Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Sequence-, Epoch-, and Integrity-Aware Detection of Replay and Peer-Spoofing Attacks in Decentralized UAV Mesh Networks: A Software-Validated Security Framework

Md Shahanur Islam Shagor

shagormd.s.i@iunp.edu.rs

Voronezh State University of Forestry and Technologies https://orcid.org/0009-0003-7730-3202

Research Article

Keywords: UAV cybersecurity, flying ad hoc networks, replay attack, peer spoofing, HMAC-SHA256, trust epoch, sequence number, packet authentication, edge swarm

Posted Date: August 26th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10812593/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Sequence-, Epoch-, and Integrity-Aware Detection of Replay and Peer-Spoofing Attacks in Decentralized UAV Mesh Networks: A Software-Validated Security Framework

Md Shahanur Islam Shagor
Independent Researcher, Voronezh State University of Forestry and Technologies, Voronezh, Russia
Corresponding author: shagormd.s.i@iunp.edu.rs

Abstract

Decentralized unmanned aerial vehicle (UAV) swarms depend on rapidly changing peer-to-peer links in which a stale but previously valid state, a forged sender identity, or an unauthenticated emergency message can influence collective behavior before a centralized authority can intervene. This paper develops and analyzes a packet-admission framework for replay and peer-spoofing resistance that is grounded in the implemented edge-swarm security path of an open GPS-denied UAV research platform. Each peer packet is represented by sender identity, timestamp, time-to-live, trust epoch, sequence number, source tag, payload, and authentication hook. The implemented admission chain canonicalizes the packet with the authentication field removed, authenticates the canonical CBOR representation with a full HMAC-SHA256 tag, enforces a bounded timestamp/TTL window, requires equality with the configured trust epoch, and rejects any sequence number that is not strictly greater than the last authenticated value. Authentication failure occurs before state-cache mutation or safety-relevant peer action. We formalize this compound predicate, derive replay-rejection, tamper-rejection, epoch-isolation, and pre-action authorization properties, and quantify the operational trade-off introduced by strict monotonic sequencing under packet reordering. Source-level tests cover valid HMAC verification, payload and sender tampering, missing signatures, simulation-only unsigned exceptions, stale epochs, stale sequences, unsupported post-quantum placeholders, cache-update suppression, and authentication of emergency-corridor messages. A repository benchmark reports a swarm-authentication path median of 12.219 ms and mean of 22.603 ms, but the harness includes process-launch overhead and is therefore treated only as a coarse software-path proxy rather than isolated cryptographic latency. The analysis also identifies the principal residual risk: a shared-secret member that is fully compromised can generate valid HMACs and impersonate other identifiers. The resulting framework provides an evidence-bounded baseline for lightweight anti-replay and peer-spoof protection and a clear migration path toward per-node keys, replay windows, hardware-bound identity, and post-quantum authentication.

Keywords: UAV cybersecurity; flying ad hoc networks; replay attack; peer spoofing; HMAC-SHA256; trust epoch; sequence number; packet authentication; edge swarm

1 Introduction

Decentralized UAV swarms exchange state, health, consensus, obstacle, and emergency information over wireless links that are inherently observable and injectable by nearby adversaries. Unlike a conventional client-server application, a flying ad hoc network (FANET) may change topology on the order of seconds, operate with intermittent infrastructure, and make time-sensitive collective decisions using peer messages. These characteristics turn message freshness and peer identity into control-system properties rather than purely network-security concerns. A packet that is cryptographically valid but stale can be dangerous if it reintroduces a previous formation command; a packet with a forged sender identifier can corrupt peer state or leader selection; and an unauthenticated emergency message can create a false safety corridor or halt condition.

Recent surveys of UAV and FANET security consistently identify impersonation, replay, false-data injection, routing manipulation, and compromised-node behavior as persistent threats [1,2]. Authentication research has therefore explored physical-unclonable functions, lightweight challenge-response protocols, dynamic keying, and distributed ledgers [3-5]. At the same time, practical UAV communication stacks such as MAVLink 2 employ message signing and monotonic timestamps to reject forged and replayed traffic [10]. The broader networking literature has long treated anti-replay as a stateful receiver function, as exemplified by IPsec sequence numbers and sliding windows [12,13].

However, three implementation-level problems remain important for resource-constrained swarms. First, freshness cannot safely be reduced to wall-clock time alone. Time synchronization can drift, GNSS-derived time can be manipulated, and recent work has shown that timestamp assumptions in UAV communication protocols can themselves become an attack surface [6]. Second, message authentication must bind not only the payload but also the identity, epoch, sequence, and policy-relevant metadata; otherwise an attacker may preserve a valid payload while modifying routing or authority context. Third, authentication must occur before the message mutates peer state or reaches a safety-relevant action. A correct cryptographic primitive placed after state mutation does not provide a correct security boundary.

This paper studies these issues through the peer-packet authentication path of an open GPS-denied UAV swarm platform at commit c040073969a6111f4c652d08e8de8629d70b2317 [16]. The implementation supports an HMAC-SHA256 mode for edge-swarm packets, along with runtime-profile policy, timestamp/TTL screening, exact trust-epoch validation, strictly monotonic sequence validation, canonical CBOR hashing, and constant-time comparison of the supplied and recomputed MAC. Ed25519 and post-quantum hybrid modes exist only as explicit placeholders and return unsupported in the studied build; they are not treated as implemented security.

The paper is intentionally narrower than a general zero-trust or runtime-resilience study. It focuses on the packet admission boundary: what the receiver can establish about authenticity, freshness, epoch membership, and replay status before accepting peer state. This creates a distinct research contribution from higher-level command authorization or firmware trust. The goal is not to claim protection against every compromised-node attack. Instead, the paper asks what can be proven from the current shared-secret design, what can be demonstrated from the repository tests, and where the design must evolve to contain an insider that already possesses the group secret.

The contributions are sixfold. First, a formal packet and adversary model is defined for replay, tamper, sender-spoof, epoch-reuse, unsigned-injection, and insider-key-compromise attacks. Second, the implemented packet-admission chain is expressed as a compound mathematical predicate over time, trust epoch, sequence number, authentication mode, and HMAC integrity. Third, replay, sender-tamper, and pre-action cache-safety properties are derived under explicit assumptions. Fourth, the operational cost of strict monotonic sequencing is analyzed, including out-of-order false rejection and 32-bit sequence exhaustion. Fifth, source-level test cases and repository validation evidence are mapped to the formal properties without inventing unexecuted experiments. Sixth, the residual shared-secret compromise boundary is characterized and translated into concrete design requirements for per-node keys, hardware-bound identity, bounded replay windows, and future asymmetric or post-quantum authentication.

2 Background and Related Work

2.1 UAV and FANET security context

Ceviz, Sen, and Sadioglu provide a broad attack-surface analysis of UAVs and FANETs, emphasizing that dynamic topology, wireless exposure, and decentralized cooperation create both outsider and insider threats [1]. Kundu et al. similarly survey trust-based FANET mechanisms and argue that node reputation and trust computation become important when cooperative routing and coordination depend on potentially untrustworthy peers [2]. These surveys motivate a layered view: cryptographic admission is necessary to reject packets that cannot be authenticated, while behavioral or trust mechanisms are needed for insiders that remain cryptographically valid.

Authentication schemes proposed for UAV networking increasingly combine lightweight identity with freshness mechanisms. Sen, Al-Rubaye, and Tsourdos develop a PUF-oriented UAV authentication framework and explicitly analyze replay and masquerade resistance [3]. Chen et al. combine PUF-based dynamic secret keys with hierarchical blockchain to reduce secret exposure under node capture [4]. Liu et al. propose an anonymous authentication and key-exchange protocol for FANETs and report formal resistance to identity spoofing and replay attacks [5]. These approaches provide stronger identity separation than a group-HMAC design, but they also introduce registration, hardware, public-key, or key-management assumptions that are not present in the current software baseline studied here.

Runtime monitoring is a complementary line of work. R2U2 demonstrates that security and safety properties can be monitored onboard a UAS and diagnosed at runtime [7]. More recent Springer Nature reviews catalog machine-learning-based attack detection, including spoofing and intrusion detection, but also highlight computational and

deployment challenges for resource-constrained UAV platforms [8,9]. The present work deliberately remains below that layer: it uses deterministic protocol invariants rather than statistical anomaly detection for attacks that can be rejected directly at packet admission.

2.2 Replay protection in operational protocols

MAVLink 2 message signing is a useful operational reference because it binds a shared secret to packet content and a monotonically increasing timestamp and rejects messages that are older than the last accepted value for a logical stream [10]. This design recognizes that a valid signature alone is insufficient for replay resistance. At the same time, Colton, Oracevic, and Dilek show that timestamp manipulation can become a practical weakness when the time source itself is attackable [6]. The lesson is not that timestamps are useless, but that freshness is more robust when independent dimensions such as sequence state and epoch state are also enforced.

IPsec ESP defines anti-replay protection using monotonically increasing sequence numbers and receiver state [12]. RFC 6479 refines the implementation of a sliding anti-replay window so that limited packet reordering can be tolerated without reopening previously authenticated sequence positions [13]. The edge-swarm implementation studied here currently adopts the simpler strict-monotonic rule: a packet is rejected whenever its sequence number is less than or equal to the last accepted sequence for that peer. This has a clean safety invariant but can reject legitimately reordered packets. Section 5 formalizes that trade-off and Section 9 discusses a bounded-window generalization.

2.3 Research gap

Existing UAV authentication literature often evaluates a complete protocol as a monolithic object. In contrast, this study isolates a smaller but operationally important boundary: a receiver-side admission gate that combines canonical message authentication, profile policy, time freshness, trust epoch, and sequence state before any cache mutation or peer action. The contribution is therefore an implementation-grounded security model and evidence map rather than a new cryptographic primitive. This narrower framing makes it possible to state exactly which attack classes the current code can reject and exactly which insider threats remain unresolved.

3 System Model and Threat Model

3.1 Packet model

Let an edge-swarm packet from peer i be represented as the tuple

$$p_i = (\tau_i, u_i, t_i, \ell_i, e_i, s_i, \sigma_i, m_i, a_i) \tag{1}$$

where τ_i is the packet type, u_i the sender identifier, t_i the sender timestamp in milliseconds, ℓ_i the packet time-to-live, e_i the trust epoch, s_i the sequence number, σ_i the source tag, m_i the structured payload, and a_i the authentication hook. The implementation serializes the canonical packet as CBOR after clearing a_i. Consequently, identity, timestamp, sequence, epoch, source, TTL, packet type, and payload are all covered by the authentication calculation rather than authenticating the application payload alone.

Field	Security role	Failure consequence
sender_id u_i	Peer identity bound into canonical MAC	Identity spoof or cache pollution if mutable
timestamp t_i	Wall-clock freshness signal	Stale or future packet acceptance if unchecked
ttl ℓ_i	Per-packet validity horizon	Excessively old message acceptance
trust_epoch e_i	Membership/key-generation context	Cross-epoch replay if not enforced
sequence s_i	Per-peer anti-replay state	Duplicate or reordered stale state acceptance
source σ_i	Runtime-profile policy context	Unsigned simulation packet leaking into edge mode
payload m_i	Operational state/action content	False state or safety action
auth_hook a_i	HMAC tag or policy marker	Forgery accepted if verification omitted

3.2 Adversary classes

The analysis distinguishes outsider and insider capabilities. This distinction is essential because a shared-secret HMAC has strong integrity properties against an adversary that does not know the secret, but it cannot attribute a valid tag to a unique physical UAV when every authorized member holds the same group secret.

Adversary	Capability	Target	Current gate
Passive recorder	Capture valid packets and replay later	State rollback / stale commands	Time + sequence + epoch
Payload tamperer	Modify payload after capture	False health/obstacle/consensus data	HMAC
Sender spoofer without key	Change sender_id or construct fake identity	Peer-cache poisoning / false authority	HMAC
Epoch replayer	Replay valid packet from previous epoch	Old membership/key context	Trust-epoch equality
Unsigned injector	Transmit unauthenticated packet	Edge-state mutation	Runtime-profile policy
Clock manipulator	Shift apparent packet time	Freshness bypass	Sequence + epoch provide independent gates
Compromised authorized peer	Possess group secret and generate valid HMAC	Insider impersonation or fabricated state	Not fully contained by group HMAC

3.3 Security assumptions and non-assumptions

The proofs below assume that HMAC-SHA256 behaves as a secure message-authentication code, the configured secret is unknown to the outsider adversary, the receiver protects its last accepted sequence state during a running session, and the packet is admitted to the peer-state cache only after verification succeeds. These assumptions correspond to the current software flow and standard HMAC security expectations [11,14].

The paper does not assume that a shared secret provides non-repudiation or per-node cryptographic identity. It does not assume that the edge-swarm transport provides confidentiality. It does not claim resistance to radio jamming, denial of service, physical key extraction, or a fully compromised peer that already knows the group secret. Ed25519 and post-quantum modes are explicitly excluded from implemented claims because the studied source marks them as placeholders that return UNSUPPORTED.

4 Sequence-, Epoch-, and Integrity-Aware Packet Admission

4.1 Canonicalization and HMAC binding

Let C(p) denote the canonical CBOR encoding of p after clearing the authentication hook. The implemented canonicalization is

$$\mathrm{C}(\mathrm{p}) = \mathrm{CBOR}(\mathrm{p} \mid \mathrm{a} := \emptyset) \tag{2}$$

For HMAC mode with shared secret K, the sender computes

$$\mathrm{h}(\mathrm{p}) = \mathrm{HMAC\text{-}SHA256}(\mathrm{K}, \mathrm{C}(\mathrm{p})) \tag{3}$$

and encodes the full 32-byte tag in hexadecimal form. The receiver decodes the supplied tag, recomputes h(p), and uses a constant-time byte comparison. Because C(p) includes sender identity and the packet metadata, changing sender_id after signing is equivalent to changing the authenticated message and therefore invalidates the tag under the outsider model.

4.2 Time and TTL freshness

Let T be the receiver time, Δ the configured maximum clock skew, and ℓ the packet TTL. The implementation rejects a packet when the absolute age exceeds the sum of the clock-skew allowance and TTL. Define

$$\mathrm{F_time}(\mathrm{p}) = 1\{\,|\mathrm{T} - \mathrm{t}| \le \Delta + \ell\,\} \tag{4}$$

The default edge authentication configuration uses Δ = 2000 ms, while test packets carry their own TTL. This rule tolerates bounded clock disagreement but does not make the wall clock a sole replay oracle. A packet that remains inside the time window must still pass epoch and sequence validation.

4.3 Trust-epoch isolation

Let e* be the receiver-configured trust epoch. The epoch gate is

$$F_epoch(p) = 1\{ e = e^* \} \tag{5}$$

A packet associated with a different epoch is rejected before HMAC verification. Operationally, the epoch acts as a coarse generation boundary for membership or key context. Its security value depends on disciplined epoch management: rotating a key without changing the relevant epoch can preserve an unnecessary replay relationship, whereas changing the epoch creates an immediate rejection boundary for packets labeled with the previous generation.

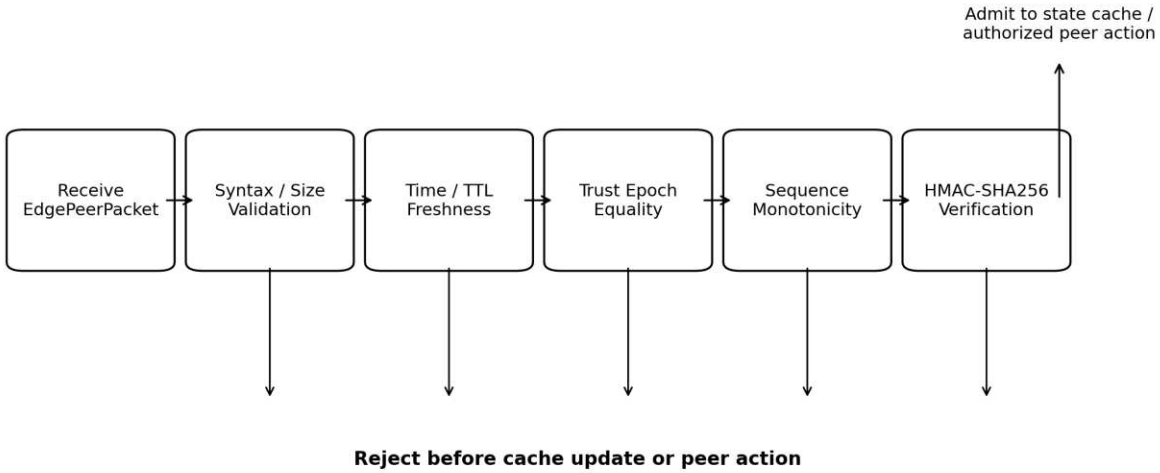


Fig. 1 Receiver-side packet admission chain. Every reject branch terminates before peer-state cache mutation or safety-relevant peer action.

4.4 Strict sequence monotonicity

Let S_u be the largest authenticated sequence previously accepted from peer identity u. The current receiver rule is

$$F_seq(p) = 1\{ S_u \text{ is undefined} \vee s > S_u \} \tag{6}$$

Thus an equal or lower sequence is classified as REPLAY_DETECTED. This is stronger than merely checking for exact duplicates because every backward movement is rejected. It also means the current implementation does not tolerate legitimate packet reordering once a higher sequence has been authenticated. That design choice is favorable for state machines whose newest packet dominates older state, but it must be acknowledged when transport behavior can reorder messages.

4.5 Runtime-profile policy

Unsigned packets are not universally accepted. Simulation mode may permit unsigned simulation-source packets when explicitly configured, whereas edge-swarm mode rejects missing authentication. This creates a policy predicate F_profile(p) that prevents a permissive development rule from silently propagating into the hardened swarm runtime.

4.6 Compound acceptance predicate

For the implemented HMAC edge-swarm path, receiver acceptance can be represented as

$$A(p) = F_time(p) \cdot F_epoch(p) \cdot F_seq(p) \cdot F_profile(p) \cdot F_HMAC(p) \tag{7}$$

where each factor is binary and F_HMAC(p)=1 only when the supplied tag equals the recomputed HMAC under constant-time comparison. The multiplicative form emphasizes fail-closed composition: one failed predicate is sufficient to reject the packet.

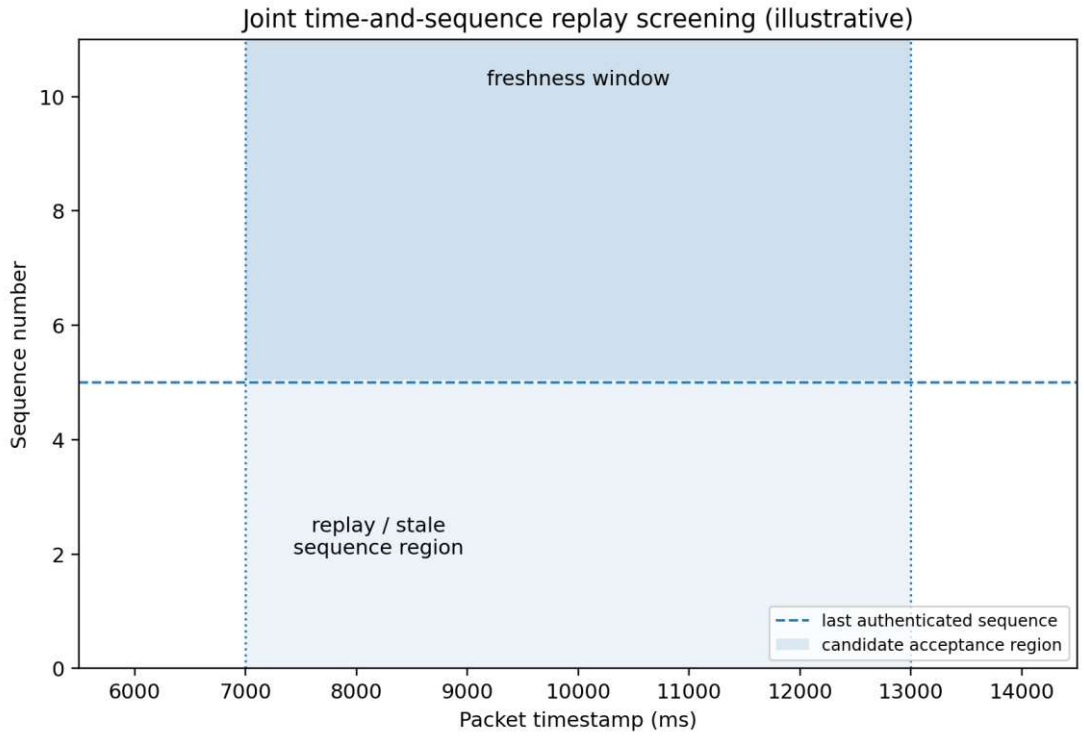


Fig. 2 Illustrative joint replay screen. A fresh timestamp is necessary but not sufficient; a packet must also advance the per-peer sequence state.

5 Formal Security Properties

5.1 Payload- and sender-tamper rejection

Assume the adversary does not know K. If any authenticated field $x \in \{\tau,u,t,\ell,e,s,\sigma,m\}$ is modified after a valid tag is created, the canonical representation changes from C(p) to C(p′). Acceptance then requires a successful MAC forgery.

$$\Pr[A(p')=1 \mid C(p')\neq C(p), K \text{ unknown}] \leq \text{Adv_HMAC}(q) \tag{8}$$

Under the standard idealized bound for a full 256-bit tag, the generic online-guessing component is on the order of $q/2^{256}$ for q verification attempts, while practical security also depends on key quality, key exposure, implementation correctness, and the cryptographic assumptions underlying HMAC [11,14]. The repository test that signs a packet and then changes sender_id demonstrates the intended identity-binding behavior at source level: verification returns INVALID_SIGNATURE.

5.2 Replay rejection

For a packet p that was previously accepted from peer u, the receiver stores $S_u \geq s$. Replaying the same packet produces

$$s_replay = s \leq S_u \Rightarrow F_seq(p_replay)=0 \tag{9}$$

Therefore the replay is rejected even if its HMAC remains valid and its timestamp remains inside the freshness window. This independence is useful when the clock is imprecise or manipulated within $\Delta+\ell$.

5.3 Cross-epoch replay rejection

Suppose a previously valid packet belongs to epoch e_old and the receiver now expects $e_new \neq e_old$. Then

$$e_old \neq e_new \Rightarrow F_epoch(p_old)=0 \Rightarrow A(p_old)=0 \tag{10}$$

This property does not require the receiver to remember every sequence value from earlier epochs. It creates a coarse revocation boundary, provided epoch changes are coupled to the operational trust transition that motivates them.

5.4 Pre-action state-safety property

Let U be the state-cache update function and Q be a safety-relevant peer action such as acceptance of an emergency-corridor message. The required ordering is

$$A(p)=0 \Rightarrow U(p)=\emptyset \wedge Q(p)=\emptyset \tag{11}$$

The source tests explicitly encode this ordering for two cases: an authentication failure leaves the SwarmStateCache peer count unchanged, and an emergency-corridor packet is rejected while unsigned and considered by the consensus manager only after successful signing and verification. This is a protocol-placement property rather than a property of HMAC itself.

5.5 Shared-secret compromise boundary

If a fully compromised authorized member obtains K, then the outsider-forgery assumption no longer applies. The attacker can choose arbitrary identity and payload values and compute a valid tag.

$$K_attacker = K \Rightarrow \exists p^* : F_HMAC(p^*)=1 \text{ for attacker-chosen } C(p^*) \tag{12}$$

The remaining sequence, time, and epoch gates can still reject stale packets, but they do not prove that sender_id belongs to the physical node generating the packet. Consequently, group HMAC provides message integrity and group membership under key secrecy, not per-node non-repudiation or insider attribution. This limitation is central to the migration roadmap in Section 9.

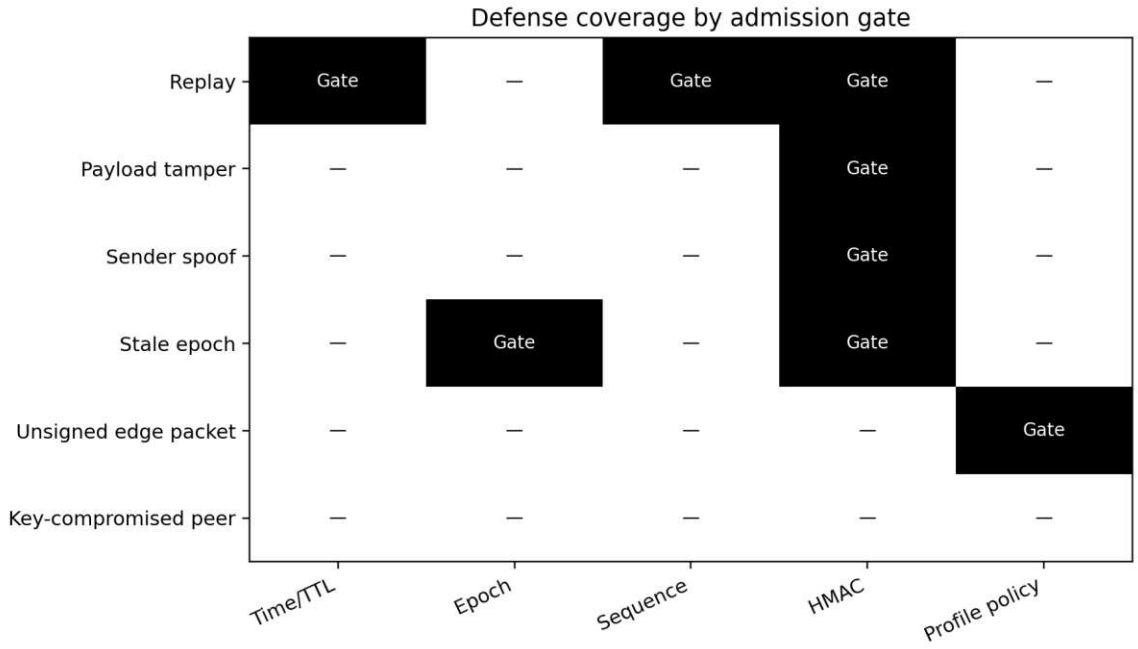


Fig. 3 Defense coverage of the implemented admission gates. A key-compromised authorized peer is intentionally shown as an uncovered residual threat.

6 Operational Analysis of the Replay State

6.1 Strict monotonicity and packet reordering

The current sequence rule is simple and deterministic, but it converts network reordering into a security rejection. Let R be the event that a legitimate packet with sequence s arrives after a later legitimate packet s+k has already been accepted. Under strict monotonicity, every such event is rejected:

$$R = \{s < S_u\} \Rightarrow P(\text{reject} \mid R)=1 \tag{13}$$

If p_R is the probability that a legitimate packet arrives behind the current maximum sequence, then the security-induced false-rejection probability contributed by reordering is approximately

$$P_FR,seq = p_R \tag{14}$$

No repository evidence presently quantifies p_R on a production radio. Therefore this paper does not assign a numerical packet-loss penalty to the rule. In a network where only the newest peer state is meaningful, rejecting out-of-order state may be acceptable or desirable. For event streams in which every message matters, a bounded replay window is more appropriate.

6.2 Sliding-window generalization

A future receiver can retain the highest authenticated sequence H_u and a W-bit replay bitmap. A packet sequence s is admissible when it is newer than H_u or lies inside the recent window and has not already been observed. A generic condition is

$$F_win(s)=1\{\, s>H_u \lor (H_u-W < s \leq H_u \land seen_u[s]=0) \,\} \tag{15}$$

This preserves anti-replay while allowing bounded reordering, following the design principle used by IPsec anti-replay windows [12,13]. It requires additional state and careful handling of wraparound, but the memory cost is only W bits plus counters per peer.

6.3 Sequence exhaustion

The edge packet uses a 32-bit sequence number. If a peer transmits authenticated packets at a constant rate r packets/s and the counter begins near zero, the approximate time to exhaust the sequence space is

$$T_wrap(r) = (2^{32} - 1) \,/\, r \tag{16}$$

Packet rate per peer	Time to 32-bit exhaustion
1	136.10 years
10	13.61 years
50	2.72 years
100	1.36 years
500	99.42 days
1,000	49.71 days
10,000	4.97 days

The result shows that wraparound is not merely theoretical at very high packet rates: 1000 packets/s reaches the 32-bit limit in roughly 49.7 days. A production design should therefore rotate the sequence context before wrap, preferably together with a key/session or trust-epoch transition, rather than permitting the counter to cycle back to zero.

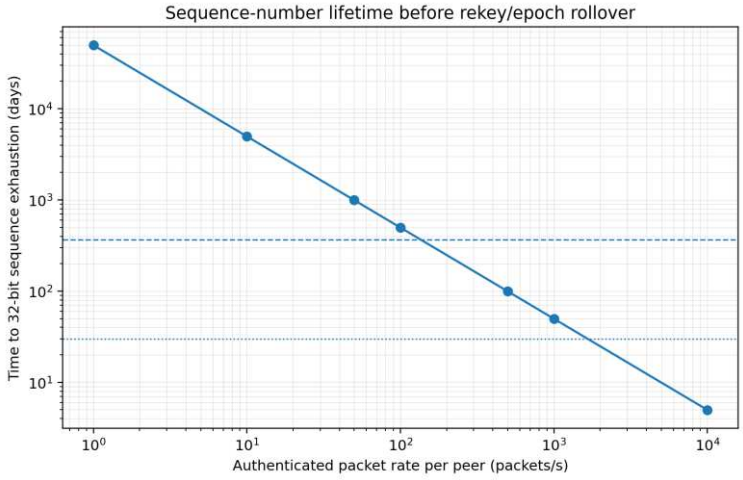


Fig. 4 Analytical lifetime of a 32-bit per-peer sequence space as a function of authenticated packet rate.

7 Software Implementation Mapping and Validation Evidence

7.1 Source mapping

The security model is directly traceable to the studied repository. Table 4 maps the mathematical terms to concrete code paths. This mapping is important because security claims can otherwise drift away from what the software actually enforces.

Model element	Implementation evidence	Observed behavior
Canonical C(p)	PeerPacketAuth.cpp clears auth_hook and serializes canonical CBOR	Authentication covers packet metadata and payload
F_time	verifyPacket computes absolute age and compares with max_clock_skew_ms + ttl_ms	Outside-window packets rejected
F_epoch	validateTrustEpoch	Mismatched trust epoch returns STALE_EPOCH
F_seq	validateNonceOrSequence	sequence ≤ last_sequence returns REPLAY_DETECTED
F_HMAC	HMAC-SHA256 over canonical bytes + constant-time equality	Tamper returns INVALID_SIGNATURE
F_profile	unsigned_allowed_for_profile	Unsigned simulation exception is profile-scoped
Unsupported modes	ED25519_PLACEHOLDER and PQC_HYBRID_PLACEHOLDER	Verification returns UNSUPPORTED
Pre-action gate	test cache and emergency-corridor cases	Failed auth does not authorize downstream use

7.2 Source-level security tests

The test_edge_swarm.cpp target defines packet-authentication cases that correspond directly to the threat model. The repository-wide validation report records the native tests as remaining green at the studied commit [18]. This paper did not rerun the suite in a physical-flight environment; the evidence is therefore software-test evidence at the referenced source revision, not flight validation.

Test case	Expected security result	Property supported
HmacSignedPacketVerifies	ACCEPTED	Correct authenticated packet path
TamperedPayloadRejected	INVALID_SIGNATURE	Payload integrity
TamperedSenderRejected	INVALID_SIGNATURE	Sender identifier bound to MAC
MissingSignatureRejectedInEdgeSwarm	MISSING_SIGNATURE	Fail-closed edge profile
UnsignedAllowedOnlyInSimulationWhenConfigured	Simulation ACCEPTED; edge mode rejected	Profile separation
StaleTrustEpochRejected	STALE_EPOCH	Cross-epoch isolation
StaleSequenceRejected	REPLAY_DETECTED	Strict anti-replay
PqcPlaceholderReturnsUnsupported	UNSUPPORTED	No false PQC implementation claim
AuthFailurePreventsCacheUpdate	Cache remains unchanged	Pre-state-mutation gate
EmergencyCorridorRequiresAuthBeforePeerAction	Unsigned rejected; signed accepted	Pre-action authorization

7.3 Performance evidence and its boundary

The repository Phase 6 benchmark report includes a “swarm auth path” proxy with the values shown in Table 6 [17]. The report explicitly warns that repeated process launch dominates native microbenchmark tails and that these timings are representative wall-clock path timings rather than isolated cryptographic measurements. Accordingly, the numbers are useful only as a coarse software-path sanity check.

Metric	Mean	p50	p95	p99 / max
Swarm authentication path	22.603 ms	12.219 ms	115.594 ms	124.142 ms
Telemetry serialization path	15.740 ms	9.985 ms	22.272 ms	121.210 ms

The median authentication-path value is materially lower than its p95, which is consistent with the benchmark report's warning about launch and scheduling noise. No claim is made that HMAC-SHA256 itself requires 12.219 ms, nor that 44.2 Hz is a hard real-time security limit. A publication-grade cryptographic performance result would require an in-process benchmark that isolates canonical serialization, HMAC generation, HMAC verification, cache lookup, and sequence-state update across representative embedded hardware.

8 Analytical Results

8.1 Gate independence against timestamp manipulation

The compound design is stronger than a single freshness signal because an attacker who manipulates apparent time but lacks K still faces HMAC verification, while an attacker replaying an authentic packet with a fresh-looking time cannot modify that time without invalidating the MAC. Even if the original timestamp remains within the accepted window, the stored sequence state rejects the replay. Formally, if $F_time(p)=1$ for a replayed packet but $s \leq S_u$, then

$$F_time(p)=1 \wedge F_seq(p)=0 \Rightarrow A(p)=0 \tag{17}$$

Similarly, if a previously valid packet advances the sequence but carries an obsolete trust epoch, the epoch gate rejects it. These independent predicates reduce reliance on any single freshness assumption, which is particularly relevant given demonstrated timestamp-manipulation attacks on UAV communication protocols [6].

8.2 Attack success conditions

For an outsider without K, a successful sender-spoof or payload-forgery attack must simultaneously satisfy all non-cryptographic gates and forge the HMAC. An approximate factorization for conceptual analysis is

$$P_success,out \approx P(F_time)\cdot P(F_epoch)\cdot P(F_seq)\cdot Adv_HMAC(q) \tag{18}$$

This is not an empirical probability estimate because the terms depend on attacker strategy and system state. It instead shows that time, epoch, and sequence constraints reduce the set of packets on which the cryptographic forgery problem is even relevant.

8.3 Insider compromise

For an insider holding K, the HMAC factor is no longer a barrier. If the attacker can choose a new sequence, current epoch, and valid time, then the packet can satisfy the admission predicate. Therefore the security problem changes from message authenticity to node attribution and behavior consistency. This is why compromised-node detection should not be claimed from the current packet authenticator alone. A realistic extension is to assign each node a distinct key or asymmetric identity and then combine cryptographic attribution with runtime trust or anomaly scoring.

8.4 Communication overhead

The current authentication hook carries a full 32-byte HMAC encoded as hexadecimal, so the textual representation contributes 64 hexadecimal characters plus the "hmac_sha256:" prefix. Let B_0 be the canonical packet size before the authentication string and B_a the added authentication bytes in the serialized wire format. The relative authentication expansion is

$$\eta_auth = B_a \,/\, B_0 \tag{19}$$

Because B_0 varies by packet type, the repository does not provide a single measured η_auth for all edge messages. A future compact binary tag representation can remove hexadecimal expansion without changing the HMAC security model. Truncation could further reduce bandwidth, but would lower the generic forgery work factor and therefore requires an explicit security-performance choice rather than an implementation convenience.

9 Security Discussion and Design Evolution

9.1 What the current design establishes

Within its stated assumptions, the implementation provides a coherent receiver-side defense against several high-value outsider attacks. Captured packets cannot be replayed once their sequence has been accepted. Payload or sender-identity changes invalidate the HMAC. Packets from a mismatched epoch are rejected. Unsigned packets are rejected in the edge-swarm profile while a controlled simulation exception remains available for development. Most importantly, verification is positioned before state-cache update and before the tested emergency-corridor action path.

9.2 What it does not establish

The group secret is the dominant trust boundary. Any authorized node that leaks or maliciously uses K can generate a valid tag for arbitrary sender identifiers. The present HMAC design therefore authenticates possession of a swarm secret, not a unique UAV identity. It also does not encrypt packet content, does not prevent traffic analysis, and does not provide forward secrecy. Sequence state is memory-resident and the current strict rule does not tolerate reordering. The source includes Ed25519 and PQC hybrid labels only as placeholders; neither is implemented in the studied build. These are not minor caveats but the exact boundary between the current evidence and future claims.

9.3 Per-node keying and cryptographic attribution

The highest-value next step is to replace or supplement the group key with per-node authentication material. With distinct symmetric keys K_u distributed through a trusted provisioning system, the receiver can verify a claimed sender u using K_u. A stronger asymmetric design would bind each UAV identifier to a public verification key, enabling source attribution without sharing signing capability among all peers. Hardware-backed storage can then reduce the risk that physical capture yields exportable keys. These changes directly address the compromised-member spoofing gap identified in Fig. 3.

9.4 Replay windows and restart semantics

A production transport should define restart behavior explicitly. If a process restarts and forgets S_u while an old key/epoch remains valid, replay state can be weakened. Possible solutions include persisting sequence checkpoints, rotating the trust epoch on restart, deriving session-specific keys, or accepting only a bounded handshake that reestablishes a new sequence base. For lossy radios, a sliding replay window should be considered so that legitimate reordering does not become avoidable packet loss.

9.5 Time-source hardening

The multi-gate design reduces dependence on wall-clock time but does not eliminate time security. Systems that derive packet time from GNSS should protect the time source through cross-checks with monotonic local clocks, oscillator bounds, or multi-source synchronization. The recent MAVLink timestamp-manipulation study [6] is particularly relevant: a cryptographically signed timestamp is still only as meaningful as the state machine that generates and validates it. Sequence and epoch gates should therefore remain independent of GNSS time.

9.6 Post-quantum and asymmetric roadmap

The source contains ED25519_PLACEHOLDER and PQC_HYBRID_PLACEHOLDER modes that intentionally return UNSUPPORTED. A future paper may evaluate asymmetric or hybrid post-quantum authentication, but such a claim requires actual implementation, key provisioning, packet-size measurements, computational benchmarks on the target edge hardware, and adversarial tests. Treating a placeholder enum as deployed cryptography would violate the evidence boundary maintained in this study.

10 Comparison with Related Replay-Control Designs

Table 7 places the current receiver gate beside representative replay-control patterns. The comparison is architectural rather than a performance ranking because the systems have different goals and threat models.

Design	Integrity primitive	Freshness state	Reordering tolerance	Identity granularity	Key compromise implication
Current edge-swarm baseline	Full HMAC-SHA256 over canonical packet	Timestamp+TTL, trust epoch, strict sequence	No (strict monotonic)	Group-secret possession	One shared secret can enable insider spoofing
MAVLink 2 signing [10]	SHA-256-derived truncated signature with shared key	Monotonic per-link timestamp	Timestamp-order based	Logical stream + shared key	Shared key compromise affects authenticated link
IPsec ESP anti-replay [12,13]	Integrity/authentication depends on negotiated SA	Sequence number + replay window	Yes, within window	Security association	SA key compromise affects that association
PUF / per-node protocols [3-5]	Protocol-specific per-device authentication	Nonces/session state	Protocol-dependent	Per-device	Designed to improve captured-node/key isolation

The current design's main advantage is implementation simplicity and deterministic state cost; its main weakness is shared-secret attribution. Conversely, a replay window improves network tolerance but adds receiver state, and per-node or PUF-based identity improves attribution at the cost of provisioning and protocol complexity. The correct choice depends on swarm size, radio behavior, node-capture threat, and available secure storage.

11 Limitations and Reproducibility

The study is bounded to software evidence at the referenced repository revision. No physical-flight campaign, adversarial RF experiment, production radio test, or hardware-in-the-loop replay campaign is claimed. The repository benchmark is a process-level proxy and not an isolated HMAC benchmark. The strict-sequence false-rejection analysis is mathematical because no production-radio reordering distribution is available in the repository.

The group-HMAC design does not detect a compromised authorized node that knows the secret and emits fresh, correctly authenticated malicious messages. Peer-spoofing resistance in this paper therefore means resistance to identity or packet modification by an adversary without the group secret. Insider attribution requires per-node credentials or an additional trust mechanism.

The repository test cases provide reproducible software checks for the packet-admission logic. The studied source revision is c040073969a6111f4c652d08e8de8629d70b2317. The associated research artifact supplied for this project is available at https://doi.org/10.5281/zenodo.22092648. Reproducibility should pin the source revision because future authentication modes or replay-window changes can alter the security semantics analyzed here.

12 Conclusion

This paper presented a sequence-, epoch-, and integrity-aware packet-admission framework for decentralized UAV mesh networks and analyzed it against replay, payload tampering, sender spoofing, stale-epoch reuse, and unsigned injection. The studied implementation combines canonical CBOR authentication with full HMAC-SHA256, bounded timestamp/TTL freshness, exact trust-epoch equality, strictly increasing per-peer sequence numbers, and runtime-profile policy. The resulting acceptance predicate is intentionally fail-closed: a packet that fails any gate is rejected before peer-state cache mutation or the tested safety-relevant emergency action path. Source-level tests directly encode the core behaviors, while repository validation records the native test suite as green at the studied revision.

The analysis also clarifies the limits of the current design. Strict monotonic sequencing trades replay simplicity for zero tolerance of legitimate reordering; 32-bit counters require defined rollover semantics at sufficiently high packet rates; and the shared swarm secret cannot attribute messages to a unique uncompromised UAV once an insider obtains the key. These limitations are actionable rather than incidental. A production evolution should combine per-node or asymmetric credentials, protected key storage, restart-safe epoch/session management, bounded replay windows, and hardened time-source validation. With those extensions, the current deterministic admission gate can serve as a compact security foundation for more advanced trust and compromised-node detection mechanisms without overstating what the present software already proves.

Declarations

Funding: No external funding was received for this study.

Conflict of interest: The author declares no competing interests.

Ethics approval: Not applicable. The study concerns software and analytical evaluation and does not involve human participants or animals.

Data and code availability: The software evidence analyzed in this paper is associated with the open UAV research platform at GitHub revision c040073969a6111f4c652d08e8de8629d70b2317. An associated research artifact is archived at https://doi.org/10.5281/zenodo.22092648.

Author contribution: M.S.I.S. conceived and designed the study, developed the research methodology and mathematical security model, analyzed the software implementation and validation evidence, interpreted the results, prepared the figures and tables, and wrote, reviewed, and approved the final manuscript.

Generative AI statement: During manuscript preparation, the author used OpenAI ChatGPT to assist with language drafting, mathematical presentation, and document formatting. The author reviewed and verified the technical claims, source mappings, calculations, references, and final text and takes full responsibility for the manuscript.

References

1. Ceviz, O., Sen, S., Sadioglu, P.: A Survey of Security in UAVs and FANETs: Issues, Threats, Analysis of Attacks, and Solutions. IEEE Communications Surveys & Tutorials 27(5), 3227–3265 (2025). https://doi.org/10.1109/COMST.2024.3515051
2. Kundu, J., Alam, S., Das, J.C., Dey, A., De, D.: Trust-Based Flying Ad Hoc Network: A Survey. IEEE Access 12, 99258–99281 (2024). https://doi.org/10.1109/ACCESS.2024.3419904
3. Sen, M.A., Al-Rubaye, S., Tsourdos, A.: Securing UAV Flying Ad Hoc Wireless Networks: Authentication Development for Robust Communications. Sensors 25(4), 1194 (2025). https://doi.org/10.3390/s25041194
4. Chen, L., Zhu, Y., Liu, S., Yu, H., Zhang, B.: PUF-based dynamic secret-key strategy with hierarchical blockchain for UAV swarm authentication. Computer Communications 218, 31–43 (2024). https://doi.org/10.1016/j.comcom.2024.02.001
5. Liu, Y., Wang, S., Cao, L., Wang, P., Zhang, Z., Qiu, S., Dong, R.: An Anonymous Authentication and Key Exchange Protocol for UAVs in Flying Ad-Hoc Networks. Computers, Materials & Continua 86(3), 52 (2026). https://doi.org/10.32604/cmc.2025.072710
6. Colton, Z., Oracevic, A., Dilek, S.: Timestamp Manipulation-Based GPS Spoofing Attacks on the MAVLink 2.0 Protocol for UAV Communication: An Empirical Study. IEEE Transactions on Communications 74, 2564–2579 (2026). https://doi.org/10.1109/TCOMM.2025.3644474
7. Moosbrugger, P., Rozier, K.Y., Schumann, J.: R2U2: monitoring and diagnosis of security threats for unmanned aerial systems. Formal Methods in System Design 51, 31–61 (2017). https://doi.org/10.1007/s10703-017-0275-x
8. Mouatassim, T., Airaj, M., El Guarmah, E.M.: Safeguarding the skies: the rise of machine learning approaches for cyberattack detection in unmanned aerial systems. Cybersecurity 9, 62 (2026). https://doi.org/10.1186/s42400-025-00466-2
9. Tariq, U., Ahanger, T.A., Ihsan, M.: Systematic review of machine and deep learning models for unmanned aerial vehicles cyber threat defense. Discover Artificial Intelligence 6, 216 (2026). https://doi.org/10.1007/s44163-026-00960-7
10. MAVLink Project: Message Signing (Authentication), MAVLink Guide. https://mavlink.io/en/guide/message_signing.html. Accessed 25 Aug 2026.
11. National Institute of Standards and Technology: The Keyed-Hash Message Authentication Code (HMAC), FIPS PUB 198-1 (2008). https://doi.org/10.6028/NIST.FIPS.198-1
12. Kent, S.: IP Encapsulating Security Payload (ESP), RFC 4303 (2005). https://doi.org/10.17487/RFC4303
13. Zhang, X., Tsou, T.: IPsec Anti-Replay Algorithm without Bit Shifting, RFC 6479 (2012). https://doi.org/10.17487/RFC6479
14. Krawczyk, H., Bellare, M., Canetti, R.: HMAC: Keyed-Hashing for Message Authentication, RFC 2104 (1997). https://doi.org/10.17487/RFC2104
15. Yan, M., Dong, R., Han, Q.-L., et al.: Securing the low-altitude economy: a survey. Science China Information Sciences 69, 141202 (2026). https://doi.org/10.1007/s11432-025-4811-2
16. Islam Shagor, M.S.: UVA GPS Denied Navigation in Dynamic Environments, source revision c040073969a6111f4c652d08e8de8629d70b2317. GitHub repository (2026).
17. Islam Shagor, M.S.: Phase 6 Benchmark Report: Performance Engineering and Stability Validation. Repository artifact, 16 July 2026.
18. Islam Shagor, M.S.: Phase 10 Validation Report: Enterprise Deployment and Security Readiness. Repository artifact, 17 July 2026.
19. National Institute of Standards and Technology: Recommendation for Applications Using Approved Hash Algorithms. NIST SP 800-107 Rev. 1 (2012). https://doi.org/10.6028/NIST.SP.800-107r1
20. Diockou, D., Wu, D., Lu, X., Jing, Y., Shabut, A., Barwood, M.: A review of security frameworks in flying ad-hoc networks. In: 2024 IEEE International Conference on e-Business Engineering (ICEBE), pp. 183–190 (2024). https://doi.org/10.1109/ICEBE62490.2024.00036