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.

Cyber-Resilience and Trust-Aware Command Authorization for Autonomous UAV Swarms in GPS-Denied Environments

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, cyber resilience, command authorization, zero trust, drone swarm, fail-closed control

Posted Date: August 26th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10812590/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.

Cyber-Resilience and Trust-Aware Command Authorization for Autonomous UAV Swarms in GPS-Denied Environments

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

Abstract

Autonomous UAV swarms increasingly depend on remote and peer-issued commands while operating under uncertain communication, localization, and infrastructure conditions. A command can be cryptographically authentic yet operationally unsafe because the issuer is no longer trusted, the command is stale, the swarm is inconsistent, localization has degraded, or the vehicle has entered a safety-critical state. This paper develops a fail-closed runtime cyber-resilience and command-authorization framework that treats command acceptance as a joint security-and-safety decision rather than a packet-authentication decision alone. The framework is derived from an implemented C++20 UAV research platform and combines authenticated swarm ingress, replay and freshness checks, a prioritized nine-state cyber-resilience supervisor, trust-epoch transitions, composite link-integrity and tamper scoring, trust-aware command gating, security-triggered autonomous fallback, and an independent local safety manager. The mathematical model formalizes state predicates, non-critical command acceptance, emergency-command semantics, and monotonic authority reduction under degraded trust. Source-faithful scenario evaluation shows how link, timing, localization, issuer trust, geofence, swarm consistency, and survivability evidence map to deterministic security states and command outcomes. Repository tests reject tampered, replayed, wrong-secret, and broken command-history frames, while an existing local benchmark reports a swarm-authentication path median of 12.219 ms and mean of 22.603 ms, with large process-noise tails. The evidence supports software-level cyber-resilience claims, not physical-flight or production-radio certification.

Keywords: UAV cybersecurity; cyber resilience; command authorization; zero trust; drone swarm; fail-closed control

1 Introduction

Unmanned aerial vehicles (UAVs) have moved from remotely piloted platforms toward increasingly autonomous cyber-physical systems that combine sensing, estimation, networking, planning, and closed-loop control. In a swarm, the control surface becomes larger: a vehicle may accept formation commands, mission synchronization messages, emergency instructions, and peer-coordination state from entities that are mobile, intermittently connected, and themselves potentially degraded. Recent surveys emphasize that autonomous UAV operation depends simultaneously on reliable sensing, real-time processing, security, and adaptive decision making [1, 2]. The resulting security question is therefore not limited to whether a packet is authentic. The runtime must also decide whether an otherwise authentic command remains permissible under the current physical and cyber state.

Zero-trust thinking is useful in this setting because it removes implicit trust based on network location or prior admission [3]. UAV-specific reviews similarly argue for continuous verification, least privilege, and contextual policy decisions [4]. Authentication protocols for drone swarms have improved substantially, including scalable group authentication [5], hierarchical entity authentication [6], and lightweight mutual authentication for decentralized swarms [7]. However, authentication alone does not answer several control-critical questions: Should a valid formation command be executed when localization confidence has fallen below a safe threshold? Should a peer command be accepted after repeated replay evidence? Should a remote return-home request supersede a locally triggered safe-return state? Can a compromised but correctly keyed node retain command authority? These questions require a runtime policy that composes communications security with vehicle safety.

Cyber-resilience research has begun to address this broader problem. d'Ambrosio et al. proposed an open UAV control architecture and a model for reasoning about cyber threats and hazardous scenarios [8]. Related work has explored operator authentication against hijacking [9], zero-trust continuous authentication for UAV ecosystems [10], and attack recovery in autonomous platforms [11]. These approaches motivate a shift from static perimeter security toward closed-loop resilience. Yet a practical gap remains between security evidence and actuator authority: many systems detect anomalies or authenticate nodes but leave unclear the exact predicate under which a command is allowed to influence the vehicle.

This work addresses that gap through a deterministic, fail-closed runtime supervisor implemented in an open GPS-denied UAV swarm research platform. The supervisor consumes link quality, time-synchronization confidence, localization confidence, issuer trust, swarm consistency, backend trust, geofence state, battery state, and security event counters. It assigns one of nine ordered security states and exposes whether remote commands and telemetry uplink are allowed. A command-policy layer then evaluates command age, issuer trust, geofence restrictions, swarm consistency, localization confidence, battery margin, and the current security state. Finally, a local safety manager enforces sensor, link, localization, and motor constraints after command processing. The resulting design forms a defense-in-depth chain from authenticated ingress to bounded control output.

The paper makes five contributions. First, it formulates the implemented runtime security monitor as a prioritized finite-state cyber-resilience supervisor with explicit fail-closed authority semantics. Second, it derives the composite integrity and tamper scores directly from the runtime coefficients and analyzes their thresholds rather than treating them as opaque heuristics. Third, it formalizes command authorization as a composition of authenticated ingress, freshness, issuer trust, mission constraints, security authority, and local safety. Fourth, it establishes direct properties of the deterministic policy, including non-emergency authority monotonicity and safe local fallback under suspicious states. Fifth, it evaluates the design using source-faithful

scenario analysis, repository unit-test evidence, a security audit artifact, and existing authentication-path latency measurements while explicitly separating software evidence from physical-flight evidence.

The paper does not propose a new cipher, blockchain, or key-exchange protocol. Its novelty is the control-facing composition of security evidence into runtime authority. This boundary deliberately distinguishes the present work from communication-protocol papers and from blockchain-assisted trust approaches. The central research question is: **how should an autonomous UAV reduce or remove remote command authority when cyber and physical trust signals degrade, without depending on a continuously available central policy server?**

2 Background and Related Work

2.1 Zero trust and continuous authorization

NIST SP 800-207 defines zero trust around resource protection, explicit verification, and the absence of implicit trust based solely on network location [3]. For UAVs, the principle is especially relevant because network membership is transient and wireless links are exposed. Alquwayzani and Albuali reviewed zero-trust architectures for military UAV security and highlighted authentication, authorization, continuous monitoring, and least privilege as recurring requirements [4]. Kholidy further studied continuous UAV authentication in a 5G-enabled setting using blockchain-assisted zero-trust concepts [10]. These works motivate contextual authorization, but the present study focuses on a different layer: an onboard runtime policy that can remove command authority even after communication-layer authentication succeeds.

2.2 Authentication and secure swarm communication

Drone-swarm authentication research seeks to reduce the cost and dependence associated with centralized public-key infrastructures. Han et al. proposed scalable fog-edge authentication and reported substantially lower computation overhead than PKI-based alternatives [5]. Constantinescu et al. presented hierarchical authentication and malicious-entity detection for drone missions [6]. Illyass et al. developed a lightweight inter-drone mutual-authentication scheme based on challenge-response, HMAC, and authenticated encryption [7]. Such protocols are necessary because an unauthenticated command must never reach the control layer. They are not sufficient, however, when a legitimate peer becomes behaviorally suspect or when the local vehicle's safety margin is degraded.

The implementation studied here includes two security paths. A packet-authentication layer supports HMAC-SHA256, trust-epoch validation, monotonic sequence checking, timestamp/TTL freshness, and constant-time MAC comparison. A swarm security context additionally provides PBKDF2-SHA256 key derivation, AES-256-CBC encryption, HMAC-SHA256 integrity, and hash-chain continuity for selected mission/leader messages. Ed25519 and post-quantum modes appearing in the packet-authentication enumeration are placeholder-only and are not treated as implemented evidence in this paper.

2.3 Cyber resilience and control safety

Cyber resilience differs from prevention because it concerns continued safe operation after trust is degraded. d'Ambrosio et al. connected UAV software architecture with cyber-threat impact and safety analysis [8]. Lee and Segev studied cyber-physical operator authentication for preventing drone hijacking [9]. Recent work also explores runtime recovery from attacks through model-based or learning-based mechanisms [11]. In parallel, safety engineering for autonomous systems emphasizes layered constraints rather than reliance on a single detector. This paper follows that systems view: packet validation, security-state supervision, command authorization, security fallback, and local safety enforcement are separate gates with distinct failure semantics.

2.4 Research gap

Existing UAV security literature often evaluates authentication cost, cryptographic strength, intrusion detection, or attack recovery independently. The under-explored problem is **authority composition**: given an authenticated message and a changing physical/cyber context, exactly when does remote control remain admissible? A useful design must answer this question deterministically, expose the reason for rejection, and ensure that moving to a less trusted state never accidentally expands non-emergency authority. The framework studied here provides those explicit predicates and therefore supports both implementation review and mathematical analysis.

3 System Context and Threat Model

3.1 Swarm and command model

Let the swarm be

$$\mathcal{U} = \{U_1, U_2, \dots, U_N\}. \tag{1}$$

For a vehicle U_i, let the time-varying set of reachable peers be

$$\mathcal{N}_i(t) \subseteq \mathcal{U}, \qquad U_i \notin \mathcal{N}_i(t). \tag{2}$$

A received remote command is modeled as

$$c = (a, j, n, t_c, \kappa, \mu), \tag{3}$$

where a is the requested action, j is the source identifier, n is the sequence number, t_c is the issue time, $\kappa \in \{0,1\}$ indicates whether the command is critical, and μ carries the command summary or payload context.

The implemented action alphabet is

$$\mathcal{A} = \{\text{HOLD}, \text{RETURN}, \text{LAND}, \text{FORM}\}. \tag{4}$$

The networking layer may also carry non-command messages, but this paper restricts control authority to the command classes that are mapped into the runtime command envelope.

3.2 Adversary model

The adversary is assumed to control the wireless channel in the standard Dolev-Yao sense [12]: it may observe, delay, replay, drop, reorder, and inject messages, but cannot break a secure primitive by pure computation within the modeled assumptions. In addition, this paper considers *valid-credential misbehavior*: a legitimate or previously trusted peer may send commands that conflict with local safety or may become suspect due to repeated authentication errors, timing anomalies, or replay evidence. The following attacks are in scope:

- replay or stale command delivery;
- message tampering or wrong-key authentication attempts;
- peer spoofing or identity/timing inconsistency;
- stale trust-epoch use;
- unauthorized formation or mission commands;
- control-plane trust loss;
- command attempts during geofence, no-fly, localization, link, battery, or swarm-consistency degradation.

Confidentiality attacks against implementation flaws, RF jamming, physical capture, side channels, and compromised cryptographic libraries are not experimentally evaluated. Denial of service is represented only indirectly through link-quality and control-plane trust degradation; no RF-layer anti-jamming claim is made.

3.3 Evidence classes

To avoid conflating implementation with inference, evidence is divided into four classes. **Implemented evidence** refers to source-inspected C++ policy and security mechanisms. **Test evidence** refers to repository unit/integration scenarios that assert acceptance or rejection behavior. **Measured evidence** refers to stored benchmark artifacts from the repository. **Analytical evidence** refers to calculations performed from the published coefficients and thresholds. Physical flight, production-radio qualification, formal model checking, and penetration testing are outside the evidence set.

Table 1 summarizes the core runtime inputs used by the cyber-resilience supervisor.

Table 1 Runtime evidence inputs and semantics

Symbol	Runtime input	Range / type	Role in policy
q	link quality	$[0,1]$	link integrity and critical-state trigger
s	synchronization confidence	$[0,1]$	integrity score and control-plane/consistency checks
ℓ	localization confidence	$[0,1]$	integrity score and command safety
u	issuer trust score	$[0,1]$	integrity score and command authorization
Δ_p	peer clock offset	ms	peer-spoof suspicion
b	battery	%	survivability and command safety
g	geofence clear	Boolean	safe-return / command gating
m	no-fly lock	Boolean	safe-return / command gating
w	swarm consistency	Boolean	safe-return / formation gating
z	backend trust	Boolean	control-plane trust
r, a, p, c	event indicators	Boolean/counters	replay, auth, spoof, control anomalies

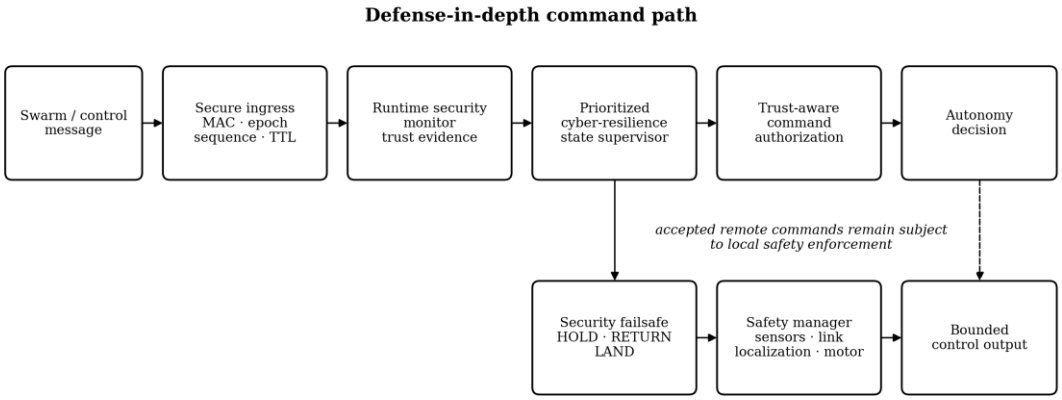


Fig. 1 Defense-in-depth runtime path from secure ingress to bounded control output.

4 Runtime Cyber-Resilience Model

4.1 Composite link-integrity score

The runtime exposes a composite diagnostic score that combines communication, timing, localization, issuer, and peer-availability evidence. Let

$$p_i = \begin{cases} 1, & |\mathcal{N}_i| > 0, \\ 0.65, & |\mathcal{N}_i| = 0. \end{cases} \tag{5}$$

Define the weighted evidence sum

$$J_i = 0.35q_i + 0.25s_i + 0.15\ell_i + 0.15u_i + 0.10p_i. \tag{6}$$

The implemented link-integrity score is

$$I_i = \mathrm{clip}(J_i, 0, 1). \tag{7}$$

The coefficients sum to one when a peer is present. Consequently, the link and time-synchronization channels jointly contribute 0.60 of the score, while localization and issuer trust contribute 0.30 and peer availability contributes 0.10. This weighting makes the score interpretable as a communications-centric health summary. Importantly, I_i is diagnostic; it is **not** the sole state-transition variable. Hard predicates remain authoritative for fail-closed state selection.

For a nominal example with $(q, s, \ell, u, p) = (0.90, 0.90, 0.90, 0.85, 1)$,

$$I_{\mathrm{nom}} = 0.9025. \tag{8}$$

With a weak link $q = 0.15$ while $(s, \ell, u, p) = (0.80, 0.85, 0.85, 1)$,

$$I_{\mathrm{weak\ link}} = 0.6075. \tag{9}$$

Figure 2 later shows the discrete state precedence, while Figure 3 visualizes how I_i varies over link quality and synchronization confidence for fixed localization and issuer trust.

4.2 Tamper accumulation

The runtime tracks a base tamper contribution T_i^0 and four event classes: replay (R_i), peer spoofing (P_i), control-plane failure (C_i), and authentication failure (A_i). Using indicator notation,

$$\mathbf{1}_R = \mathbf{1}[R_i > 0], \quad \mathbf{1}_P = \mathbf{1}[P_i > 0], \tag{10}$$

$$\mathbf{1}_C = \mathbf{1}[C_i > 0], \quad \mathbf{1}_A = \mathbf{1}[A_i > 0]. \tag{11}$$

Define the accumulated tamper evidence

$$Z_i = T_i^0 + 0.20\mathbf{1}_R + 0.25\mathbf{1}_P + 0.12\mathbf{1}_C + 0.10\mathbf{1}_A. \tag{12}$$

The implemented tamper score is

$$T_i = \mathrm{clip}(Z_i, 0, 1). \tag{13}$$

A tamper-suspect health flag is raised when

$$T_i \geq 0.60. \tag{14}$$

With zero base tamper, the maximum contribution of any three event categories is 0.57 (spoof + replay + control), whereas all four contribute 0.67. Therefore, with $T_i^0 = 0$, all four event classes are required to cross the 0.60 flag threshold. If $T_i^0 \geq 0.05$, the spoof + replay + control combination is sufficient because $0.05 + 0.25 + 0.20 + 0.12 = 0.62$.

4.3 Event memory and decay

Anomaly counters are not permanent. Let $\tau_A = 90$ s and $\tau_R = \tau_P = \tau_C = 120$ s denote the implemented inactivity windows. A generic event-memory indicator can be written as

$$M_x(t) = \mathbf{1}[t - t_x \leq \tau_x], \tag{15}$$

where t_x is the most recent event time for category x. Once the window expires, the corresponding counter resets. This creates finite-memory hysteresis: isolated transient events can age out, while repeated events within a window accumulate enough evidence to change state.

For replay events, state escalation requires either two accumulated replay failures or an explicit stale command channel in a hardened profile:

$$P_{\text{replay}} = \mathbf{1}[R_i \geq 2 \ \vee \ (h_i \wedge \neg f_i)], \tag{16}$$

where h_i denotes hardened mode and f_i denotes command-channel freshness.

4.4 Prioritized security-state selection

Let the ordered state set be

$$\mathcal{S} = \{S_1, S_2, \ldots, S_9\}, \tag{17}$$

with precedence shown in Figure 2. Each state has a Boolean predicate $P_k(x_i)$. The selected state is the first satisfied predicate:

Let the selected predicate index be

$$k_c = \min\{k: P_k\big(x_i(t)\big) = 1\}. \tag{18}$$

Then

$$S_i(t) = S_{k_c}. \tag{19}$$

The highest-priority critical predicate is

$$P_1 = \mathbf{1}[e_i \vee b_i \leq 7 \vee q_i < 0.05], \tag{20}$$

which maps to LAND_IMMEDIATELY, where e_i is the emergency-fault indicator.

The hardened secure-command-loss predicate is

$$P_2 = \mathbf{1}[h_i \wedge (\neg\sigma_i \vee \pi_i)], \tag{21}$$

where σ_i means swarm security is enabled and π_i means a placeholder secret is in use. It maps to ISOLATED_AUTONOMY.

Define the localization-link loss term

$$D_i = \lambda_i \wedge (s_i < 0.35 \vee q_i < 0.18). \tag{22}$$

The safe-return predicate is

$$P_3 = \mathbf{1}[m_i \vee \neg g_i \vee \neg w_i \vee D_i]. \tag{23}$$

Here λ_i denotes localization lost. The remaining predicates are replay suspicion, peer spoof suspicion, control-plane untrustworthiness, authorization suspicion, and degraded link, followed by trusted fallback.

The degraded-link predicate is

$$P_8 = \mathbf{1}[q_i < 0.18 \vee s_i < 0.45 \vee \ell_i < 0.35]. \tag{24}$$

If no higher-priority predicate is satisfied,

$$P_9 = 1, \tag{25}$$

and the system is TRUSTED.

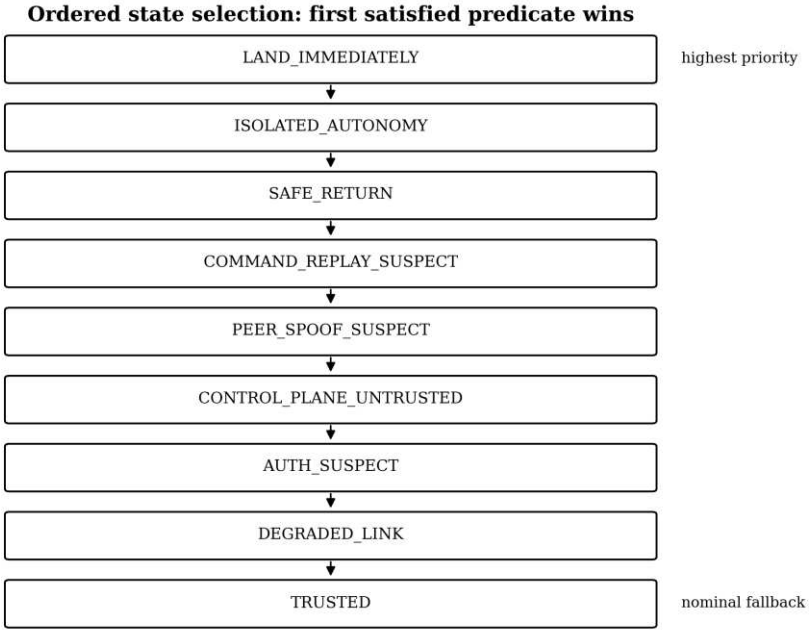


Fig. 2 Ordered runtime security-state precedence. Higher predicates dominate lower states.

4.5 Remote-command and telemetry authority

Define remote-command authority $R_i \in \{0,1\}$ and telemetry-uplink authority $Y_i \in \{0,1\}$. The implemented state mapping can be summarized as

$$R_i = \begin{cases} 1, & S_i = \text{TRUSTED}, \\ \neg h_i, & S_i = \text{DEGRADED_LINK}, \\ 0, & \text{otherwise}, \end{cases} \tag{26}$$

and

$$Y_i = \begin{cases} 0, & S_i = \text{LAND_IMMEDIATELY}, \\ 1, & \text{otherwise}. \end{cases} \tag{27}$$

Thus hardened mode removes remote authority even in the lowest-severity degraded-link state. Higher suspicious or protective states always remove remote command authority.

4.6 Trust epochs

A state transition increments a monotonic trust epoch. If $E_i(k)$ is the epoch at evaluation step k,

$$E_i(k+1) = E_i(k) + \mathbf{1}[S_i(k+1) \neq S_i(k)]. \tag{28}$$

The runtime propagates this epoch in local edge state, and the packet-authentication subsystem supports rejection of mismatched trust epochs. The present evidence does not establish that every live security-state transition automatically rebinds the packet-authentication configuration without restart; therefore, this paper treats dynamic epoch propagation as implemented state telemetry and epoch validation as an implemented packet primitive, while full online rebinding remains an integration item.

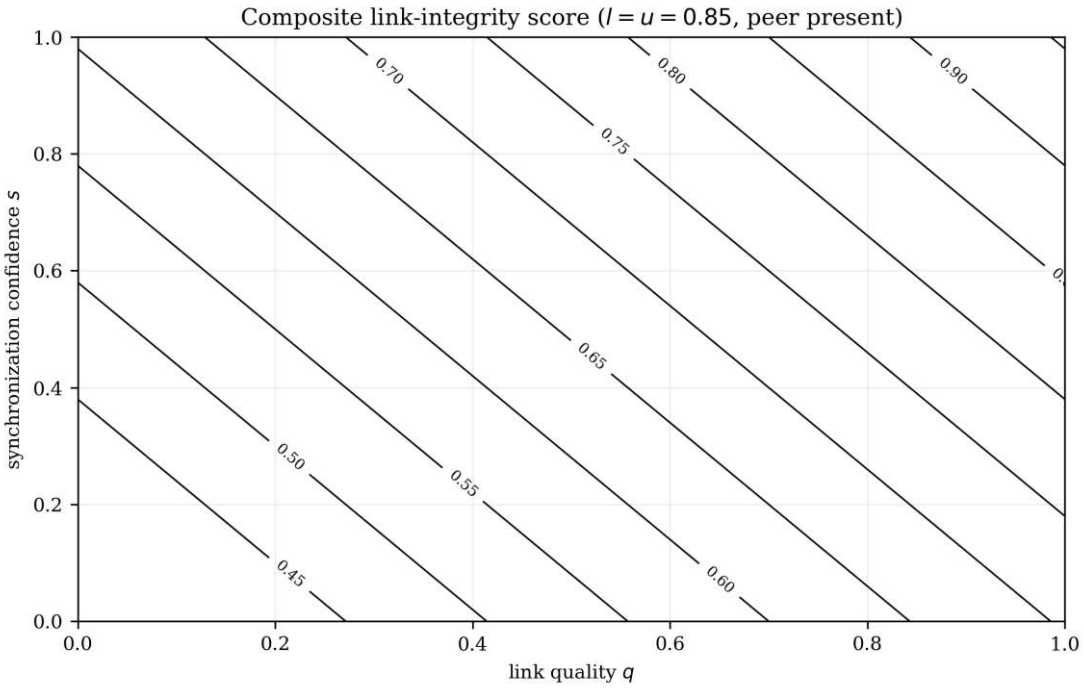


Fig. 3 Composite link-integrity score over link quality and synchronization confidence.

5 Authenticated Ingress and Command Authorization

5.1 Secure ingress predicate

When swarm security is enabled, received datagrams are passed through the secure-frame open operation before they can become SwarmMessage objects. The callback that creates a remote command envelope is downstream of this parsing step. At the packet-authentication layer, the implemented HMAC path validates message integrity, trust epoch, sequence freshness, and time freshness.

Let V_{mac}, V_E, V_n, and V_t denote these four predicates. The packet-level acceptance condition is

$$V(m) = V_{\mathrm{mac}}(m) \wedge V_E(m) \wedge V_n(m) \wedge V_t(m). \tag{29}$$

For sequence freshness,

$$V_n(m) = \mathbf{1}[n_m > n_{\mathrm{last}}]. \tag{30}$$

For time freshness, using maximum clock skew $\Delta_{\max}$ and message TTL,

$$V_t(m) = \mathbf{1}[|t_{\mathrm{now}} - t_m| \le \Delta_{\max} + TTL_m]. \tag{31}$$

The default packet-auth clock-skew allowance in the inspected configuration is 2000 ms. HMAC-SHA256 is currently implemented; Ed25519 and hybrid post-quantum modes are explicitly marked unsupported in this build and are therefore excluded from current claims.

5.2 Command-age and issuer-trust gates

The command-policy layer applies a separate command-age limit. Let

$$\Delta_c = t_{\mathrm{now}} - t_c. \tag{32}$$

For non-emergency commands, freshness requires

$$F_c = \mathbf{1}[\Delta_c \le 3.0\ \mathrm{s}]. \tag{33}$$

Issuer trust is compared against a runtime-dependent minimum:

$$\theta_u = \begin{cases} 0.70, & h_i = 1, \\ 0.55, & h_i = 0. \end{cases} \tag{34}$$

and

$$U_c = \mathbf{1}[u_i \ge \theta_u]. \tag{35}$$

In the inspected runtime, the locally assigned issuer-trust value is 0.85 when secure swarm communication is enabled with a non-placeholder secret, 0.45 with a placeholder secret, 0.30 for an unsecured hardened profile, and 0.65 for an unsecured non-hardened profile. These values are policy constants, not learned trust estimates.

5.3 Mission and physical-safety gates

For a formation-hold command, the runtime additionally checks localization, battery, generic health, and swarm consistency. Define

$$L_c = \mathbf{1}[\ell_i \geq 0.35], \qquad B_c = \mathbf{1}[b_i \geq 18], \tag{36}$$

$$H_c = \mathbf{1}[\neg e_i \wedge T_{\mathrm{CPU}} < 82^{\circ}\mathrm{C}], \tag{37}$$

and

$$W_c = \mathbf{1}[w_i = 1]. \tag{38}$$

Then the formation-specific safety condition is

$$Q_{\mathrm{form}} = L_c \wedge B_c \wedge H_c \wedge W_c. \tag{39}$$

Geofence and no-fly restrictions allow only locally safer action classes to pass those particular checks:

$$G_c = \mathbf{1}[g_i \vee a \in \{\mathrm{RETURN}, \mathrm{HOLD}\}], \tag{40}$$

$$N_c = \mathbf{1}[\neg m_i \vee a \in \{\mathrm{RETURN}, \mathrm{HOLD}\}]. \tag{41}$$

However, these exceptions do not override a security state that has already set $R_i = 0$.

5.4 Non-emergency authorization contract

For a non-emergency command, the effective authorization condition is represented as

$$A_{\mathrm{noncrit}} = V(m) \wedge F_c \wedge U_c \wedge G_c \wedge N_c \wedge R_i \wedge Q_a, \tag{42}$$

where $Q_a = Q_{\mathrm{form}}$ for formation commands and $Q_a = 1$ for actions not subject to the formation-specific physical checks.

The policy then applies a final state restriction: only TRUSTED and DEGRADED_LINK are command-accepting states at this layer. Because $R_i = 0$ in hardened DEGRADED_LINK, a hardened vehicle effectively accepts non-emergency remote commands only while trusted.

5.5 Emergency-command semantics

The source policy treats EMERGENCY_LAND specially: the command-policy function accepts it before freshness, issuer-trust, geofence, and security-state checks. This is an intentional emergency override **within the policy layer**. A secure design therefore depends on the outer authenticated-ingress boundary:

$$A_{\mathrm{emg}} = V(m) \wedge \mathbf{1}[a = \mathrm{LAND}]. \tag{43}$$

If secure ingress were disabled or bypassed, the early policy return would be a materially weaker trust boundary. The implementation normally constructs remote commands from messages received through the swarm networking layer; nevertheless, the emergency override deserves explicit treatment in security review. A future hardened profile could require stronger provenance, multi-party confirmation, or a dedicated emergency key class without removing the low-latency safety intent.

Table 2 Command authorization checks

Check	Implemented condition	Rejection effect
Command freshness	age ≤ 3.0 s	stale command rejected
Issuer trust	$u \geq 0.70$ hardened; $u \geq 0.55$ otherwise	low-trust issuer rejected
No-fly / geofence	unsafe action blocked	maneuver rejected
Swarm consistency	required for formation hold	formation rejected
Localization	$\ell \geq 0.35$ for formation	formation rejected
Battery	$b \geq 18\%$ for formation	formation rejected
Health	no emergency fault and CPU < 82 °C	formation rejected
Security authority	remote_command_allowed true	remote command rejected
Security state	trusted/degraded only at policy layer	suspicious states rejected
Emergency land	early policy override	accepted subject to outer ingress

6 Security Fallback and Safety Composition

6.1 Security-triggered local behavior

The runtime applies a security failsafe to the locally generated autonomy decision before processing a pending remote command. Let v be the current velocity and R_{wb} the body-to-world rotation. The LAND_IMMEDIATELY security state maps to

$$v_{\text{land}} = -0.8R_{wb}[0 \quad 1 \quad 0]^T. \tag{44}$$

The SAFE_RETURN state uses a damped return posture,

$$v_{\text{return}} = -0.45v + 0.18R_{wb}[0 \quad 1 \quad 0]^T. \tag{45}$$

Suspicious states (ISOLATED_AUTONOMY, CONTROL_PLANE_UNTRUSTED, PEER_SPOOF_SUSPECT, COMMAND_REPLAY_SUSPECT, and AUTH_SUSPECT) map to hold-position damping,

$$v_{\text{hold}} = -0.35v. \tag{46}$$

TRUSTED and DEGRADED_LINK do not directly override the autonomy command at this stage.

6.2 Independent safety supervisor

After any accepted remote command is applied, the safety manager evaluates sensors, localization, link status, motor state, and the current security assessment. Its decision supplies maximum speed and acceleration constraints and may replace a command with hold, localization-recovery, or emergency-land behavior. This ordering gives a useful invariant: remote authorization does not bypass the final local safety envelope.

If the safety manager detects emergency stop or security LAND_IMMEDIATELY, it forces emergency landing. A link loss or sensor fault can force hold; localization loss can force a recovery/descent posture. Let C_a denote an authorized command and Π_{safety} the safety projection. The actual control intent is

$$C_{\text{out}} = \Pi_{\text{safety}}(C_a, x_i). \tag{47}$$

The safety projection is non-expansive with respect to configured speed bounds:

$$\| v_{\text{out}} \| \leq v_{\max}(x_i). \tag{48}$$

Thus command authenticity and authorization determine whether a remote command may enter the local control path, while the safety manager determines whether the resulting action remains physically admissible.

6.3 Defense-in-depth composition

The full non-emergency control condition can be viewed as

$$C_{\text{out}} = \Pi_{\text{safety}}\left(\Pi_{\text{sec}}\left(\Pi_{\text{auth}}(c)\right)\right), \tag{49}$$

where Π_{auth} rejects invalid ingress, Π_{sec} applies state-aware command authorization and cyber fallback, and Π_{safety} enforces the final local physical envelope. These projections need not be linear; their value is the explicit ordering of authority.

7 Analytical Properties

7.1 Fail-closed non-emergency authority

Proposition 1. For every non-emergency command c, if the security supervisor sets $R_i = 0$, the command cannot be accepted by the command-policy layer.

From the implemented decision order, the policy checks security.remote_command_allowed before the accepting state switch. Therefore

$$R_i = 0 \;\Rightarrow\; A_{\text{noncrit}} = 0. \tag{50}$$

This is a direct property of the deterministic predicates, not a machine-checked formal proof.

7.2 Authority monotonicity under state escalation

Define an authority ordering where TRUSTED has the largest non-emergency remote authority and every state with $R_i = 0$ has no such authority. In hardened mode, DEGRADED_LINK also has $R_i = 0$. For any state transition from trusted to a suspicious or protective state,

$$\mathcal{A}_{\text{remote}}(S_{k+1}) \subseteq \mathcal{A}_{\text{remote}}(S_k). \tag{51}$$

Hence trust degradation cannot expand non-emergency command authority. It can only preserve or reduce it.

7.3 Local fallback dominance

When the security supervisor enters SAFE_RETURN or LAND_IMMEDIATELY, the local security failsafe establishes a protective autonomy command before pending remote command handling. Because $R_i = 0$ in these states, non-emergency remote commands cannot replace the fallback. Therefore

Let $\mathcal{S}_F$ denote the two protective states SAFE_RETURN and LAND_IMMEDIATELY. Then

$$S_i \in \mathcal{S}_F \Rightarrow A_{\text{noncrit}} = 0. \tag{52}$$

The exception is the policy-layer emergency-land override, which is directionally compatible with the most conservative local safety response but still depends on authenticated ingress.

7.4 Two-layer freshness

Packet and command freshness are intentionally distinct. Packet freshness bounds transport staleness with clock-skew and TTL terms; command freshness separately bounds control relevance at 3 s. The combined condition is

$$F_{\text{total}} = V_t(m) \wedge F_c. \tag{53}$$

This reduces the chance that a packet still valid under a broad transport TTL is nevertheless operationally stale for command execution.

8 Evaluation Methodology

The evaluation uses three sources of evidence. First, the C++ source is translated into mathematical predicates without changing thresholds or ordering. Second, a scenario matrix evaluates those predicates at selected boundary and failure points. Third, repository tests and stored validation artifacts are summarized as software evidence. No synthetic flight-accuracy numbers, attack-detection rates, or unmeasured command-latency values are invented.

8.1 Source-faithful scenario matrix

A scenario evaluator sets the runtime inputs and applies the exact state precedence. The baseline hardened configuration uses secure swarm communication, a non-placeholder secret, $q = s = \ell = 0.90$, issuer trust $u = 0.85$, battery 80%, geofence clear, no no-fly lock, swarm consistency true, backend trust true, and no anomaly counters. One variable or event family is then changed at a time.

The composite score is reported to show diagnostic behavior, but the state is computed from the hard predicates. This distinction is essential because, for example, a low issuer-trust scenario can retain a numerically moderate I_i while still entering AUTH_SUSPECT due to the explicit threshold.

8.2 Command-policy scenarios

Commands are evaluated with runtime thresholds actually used in the main loop: 3 s age, 0.35 localization confidence, 18% battery, CPU temperature below 82 C for the health check, and issuer-trust threshold 0.70 in hardened mode. Scenarios include stale formation commands, low-trust issuers, low localization, replay-suspect state, geofence violation, and emergency landing.

8.3 Repository tests

The repository's swarm-security unit tests cover: secure round-trip acceptance; rejection of a modified frame; replay rejection; rejection under a wrong shared secret; and rejection of broken leader-command hash-chain history. These tests validate security framing behavior rather than the full runtime state machine. The test build also contains broader security/swarm targets and integrates PeerPacketAuth.cpp, swarm consensus, edge protocol, state cache, and SwarmSecurity.cpp.

8.4 Measured benchmark evidence

A stored Windows automated harness reports representative swarm-authentication path timing of 22.603 ms mean, 12.219 ms p50, 115.594 ms p95, and 124.142 ms p99/max. The repository itself cautions that process launch and local scheduling dominate native microbenchmark tails; the p50 is more representative of steady-state work than the p95/p99 in that harness. These numbers are not a traced radio-to-actuator command latency and do not prove a hard real-time deadline.

9 Results

9.1 State-selection results

Table 3 shows source-faithful state outcomes. The diagnostic score remains above 0.60 in several degraded cases, yet state transitions occur because explicit predicates dominate the score. This is desirable for fail-closed control because a weighted average cannot mask a single critical failure.

Table 3 Analytical runtime-state scenarios

Scenario	Key changed input	I_i	Selected state	Remote cmd	Uplink
Nominal	baseline	0.9025	TRUSTED	yes	yes
Weak link	$q = 0.15$	0.6075	DEGRADED_LINK	no (hardened)	yes
Weak link, non-hardened	$q = 0.15, h = 0$	0.6075	DEGRADED_LINK	yes	yes
Placeholder secret	placeholder = true	0.9025	ISOLATED AUTONOMY	no	yes
No-fly lock	lock = true	0.9025	SAFE_RETURN	no	yes
Repeated replay	replay failures = 2	0.9025	REPLAY SUSPECT	no	yes
Timing/peer anomaly	peer offset = 150 ms	0.9025	PEER SPOOF SUSPECT	no	yes
Backend trust loss	backend trust = false	0.9025	CONTROL PLANE UNTRUSTED	no	yes
Low issuer trust	$u = 0.40$	0.8350*	AUTH_SUSPECT	no	yes
Critical battery	$b = 6\%$	0.9025	LAND IMMEDIATELY	no	no

*Using otherwise nominal $q = s = \ell = 0.90$; the state is triggered by the hard issuer threshold, not by I_i.

The low-issuer example illustrates the central design choice: the score is observability, whereas explicit state predicates are authorization. A system that relied only on the composite score could retain authority even when one high-value trust signal had crossed a hard limit.

9.2 Integrity-score sensitivity

With localization and issuer trust fixed at 0.85 and a peer present, the score becomes

$$I(q, s) = 0.35q + 0.25s + 0.355. \tag{54}$$

The local sensitivity coefficients are

$$\frac{\partial I}{\partial q} = 0.35, \qquad \frac{\partial I}{\partial s} = 0.25. \tag{55}$$

Thus a 0.1 decrease in link quality reduces I by 0.035, while the same decrease in synchronization reduces it by 0.025. Figure 3 makes this weighting visible. The hard degraded-link threshold $q < 0.18$ can trigger before the composite score becomes numerically small.

9.3 Tamper-score boundary

Figure 4 shows the maximum zero-base tamper score possible with zero through four active anomaly categories. The threshold is not crossed by any three categories when $T_i^0 = 0$:

$$\max_{\|x\|_0=3} T_i = 0.57 < 0.60. \tag{56}$$

All four produce

$$T_i = 0.67 > 0.60. \tag{57}$$

This means the health flag is conservative when the base tamper score is zero. In the runtime, however, base tamper can already include configuration or mission evidence, such as a placeholder secret, geofence violation, no-fly lock, or backend trust failure. The threshold can therefore be reached earlier under compound conditions.

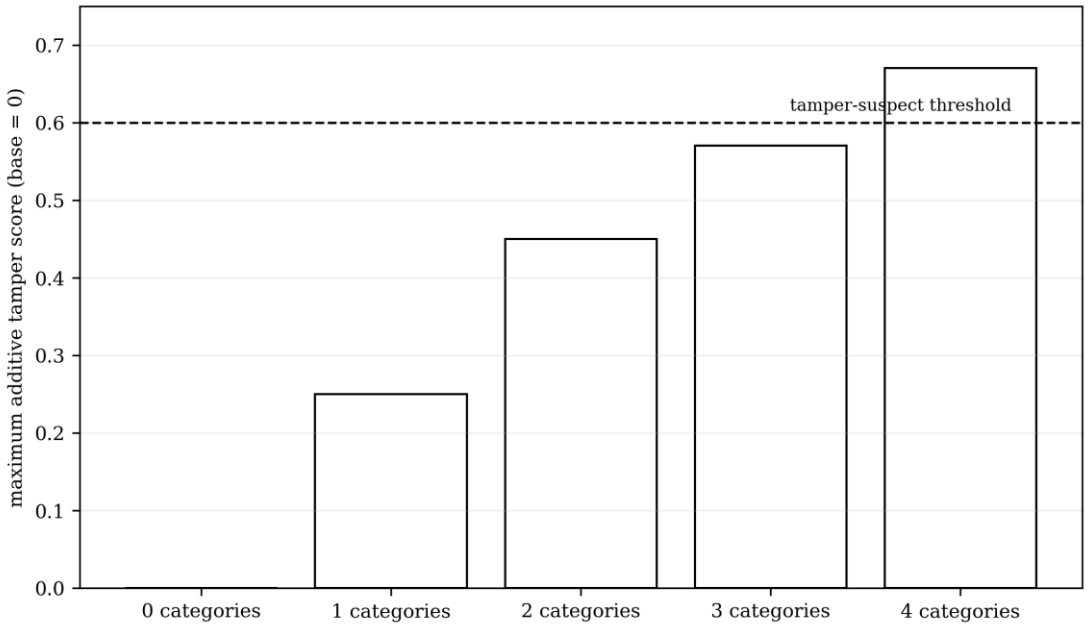


Fig. 4 Maximum additive tamper contribution by number of active anomaly categories, assuming zero base tamper.

9.4 Command-policy outcomes

Table 4 Source-faithful command authorization scenarios

Scenario	Command	Relevant evidence	Outcome	Primary reason
Nominal	FORMATION_HOLD	age 0.5 s, $u = 0.85$, $\ell = 0.9$, $b = 80$	accept	all checks pass
Stale command	FORMATION_HOLD	age 3.5 s	reject	freshness window exceeded
Low-trust issuer	FORMATION_HOLD	$u = 0.60$ hardened	reject	issuer below 0.70
Low localization	FORMATION_HOLD	$\ell = 0.30$	reject	safety margin below threshold
Replay-suspect state	HOLD	$R_i = 0$	reject	onboard security state
Safe-return state	RETURN_HOME	$R_i = 0$	reject	remote command blocked; local failsafe returns
Geofence violation, trusted hypothetical	FORMATION_HOLD	$g = 0$	reject	geofence blocks maneuver
Emergency	EMERGENCY_LAND	policy layer	accept*	critical override

*The early emergency override is secure only when the outer authenticated-ingress precondition is preserved.

The SAFE_RETURN case reveals an important composition effect. Although the command-policy's geofence branch conceptually allows RETURN_HOME and HOLD as safer action classes, the security supervisor has already set remote authority to false in SAFE_RETURN. Consequently, the remote return command is rejected; return behavior is produced locally by the security failsafe. This is stricter than a simple safe-action exception and reduces dependence on an external issuer during a compromised state.

9.5 Security framing tests

Table 5 Repository security test evidence

Test	Expected behavior	Evidence interpretation
SecureRoundTripWorks	secure frame opens successfully	baseline secure framing
TamperedFrameIsRejected	modified ciphertext/frame rejected	integrity enforcement
ReplayIsRejected	second open of same frame rejected	replay protection
WrongSecretIsRejected	mismatched secret rejected	key mismatch rejection
Broken leader-command history	broken command-history chain rejected	continuity for selected mission messages

These tests support deterministic acceptance/rejection behavior under software conditions. They do not constitute a cryptographic proof, RF adversarial test, or full penetration assessment.

9.6 Authentication-path latency

Figure 5 reports the existing benchmark artifact. The median is 12.219 ms and the mean is 22.603 ms. Tail values are much larger: 115.594 ms p95 and 124.142 ms p99/max.

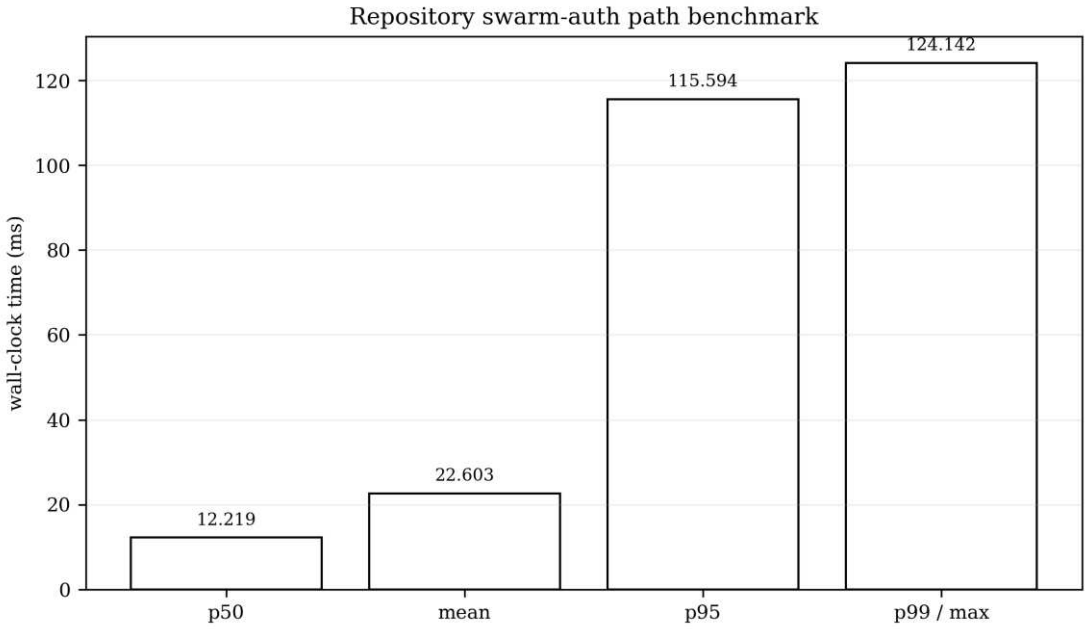


Fig. 5 Stored repository benchmark for the representative swarm-authentication path.

The tail-to-median ratio is

$$\rho_{95/50} = \frac{115.594}{12.219} \approx 9.46, \tag{58}$$

and the mean-to-median ratio is

$$\rho_{\mu/50} = \frac{22.603}{12.219} \approx 1.85. \tag{59}$$

Such dispersion is consistent with the repository note that process launch and local Windows scheduling noise affect the native harness. The result should therefore be interpreted as evidence that security processing exists in a measurable software path, not as a certified deadline for airborne control.

9.7 Security audit evidence

The stored enterprise security audit reports zero findings from its repository secret scan and confirms the presence of security headers, a TLS hardening gate, signed-command enforcement, least-privilege Compose settings, and configured CI security scanners. The audit status is PASS. This supports build/deployment security hygiene; "zero findings" must not be read as proof of zero vulnerabilities.

10 Security Analysis and Discussion

10.1 Why a continuous score is not enough

A common design pattern is to compress heterogeneous evidence into a single trust score. The implemented system computes such a score for observability, but authority is decided by ordered predicates. This is significant because weighted sums permit compensation: strong localization and issuer scores could mathematically offset a weak link. A safety-critical supervisor should not allow good evidence in one domain to cancel a categorical failure in another when the failure has a known unsafe consequence. The use of hard thresholds for critical battery, placeholder credentials in hardened mode, no-fly/geofence conflict, repeated replay, peer clock mismatch, and backend trust failure prevents this compensation.

10.2 Trust decay versus event persistence

The finite-memory counters balance two competing risks. If every anomaly were permanent, a transient fault could disable remote authority indefinitely. If counters decayed too quickly, a low-rate adversary could evade escalation by spacing attacks. The implemented 90 s authentication window and 120 s replay/spoof/control windows are engineering choices rather than mathematically optimized values. They can be interpreted as rectangular memory kernels:

$$K_x(\Delta t) = \mathbf{1}[0 \le \Delta t \le \tau_x]. \tag{60}$$

A future adaptive implementation could replace the rectangular kernel with exponential evidence decay,

$$K_x(\Delta t) = e^{-\Delta t/\tau_x}, \tag{61}$$

but such a change would alter the current deterministic semantics and is not claimed as implemented.

10.3 Emergency override as an explicit trust boundary

Emergency landing deserves special treatment. The policy accepts the emergency action before normal issuer and state checks to preserve a fast safety path. This is reasonable only when the message has already passed a trusted ingress boundary. The system architecture therefore has a security dependency that should be documented as an invariant:

$$A_{\text{emg}}^{\text{remote}} = 1 \Rightarrow V(m) = 1. \tag{62}$$

Defense-in-depth improvements could include a distinct emergency credential, two-party authorization in non-time-critical cases, origin restrictions, or a short one-time challenge. These are future design options, not current implementation claims.

10.4 Relationship to zero trust

The framework follows several zero-trust ideas without claiming to be a full NIST enterprise ZTA. It continuously re-evaluates context, reduces authority in degraded states, applies least privilege to command classes, and defaults to local autonomy when remote trust becomes inadequate. Unlike enterprise resource access, however, the protected object is a cyber-physical control action with deadlines and safety consequences. The local vehicle must therefore remain capable of autonomous fallback when the policy decision is deny.

10.5 Distinction from blockchain-assisted swarm trust

A blockchain can provide distributed auditability or membership history, but it is not necessary for the runtime authorization mechanism analyzed here. The present state machine reacts to local evidence and can deny commands without waiting for consensus. This makes the paper complementary to ledger-based approaches: distributed trust may inform issuer identity or epoch state, while the onboard supervisor remains the final authority over whether a command can affect the vehicle at a particular moment.

10.6 Practical implications

Three practical conclusions follow. First, authentication should be treated as a *precondition*, not a permission. Second, an onboard state supervisor should expose explicit remote-authority outputs rather than forcing application code to infer them from a score. Third, remote command handling should occur before a final local safety projection, so a valid but unsafe command cannot bypass vehicle constraints. These principles are applicable beyond UAV swarms to ground robots and other autonomous cyber-physical platforms with intermittent infrastructure.

11 Limitations and Threats to Validity

The main limitation is evidence scope. The security supervisor and command policy are implemented and source-inspected, and selected security framing behaviors have unit-test coverage. Nevertheless, the present evaluation does not include physical UAV flights, production radios, live adversarial RF injection, hardware fault injection, or a formal verification tool. The analytical scenario matrix replays deterministic source predicates; it is not an independent experimental detector evaluation.

The benchmark is also limited. It measures a representative swarm-authentication harness path on a local Windows environment and exhibits large scheduling/process tails. It does not trace the complete path from RF reception through authentication, policy, flight-controller command, and actuator response. No hard real-time claim is therefore justified.

The current authentication configuration uses shared-secret HMAC in the inspected packet-auth path, while stronger per-node asymmetric identity modes are placeholders there. The swarm-security context contains richer key material and secure framing, but shared-secret compromise remains an important risk. Likewise, the local hash-chain continuity mechanism for selected leader/mission commands is not a distributed blockchain consensus protocol.

The composite coefficients, state thresholds, and event decay windows were engineering choices in the platform. This paper explains their mathematical effect but does not claim that they are globally optimal. Calibration against mission-specific false-positive/false-negative costs would require a larger attack corpus and hardware or high-fidelity simulation campaigns.

Finally, the emergency-land policy override creates a deliberate asymmetry: emergency commands are favored for safety. Its secure use depends on preservation of authenticated ingress. This trust boundary should be included in future formal verification and penetration testing.

12 Reproducibility and Availability

The implementation basis is the open UAV research repository smshagor-dev/UVA-GPS-Denied-Navigation-in-Dynamic-Environments, inspected at commit c040073969a6111f4c652d08e8de8629d70b2317. The research artifact is archived at Zenodo under DOI 10.5281/zenodo.22092648. Relevant implementation components include DroneSecurity.hpp, CommandPolicy.hpp, PeerPacketAuth.hpp/.cpp, SwarmSecurity.hpp/.cpp, SafetyManager.hpp/.cpp, V2XMeshNetwork.cpp, and the runtime integration in main.cpp.

Reproduction of this paper's analytical tables requires only the published coefficients and thresholds. Reproduction of repository tests requires the repository toolchain and dependencies described by its build documentation. Stored performance figures should be interpreted within the environment recorded in the repository benchmark artifacts.

13 Conclusion

This paper presented a runtime cyber-resilience and trust-aware command-authorization framework for autonomous UAV swarms operating under GPS-denied and infrastructure-limited conditions. The central design principle is that cryptographic authenticity does not imply operational permission. Instead, the vehicle combines authenticated ingress with link, timing, localization, issuer, swarm, backend, mission, and survivability evidence to select a prioritized security state. That state explicitly controls remote authority and triggers local hold, return, or landing behavior when trust degrades.

The mathematical analysis exposed several useful properties of the implementation. The composite link-integrity score is informative but cannot override hard fail-closed predicates. Tamper evidence accumulates across distinct event classes with finite decay windows. Non-emergency command authority is monotonic under trust degradation, and locally triggered safe-return or landing behavior cannot be replaced by ordinary remote commands. Command acceptance is additionally bounded by freshness, issuer trust, formation-specific safety checks, and a final independent safety supervisor. The emergency-land override remains an explicit trust boundary that depends on authenticated ingress.

Repository tests demonstrate rejection of tampering, replay, wrong-secret frames, and broken command-history continuity, while stored benchmark evidence quantifies the software authentication path without claiming flight-level real-time guarantees. The resulting contribution is therefore a software-grounded, mathematically explicit framework for connecting cybersecurity evidence to actuator authority. Future work should add per-node asymmetric identity, hardware-in-the-loop attack campaigns, RF adversarial testing, threshold calibration, and machine-checked verification of the state and command invariants.

Statements and Declarations

Funding

The author declares that no funds, grants, or other support were received specifically for the preparation of this manuscript.

Competing Interests

The author declares no competing interests.

Author Contributions

M.S.I.S. conceived and designed the study, developed the research methodology and system architecture, formulated the mathematical cyber-resilience and command-authorization models, implemented and analyzed the software framework, designed the validation methodology, analyzed and interpreted the results, prepared the figures and tables, and wrote, reviewed, and approved the final manuscript.

Data Availability

No human-participant or proprietary dataset was used. The analysis is based on software source, deterministic policy scenarios, repository validation artifacts, and stored benchmark results. The associated research artifact is available at https://doi.org/10.5281/zenodo.22092648.

Code Availability

The implementation basis is available in the public GitHub repository smshagor-dev/UVA-GPS-Denied-Navigation-in-Dynamic-Environments. The manuscript records the inspected source snapshot to support reproducibility.

Ethics Approval

Not applicable. The study concerns software architecture, mathematical analysis, and software validation and involves no human participants or animals.

Consent to Participate

Not applicable.

Consent for Publication

Not applicable.

References

[1] Küçükerdem, H., Yilmaz, C., Kahraman, H.T., Sönmez, Y.: Autonomous control of unmanned aerial vehicles: applications, requirements, challenges. Cluster Comput. 28, 734 (2025). https://doi.org/10.1007/s10586-025-05418-6

[2] Alqudsi, Y., Makaraci, M.: UAV swarms: research, challenges, and future directions. J. Eng. Appl. Sci. 72, 12 (2025). https://doi.org/10.1186/s44147-025-00582-3

[3] Rose, S., Borchert, O., Mitchell, S., Connelly, S.: Zero Trust Architecture. NIST Special Publication 800-207 (2020). https://doi.org/10.6028/NIST.SP.800-207

[4] Alquwayzani, A.A., Albuali, A.A.: A systematic literature review of zero trust architecture for military UAV security systems. IEEE Access 12, 176033-176056 (2024). https://doi.org/10.1109/ACCESS.2024.3503587

[5] Han, K., Al Nuaimi, E., Al Blooshi, S., Psiakis, R., Yeun, C.Y.: Scalable authenticated communication in drone swarm environment. J. Internet Technol. 25(2), 255-265 (2024). https://doi.org/10.53106/160792642024032502008

[6] Constantinescu, N., Ticleanu, O.-A., Hunyadi, I.D.: Securing authentication and detecting malicious entities in drone missions. Drones 8(12), 767 (2024). https://doi.org/10.3390/drones8120767

[7] Illyass, K., Baig, Z.A., Syed, N.F.: Robust and lightweighted mutual authentication scheme for drone swarm networks. J. Netw. Comput. Appl. 242, 104264 (2025). https://doi.org/10.1016/j.jnca.2025.104264

[8] d'Ambrosio, N., Perrone, G., Romano, S.P., Urraro, A.: A cyber-resilient open architecture for drone control. Comput. Secur. 150, 104205 (2025). https://doi.org/10.1016/j.cose.2024.104205

[9] Lee, S., Segev, A.: Preventing drone hijacking by authenticating drone operator. J. Comput. Virol. Hack. Tech. 22, 11 (2026). https://doi.org/10.1007/s11416-025-00591-z

[10] Kholidy, H.A.: A novel blockchain approach for continuously authenticating UAVs to achieve zero trust principles in 5G-enabled UAVs. In: Azab, A., Malkiewicz, T. (eds.) Nordic e-Infrastructure Tomorrow, CCIS 2398, pp. 1-16. Springer, Cham (2025). https://doi.org/10.1007/978-3-031-86240-3_1

[11] Ortiz Barbosa, D., Burbano, L., Yang, S., Wang, Z., Cardenas, A.A., Xie, C., Cao, Y., et al.: Robust and efficient AI-based attack recovery in autonomous drones. In: Proceedings of the 1st GENZERO Workshop, Springer (2025). https://doi.org/10.1007/978-981-95-1050-4_2

[12] Dolev, D., Yao, A.: On the security of public key protocols. IEEE Trans. Inf. Theory 29(2), 198-208 (1983). https://doi.org/10.1109/TIT.1983.1056650

[13] Lamport, L., Shostak, R., Pease, M.: The Byzantine generals problem. ACM Trans. Program. Lang. Syst. 4(3), 382-401 (1982). https://doi.org/10.1145/357172.357176

[14] Krawczyk, H., Bellare, M., Canetti, R.: HMAC: Keyed-Hashing for Message Authentication. RFC 2104 (1997). https://doi.org/10.17487/RFC2104

[15] National Institute of Standards and Technology: Secure Hash Standard (SHS). FIPS PUB 180-4 (2015). https://doi.org/10.6028/NIST.FIPS.180-4

[16] Dworkin, M.: Recommendation for Block Cipher Modes of Operation: Methods and Techniques. NIST Special Publication 800-38A (2001). https://doi.org/10.6028/NIST.SP.800-38A

[17] Moriarty, K., Kaliski, B., Rusch, A.: PKCS #5: Password-Based Cryptography Specification Version 2.1. RFC 8018 (2017). https://doi.org/10.17487/RFC8018

[18] Joint Task Force: Security and Privacy Controls for Information Systems and Organizations. NIST Special Publication 800-53 Rev. 5 (2020). https://doi.org/10.6028/NIST.SP.800-53r5

[19] Shagor, M.S.I.: GPS-Denied Autonomous UAV Swarm Research Platform: Reproducibility Artifact. Zenodo (2026). https://doi.org/10.5281/zenodo.22092648