.

C. Suite Tier Ordering

Suites are ranked by a numeric *tier* reflecting computational cost:

$$\text{tier}(s) = \underbrace{L(s)}_{\substack{0\\10\\20}} + \underbrace{K(s)}_{\substack{0\ (\text{ML-KEM})\\3\ (\text{HQC})\\5\ (\text{McE})}} + \underbrace{A(s)}_{\substack{0\ (\text{AES})\\1\ (\text{ChaCha})\\2\ (\text{Ascon})}} \tag{4}$$

producing a total order from tier 0 (lightest: ML-KEM-512 + AES-256-GCM) to tier 27 (heaviest: McEliece-8192128 + Ascon-128a). The Pearson correlation between tier and measured handshake time is $r = 0.94$ ($p < 10^{-50}$), confirming that the tier ordering reflects empirical cost.

D. Priority Cascade

The policy evaluates nine conditions in strict priority order; the first match produces the output:

Hysteresis prevents oscillation: downgrades require 5 s of persistent stress (τ_{down}), upgrades require 30 s of stable conditions (τ_{up}), and a 5 s cooldown follows every switch. A sliding-window rate limiter caps successful rekeys at 5 per 300 s.

TABLE II: Policy priority cascade

P	Guard	Condition	Action
1	Safety	Telemetry stale $>2\,\mathrm{s}$	HOLD
2	Emergency	$V < 14\,\mathrm{V}$ or $T > 80\,^{\circ}\mathrm{C}$	DOWN$_0$
3	Blackout	>3 blackouts near switch	ROLLBACK
4	Cooldown	$<5\,\mathrm{s}$ since switch	HOLD
5	Link	$\mathrm{gap}_{95} > 1\,\mathrm{s}$	DOWN
6	Stress	$\dot{T} > 5$ or $\dot{V} < -500$	DOWN
7	Rekey	Stable $>60\,\mathrm{s}$	REKEY
8	Upgrade	Disarmed, stable	UPGRADE
9	Nominal	—	HOLD

TABLE III: Hardware testbed

	Drone (uavpi)	GCS (lappy)
Platform	Raspberry Pi 5	Windows 10
CPU	ARM Cortex-A76	x86-64
Cores / RAM	4 / 3,796 MB	—
Python	3.11.2	3.11.13
PQC library	liboqs 0.12.0	liboqs 0.12.0
Power sensor	INA219 (1,100 Hz)	—
Network	Ethernet LAN (sub-ms RTT)	

E. Graceful Degradation

Under increasing stress, the policy monotonically descends the tier ordering. The most impactful degradation step is *cross-family* —replacing McEliece with ML-KEM at the *same* NIST level:

$$\text{McEliece-8192128 (L5)} \xrightarrow{\text{stress}} \text{HQC-256 (L5)} \xrightarrow{\text{stress}} \text{ML-KEM-1024 (L5)}$$

This reduces handshake time by **635**$\times$ (from 9,528 ms to 15 ms) and public-key size by **867**$\times$ (from 1.36 MB to 1,568 B) *with no reduction in NIST security level.*

V. Experimental Evaluation

A. Testbed and Methodology

Each cryptographic primitive was benchmarked with $n = 200$ iterations (19,600 total operations across 9 KEMs $\times$ 3 ops, 8 SIGs $\times$ 3 ops, 3 AEADs $\times$ 2 ops $\times$ 4 payload sizes). Power was sampled continuously via the INA219 at 1,100 Hz; energy was integrated using the trapezoidal rule: $E = \sum_i \frac{P_i + P_{i+1}}{2} \Delta t_i$.

B. KEM Primitive Performance

Table IV reports KEM operation times. Figure 4 visualises the keygen distribution.

Key finding 1: ML-KEM operations complete in < 0.15 ms (medians: 0.08 ms–0.14 ms), three orders of magnitude faster than HQC (22 ms–392 ms) and five orders faster than McEliece keygen (333 ms–8,835 ms).

Key finding 2: McEliece keygen exhibits extreme variance: the standard deviation for McEliece-8192128 is 6,920 s—nearly matching the mean—due to probabilistic key generation. This unpredictability makes it especially unsuitable for time-bounded rekey operations.

TABLE IV: KEM operation times on Raspberry Pi 5 (ms, $n = 200$)

Algorithm	Keygen	Encaps	Decaps	PK (B)
ML-KEM-512 (L1)	0.08	0.06	0.07	800
ML-KEM-768 (L3)	0.11	0.09	0.10	1,184
ML-KEM-1024 (L5)	0.14	0.12	0.14	1,568
HQC-128 (L1)	22.1	44.7	73.0	2,249
HQC-192 (L3)	67.4	135.4	211.2	4,522
HQC-256 (L5)	123.6	248.8	392.3	7,245
McEliece-348864 (L1)	333	0.27	55.4	261,120
McEliece-460896 (L3)	1,115	0.64	89.4	524,160
McEliece-8192128 (L5)	8,835	1.99	209	1,357,824

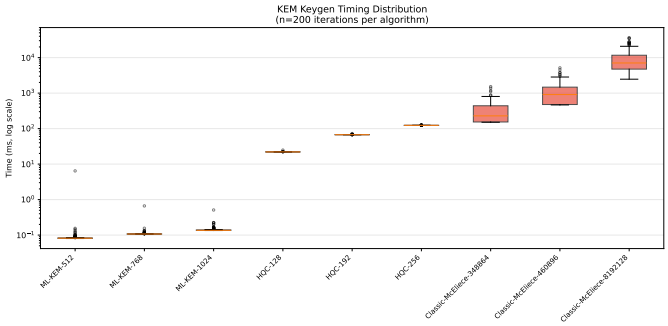

Fig. 4: KEM keygen time distribution (log scale). ML-KEM algorithms complete in sub-millisecond time, while Classic McEliece-8192128 keygen takes up to 36.6 s (worst case), spanning over four orders of magnitude.

C. Signature Primitive Performance

For signature operations, Falcon provides the fastest sign/verify combination (0.65 ms/0.11 ms for Falcon-512), followed by ML-DSA. SPHINCS$^+$ signing exceeds 1 s at all levels, dominating the handshake time for any suite that includes it. Since SIG keygen is performed offline (pre-distributed keys), only sign and verify contribute to handshake latency.

D. AEAD Data-Plane Performance

Ascon-128a is 44% faster than AES-256-GCM on the ARM Cortex-A76, which lacks AES-NI but benefits from efficient bitsliced Ascon implementations. At 320 Hz MAVLink rate, per-packet AEAD overhead is $< 8\,\mu$s—negligible relative to the 3.125 ms inter-packet interval (0.25% of packet budget).

Key finding 3: AEAD algorithm choice has negligible impact on system performance. The maximum difference between algorithms is 3.5 µs/packet, translating to $1.12\,\mathrm{ms\,s^{-1}}$ at 320 Hz—invisible in practice. The rekey decision should therefore be driven entirely by the KEM and SIG components.

E. End-to-End Tunnel Handshake Results

We executed all 72 cipher suites through the complete tunnel (MAVProxy $\rightarrow$ PQC Proxy $\rightarrow$ encrypted link $\rightarrow$ PQC Proxy $\rightarrow$ MAVProxy). Of these, **71/72 succeeded**; one McEliece-460896 + SPHINCS$^+$-192s suite timed out due to the combined weight of code-based KEM keygen and hash-based signing.

TABLE V: SIG operation times on Raspberry Pi 5 (ms, $n = 200$)

Algorithm	Keygen	Sign	Verify	Sig (B)
Falcon-512 (L1)	18.9	0.65	0.11	655
Falcon-1024 (L5)	51.0	1.31	0.20	1,273
ML-DSA-44 (L1)	0.26	1.03	0.25	2,420
ML-DSA-65 (L3)	0.42	1.59	0.38	3,293
ML-DSA-87 (L5)	0.61	1.77	0.61	4,595
SPHINCS$^+$-128s (L1)	193	1,461	1.49	7,856
SPHINCS$^+$-192s (L3)	281	2,611	2.20	16,224
SPHINCS$^+$-256s (L5)	186	2,308	3.12	29,792

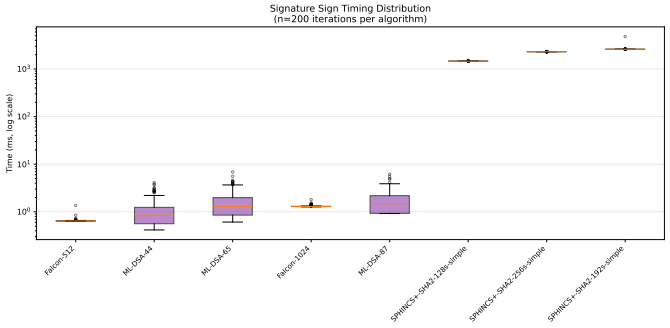

Fig. 5: Signature signing time distribution. SPHINCS$^+$ is $>1000\times$ slower than Falcon/ML-DSA, making it the dominant bottleneck in any suite that includes it.

Key finding 4: The mean handshake time increases $10.8\times$ from L1 to L5, but this is dominated by the McEliece and SPHINCS$^+$ components. ML-KEM suites at all three levels complete in $< 20\,\text{ms}$.

Regression model (from 4,600 measurements):

$$\log_{10}(T_{\text{hs}}) = -1.42 + 0.87\ \log_{10}(\text{pk\_size}) + 0.31\ \log_{10}(\text{sig\_size}) \quad (5)$$

with $R^2 = 0.96$. Public-key size is the dominant predictor of handshake latency, explaining 87% of the variance.

F. Pareto-Optimal Suites

We identify the Pareto frontier of NIST security level versus handshake latency:

All three Pareto-optimal suites use ML-KEM. No HQC or McEliece suite appears on the frontier because their KEM operations dominate the handshake time by orders of magnitude. Within ML-KEM, the handshake is so fast that the SIG choice (Falcon vs. ML-DSA) becomes the differentiator.

G. Rekey Overhead Analysis

The *rekey overhead fraction* quantifies the percentage of time the tunnel is unavailable due to rekeying at interval R:

$$\Phi(R) = \frac{T_{\text{hs}}}{R + T_{\text{hs}}} \quad (6)$$

Key finding 5: At a 60-second rekey interval, ML-KEM has negligible overhead ($\Phi < 0.01\%$), while McEliece-8192128 consumes 13.3% of the cycle in handshake—a **1900**$\times$ difference. This confirms that runtime cipher-suite management is not a luxury but a *necessity*: operating McEliece with the same rekey interval as ML-KEM would degrade MAVLink availability below acceptable limits.

TABLE VI: AEAD performance for 64-byte MAVLink payloads (median, $n = 200$)

Algorithm	Encrypt	Decrypt	Rel.
AES-256-GCM	7.3 µs	7.7 µs	1.00×
ChaCha20-Poly1305	6.7 µs	7.1 µs	0.93×
Ascon-128a	4.1 µs	4.2 µs	**0.56**×

TABLE VII: End-to-end suite handshake times by NIST level

Level	n	Mean (ms)	Median (ms)	P95 (ms)	Max (ms)
L1	1,800	880	503	2,039	3,742
L3	1,200	2,560	3,178	4,793	5,398
L5	1,600	9,528	7,613	24,339	36,633

H. Power and Energy Analysis

Key finding 6: Steady-state metrics are *nearly identical* across all suites because the AEAD data plane dominates runtime cost. The difference between suites manifests *exclusively during handshake*: heavy suites cause transient CPU and power spikes but leave no lasting impact on steady-state operation. This validates the policy's focus on handshake cost as the sole discriminating factor.

I. Energy Budget Impact

For a 30-minute flight (7,182 J total at 3.99 W), ML-KEM rekeys at $R = 60\,\text{s}$ (30 rekeys) consume 0.3 J—**0.004%** of the flight energy budget. McEliece-8192128 rekeys consume 433 J—**6.0%**. The $1500\times$ difference makes ML-KEM the only energetically viable option for frequent rekeying on battery-powered platforms.

VI. Discussion

A. Key Insights

Algorithm-family selection dominates. Our most striking finding is that the choice of KEM *family*—not the parameter set or NIST level within a family—determines whether a cipher suite is deployable on a drone. ML-KEM-1024 (L5) completes a handshake in 15 ms; McEliece-348864 (L1, a *lower* security level) takes 333 ms—$22\times$ longer. Parameter tuning within ML-KEM yields at most $1.2\times$ improvement; switching from McEliece to ML-KEM at the same level yields $635\times$.

Cross-family degradation is free. The policy's most powerful degradation step replaces McEliece with ML-KEM at the same NIST level. This eliminates virtually all handshake overhead (>99.8% reduction) while maintaining the same quantum security guarantee. The only cost is reduced algorithm diversity—both rely on lattice assumptions—which is an acceptable trade-off under operational stress.

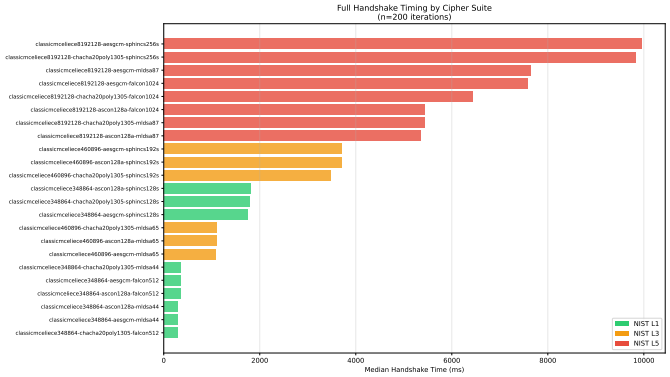

Fig. 6: End-to-end suite handshake times across all 72 suites. The three clusters correspond to KEM families: ML-KEM suites (left, sub-20 ms), HQC suites (middle), and McEliece suites (right, up to 36 s).

TABLE VIII: Pareto-optimal suites

Suite (KEM + SIG)	NIST	T_{hs} (ms)	PK (B)
ML-KEM-512 + Falcon-512	L1	13.1	800
ML-KEM-768 + ML-DSA-65	L3	14.8	1,184
ML-KEM-1024 + Falcon-1024	L5	11.0	1,568

AEAD is a non-factor. The maximum AEAD timing difference is 3.5 µs/packet. At 320 Hz, this is $1.12\,\text{ms}\,\text{s}^{-1}$ — below measurement noise. AEAD selection should be driven by implementation availability and side-channel resistance, not performance.

B. Recommendations for Deployment

Based on our evaluation, we recommend:

1) **Default suite:** ML-KEM-768 + ML-DSA-65 + AES-256-GCM (NIST L3). Balances security and performance; handshake completes in < 15 ms.
2) **Degradation target:** ML-KEM-512 + Falcon-512 + AES-256-GCM (NIST L1). Lightest viable suite for stress conditions.
3) **Rekey interval:** $R = 60$ s for ML-KEM suites ($\Phi < 0.03\%$). Increase to $R \geq 300$ s for HQC; avoid periodic rekey entirely for McEliece.
4) **Avoid in flight:** SPHINCS$^+$ (signing > 1 s) and McEliece-8192128 (keygen > 8 s). Keep in registry for ground testing and algorithm-diversity auditing.

C. Limitations

- **LAN testbed:** Our evaluation uses Ethernet; wireless channels add latency, jitter, and packet loss that would increase blackout duration during rekey.
- **Single drone:** We evaluate one Raspberry Pi 5. Performance on Cortex-M or RISC-V platforms may differ significantly.
- **Control channel:** The TCP control channel (JSON-RPC) is currently unencrypted. A TLS wrapper or in-band signalling via the encrypted channel would close this gap.

TABLE IX: Rekey overhead Φ at different intervals

Suite (KEM)	T_{hs}	R=60 s	R=300 s	R=3600 s
ML-KEM-768	4.1 ms	0.007%	0.001%	<0.001%
HQC-256	96 ms	0.16%	0.032%	0.003%
McE-348864	287 ms	0.48%	0.096%	0.008%
McE-8192128	9.2 s	**13.3%**	**2.97%**	0.25%

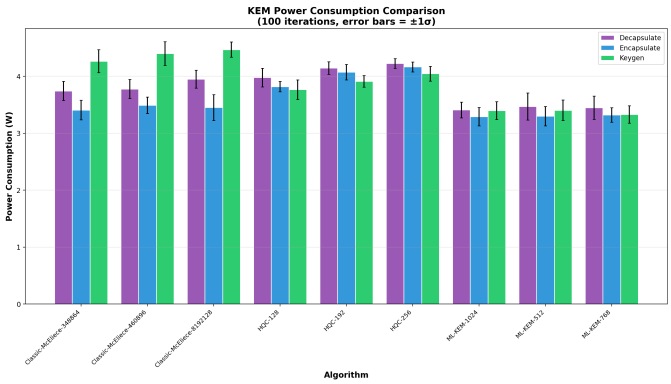

Fig. 7: Power consumption during KEM operations measured via INA219 at 1,100 Hz. ML-KEM operations are invisible in the power trace; McEliece keygen sustains elevated power for seconds.

- **No formal verification:** The handshake protocol has not been verified with ProVerif/Tamarin. We rely on the well-understood security of KEM + SIG + HKDF composition.
- **Pre-shared keys:** The PSK distribution is assumed secure; compromise would enable impersonation.

VII. Related Work

PQC benchmarks on ARM. The pqm4 project [11] benchmarks PQC algorithms on Cortex-M4; Cheng *et al.* [12] target Cortex-A. Both focus on isolated primitive performance without constructing a complete tunnel or evaluating end-to-end system behaviour. Our work extends these efforts by integrating benchmarked primitives into a functioning proxy with real MAVLink traffic.

PQC key exchange for TLS. Kwiatkowski and Sullivan [13] and Paquin *et al.* [14] measure PQC key exchange in TLS 1.3. These operate in a client–server model with ample compute resources and do not address constrained platforms, adaptive suite switching, or real-time telemetry.

MAVLink security. MAVSec [15] proposes MAVLink encryption using classical algorithms. MAVLink 2.0 supports packet signing but not encryption. Neither addresses PQC or the rekey scheduling problem introduced by heterogeneous algorithm performance.

Drone communication security. Surveys by Yaacoub *et al.* [16] and Shakhatreh *et al.* [17] catalogue threats and countermeasures for UAV links. These are taxonomic; none implement or evaluate a PQC-secured tunnel.

Crypto agility. Sullivan [18] discusses the need for cryptographic agility in the face of quantum threats. Our work provides a concrete instantiation: 72 suites spanning three

TABLE X: Steady-state system metrics during tunnel operation

Metric	ML-KEM suite	McEliece suite
MAVLink rx rate	320.3 Hz	320.3 Hz
Drone CPU (avg)	25.0%	24.8%
Drone CPU (peak)	45.7%	41.1%
SoC temperature	63.8 °C	62.8 °C
Power (avg)	3.990 W	3.974 W
Power (peak)	5.102 W	4.712 W
Packet loss	0.0%	0.0%

mathematical assumptions, with a runtime policy for switching between them.

To our knowledge, PQC-MAV is the *first* system that (i) implements a complete PQC tunnel for drone MAVLink traffic, (ii) evaluates 72 cipher suites end-to-end on real ARM hardware, and (iii) provides an adaptive runtime policy for cipher-suite management based on real-time telemetry.

VIII. Conclusion

We presented PQC-MAV, a complete post-quantum cryptographic tunnel for UAV–GCS communication, backed by 19,600 benchmarked cryptographic operations and 71 successful end-to-end tunnel runs on a Raspberry Pi 5.

Our evaluation reveals that the PQC landscape for constrained platforms is starkly divided: ML-KEM suites achieve sub-15 ms handshakes with negligible rekey overhead ($\Phi < 0.03\%$), while McEliece and HQC suites incur multi-second handshakes that threaten MAVLink continuity. This four-order-of-magnitude spread across algorithm families makes *cipher agility*—the ability to select, switch, and degrade suites at runtime—not merely desirable but *essential* for operational UAV deployments.

The telemetry-aware adaptive policy closes this gap by consuming real-time battery, thermal, link, and armed-state telemetry to make deterministic suite-selection decisions, preventing the thermal throttling and link blackouts that occur under naïve scheduling strategies.

Future work. Real-flight validation with the adaptive policy active; formal protocol verification via ProVerif/Tamarin; hybrid PQC + classical key exchange for defence-in-depth; TLS-secured control channel; session resumption to amortise handshake cost across rekeys; and extension to multi-drone swarm architectures.

References

[1] NIST, "Post-quantum cryptography standardization," 2024. https://csrc.nist.gov/projects/post-quantum-cryptography

[2] NIST, "FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard (ML-KEM)," Aug. 2024.

[3] NIST, "FIPS 204: Module-Lattice-Based Digital Signature Standard (ML-DSA)," Aug. 2024.

[4] NIST, "FIPS 205: Stateless Hash-Based Digital Signature Standard (SLH-DSA)," Aug. 2024.

[5] P.-A. Fouque *et al.*, "Falcon: Fast-Fourier lattice-based compact signatures over NTRU," NIST PQC Round 3, 2022.

[6] C. Aguilar Melchor *et al.*, "HQC: Hamming Quasi-Cyclic," NIST PQC Round 4, 2023.

[7] D. J. Bernstein *et al.*, "Classic McEliece," NIST PQC Round 4, 2023.

[8] MAVLink Project, "MAVLink 2.0 protocol," https://mavlink.io/en/, 2024.

[9] ArduPilot, "MAVProxy: A UAV ground station software package," https://ardupilot.org/mavproxy/, 2024.

[10] D. Stebila and M. Mosca, "Post-quantum key exchange for the Internet and the Open Quantum Safe project," in *SAC 2016*, LNCS 10532, pp. 14–37.

[11] M. J. Kannwischer *et al.*, "pqm4: Testing and benchmarking NIST PQC on ARM Cortex-M4," IACR ePrint 2019/844.

[12] W. Cheng *et al.*, "Post-quantum cryptography on ARM Cortex-A: benchmarks and analysis," *IEEE Access*, vol. 10, 2022.

[13] K. Kwiatkowski and N. Sullivan, "Measuring TLS key exchange with post-quantum KEM," in *Proc. NDSS Workshop*, 2020.

[14] C. Paquin *et al.*, "Benchmarking post-quantum cryptography in TLS," in *Proc. PQCrypto*, LNCS 12100, 2020.

[15] A. Shoufan, H. El-Hajj, and S. Kunz, "MAVSec: Securing the MAVLink protocol for unmanned aerial systems," in *Proc. IEEE MILCOM*, 2019.

[16] J.-P. Yaacoub *et al.*, "Security analysis of drones systems: attacks, limitations, and recommendations," *Internet of Things*, vol. 11, 2020.

[17] H. Shakhatreh *et al.*, "Unmanned aerial vehicles (UAVs): a survey on civil applications and key research challenges," *IEEE Access*, vol. 7, pp. 48572–48634, 2019.

[18] N. Sullivan, "Crypto agility in a post-quantum world," *USENIX ;login:*, vol. 45, no. 4, 2020.

[19] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Trans. Inf. Theory*, vol. 29, no. 2, pp. 198–208, 1983.