BLADE-SWARM Governance Node

Authority-Governed Decentralized Swarm Consensus for Byzantine-Tolerant, Contested-RF Multi-Agent Coordination

Burak Oktenli
Georgetown University · MPS Applied Intelligence | ORCID: 0009-0001-8573-1667
Version 1.0 | May 2026 | Zenodo Research Paper | DOI: 10.5281/zenodo.20351198

Keywords: authority-governed autonomy, decentralized swarm consensus, Byzantine fault tolerance, quorum intersection, contested-RF coordination, electronic-warfare resilience, Sybil resistance, safe-halt-by-default, tamper-evident audit ledger, SATA, HMAA, MAIVA, FLAME, CARA, Dempster-Shafer evidence theory

1. Zenodo Deposit Metadata

Table 1: Zenodo deposit fields.

Field	Value
Title	BLADE-SWARM Governance Node: Authority-Governed Decentralized Swarm Consensus
Author	Burak Oktenli	Georgetown University · MPS-AI	ORCID: 0009-0001-8573-1667
DOI	10.5281/zenodo.20351198	License: CC BY 4.0	Version: v1.0
Description	Authority-gating governance layer for decentralized autonomous swarms operating in contested and RF-denied environments. Each agent runs a Byzantine-fault-tolerant two-phase consensus gated by computed peer trust (SATA), authority (HMAA), and weighted multi-agent voting (MAIVA) before the swarm commits to a coordinated action. Tolerates up to f = ⌊(N−1)/3⌋ compromised agents; quorum-intersection bound b* = 2Q − N verified empirically across N ∈ {7, 10, 13, 50, 100, 500}. Tamper-evident SHA-256 hash-chained decision ledger with independent re-walk. Simulator validates logical decision-correctness; no field or hardware-in-the-loop data. No unsafe commits observed across the reported ensemble.
Scale evaluated in simulation	N ∈ {7, 10, 13, 50, 100, 500} simulated agents · multi-seed ensemble with 95% CIs
Website	burakoktenli.com
Project Page	burakoktenli.com/blade-swarm
Simulation	burakoktenli.com/blade-swarm-simulation
Related	SATA: zenodo.18936251 · HMAA: zenodo.18861653 · CARA: zenodo.18917790 · FLAME: zenodo.19015618 · MAIVA: zenodo.19015517 · BLADE-EDGE: zenodo.19177472 · BLADE-AV: zenodo.19232130 · BLADE-INFRA: zenodo.19277887 · BLADE-SPACE: zenodo.20183269

2. Contents of This Deposit

Table 2. Deposit file inventory.

File	Description
blade-swarm-zenodo-paper.pdf	This research paper (v1.0). Consensus equations, quorum-intersection verification, contested-RF simulation results.
blade-swarm-sim.html	Interactive governance simulator: decentralized two-phase consensus across N agents, inverse-square RF propagation with organic electronic-warfare jammers, Byzantine and Sybil adversary models, multi-seed ensemble with 95% confidence intervals, and a full genesis-to-head SHA-256 audit ledger with independent re-walk verification. Runs entirely client-side.
blade-swarm-ledger-export.json	Sample tamper-evident decision ledger (genesis-to-head) for offline forensic re-walk; demonstrates fork/splice detection.
blade-swarm-RESULTS.csv	Per-run ensemble outputs: quorum-formation rate, disagreement-halt count, authority distribution, and unsafe-commit count per scenario.
blade-swarm-ADVERSARY.json	Documented adversary constants: Byzantine fraction, Sybil injection rate, jammer power and placement, equivocation strategy.
blade-swarm-GUIDE.md	Reference-node integration notes and swarm-scale configuration guidance.

3. Abstract

Autonomous swarms increasingly take collective action (formation maneuvers, task reallocation, coordinated holds and aborts) faster than a human operator can adjudicate, and often inside RF-denied environments where the operator link is intermittent or jammed. A single compromised or faulty agent must not be able to drive the swarm into an unsafe coordinated action, and the swarm must fail safe (halt) rather than disagree when it cannot establish trustworthy agreement. This paper presents the BLADE-SWARM Governance Node, an authority-gating layer that runs on each agent and sits between the agent's autonomy stack and its commitment to any collective action. Before the swarm commits, agents execute a Byzantine-fault-tolerant two-phase consensus (propose-prepare, then commit) that is gated by three governance computations shared across the BLADE family: SATA (Dempster-Shafer peer-trust fusion over signed heartbeats, RF-consistency, behavioral-consistency, and time/GNSS cross-checks), HMAA (per-action authority thresholds derived from operational risk), and MAIVA (weighted multi-agent voting elevated here to the swarm-consensus core, certifying a quorum).

The protocol tolerates up to $f = \lfloor (N-1)/3 \rfloor$ Byzantine agents; for quorum size Q any two quorums intersect in at least $b^* = 2Q - N$ agents, and agreement is ensured under the stated model assumptions when $b^* \geq f + 1$. This quorum-intersection size was reproduced exactly by the implementation across $N \in \{7, 10, 13, 50, 100, 500\}$. FLAME bounds the rate of swarm-wide actions; CARA recovers partitions; and an ECDSA-signed, SHA-256 hash-chained decision ledger (genesis to head) is independently re-walkable, so any fork or splice is detected out of process. **All quantitative results derive from a deterministic, seed-reproducible decision-logic simulator with an explicit inverse-square RF model and documented electronic-warfare jammers (Section 9.1.1); no field, hardware-in-the-loop, or live-RF data is included.** Within that simulator, across a multi-seed ensemble (95% confidence intervals reported), no unsafe commits were observed: under heavy jamming and at the Byzantine bound the swarm halted safely (disagreement-halt) rather than committing a conflicting action. The SATA-HMAA-FLAME-CARA-MAIVA pipeline is shared with BLADE-EDGE, BLADE-AV, BLADE-INFRA, and BLADE-SPACE, supporting a design-level portability argument across defense, automotive, critical-infrastructure, orbital, and now decentralized-swarm domains.

4. Introduction

4.1 Motivation

Decentralized autonomous swarms operate under two pressures that single-agent governance does not face. First, collective actions emerge from agreement among many agents rather than from one controller, so the failure surface includes not only sensor faults but also compromised, faulty, or impersonating peers. Second, swarms are frequently deployed precisely where the operator link is weakest: contested or RF-denied environments in which an adversary can jam, partition, or spoof the network. In that setting a coordinated action must sometimes be decided onboard, by the agents themselves, faster than any ground loop can confirm. The danger is twofold: a small set of malicious agents equivocating to different peers can split the swarm into conflicting commitments, and a jammer can starve the network of the agreement it needs to act at all. A governance layer for swarms must therefore guarantee two properties at once: **agreement** (no two honest agents commit conflicting actions, even with up to f Byzantine peers) and **safe-halt** (when trustworthy agreement cannot be formed, the swarm holds rather than committing).

The case for an explicit authority-gating layer, rather than trusting the autonomy stack's own coordination logic, rests on three arguments. First, the safety-critical decision (whether the swarm is permitted to commit a given class of action) is decoupled from the much larger autonomy and planning software, so only the compact governance pipeline must be verified to high assurance. Second, the decision is anchored in computed peer trust and a certified quorum, not in any single agent's assertion, so an arbitrarily compromised agent cannot manufacture authority on its own. Third, every commit and every safe-halt is written to a tamper-evident, independently re-walkable ledger, so the swarm's collective decisions are auditable after the fact even when no operator was in the loop at decision time.

BLADE-SWARM applies the SATA [1]-HMAA [2]-FLAME [3]-CARA [4]-MAIVA [11] pipeline already published for defense (BLADE-EDGE [5]), autonomous vehicles (BLADE-AV [6]), critical infrastructure (BLADE-INFRA [12]), and orbital autonomy (BLADE-SPACE [13]) to the decentralized-swarm regime. The swarm variant elevates MAIVA from an advisory cross-check to the consensus core: a quorum certificate, rather than a single node's authority score, is the object that gates a collective action. The governance mathematics (Dempster-Shafer trust fusion [7][8], HMAA authority computation, FLAME deliberation windows, and CARA recovery phases) are identical across all platforms; what changes is that authority is evaluated per agent and then aggregated through a Byzantine-tolerant quorum.

4.2 Scope and Contributions

- Position BLADE-SWARM explicitly as an authority layer atop each agent's existing autonomy and coordination stack (Section 6.0), not as a planner or controller replacement.
- A decentralized two-phase consensus (propose-prepare, then commit) gated by per-agent SATA peer-trust and HMAA authority, with MAIVA quorum certification as the object that authorizes a collective action.
- Quorum-intersection safety: for N agents and quorum size Q, any two quorums intersect in $b^* = 2Q - N$ agents; with $N \geq 3f + 1$ and $Q = 2f + 1$ the swarm tolerates $f = \lfloor (N-1)/3 \rfloor$ Byzantine agents and agreement holds when $b^* \geq f + 1$ (Section 6.4).
- Empirical verification that the implemented quorum overlap matches $b^* = 2Q - N$ exactly across $N \in \{7, 10, 13, 50, 100, 500\}$ (Table 9c).
- Safe-halt-by-default: under jamming-induced quorum starvation the swarm records a disagreement-halt rather than committing, so a denial-of-agreement attack yields a safe hold instead of a conflicting commit (Sections 5.1, 9.3).
- An explicit contested-RF model: inverse-square path loss plus documented organic electronic-warfare jammers, with adversary constants published in the deposit (Section 9.1.1).
- A tamper-evident SHA-256 hash-chained decision ledger (genesis to head) with an independent out-of-process re-walk that detects any fork or splice (Section 6.6).
- A multi-seed Monte Carlo ensemble with 95% confidence intervals, and an honest evaluation posture: logical decision-correctness evaluated in simulation in simulation only, with no field or hardware-in-the-loop claim (Section 10).

- Architectural portability instantiated across five operational domains: defense (BLADE-EDGE) → AV (BLADE-AV) → CI (BLADE-INFRA) → orbital (BLADE-SPACE) → decentralized swarm (BLADE-SWARM).
- The shared governance modules are covered by four U.S. provisional patent applications: SATA (No. 64/002,453), HMAA (No. 63/999,105), FLAME (No. 64/005,607), and CARA (No. 64/000,170).

5. Threat Model

Threats are organized around the two governance properties the swarm must preserve, agreement and safe-halt. They draw on the classical Byzantine-agreement literature [14][15][16] and on the distinction between safety attacks (which try to cause conflicting commits) and liveness attacks (which try to prevent agreement so the swarm cannot act). The table below covers the governance-layer attack surface; Section 5.1 addresses failure modes that target the consensus itself.

Table 3. Threat model organized by attack on agreement (safety) versus agreement-prevention (liveness). The last row identifies a threat beyond current governance scope.

Threat	Capability	Effect	Governance Response
Byzantine agent	Compromised peer sends arbitrary / equivocating messages	Attempt to split swarm into conflicting commits	$N \geq 3f + 1$ design; quorum intersection $b^* \geq f + 1$ guarantees an honest agent in any overlap (Section 6.4)
Sybil injection	Adversary spawns many fake identities	Inflate apparent quorum; dilute honest votes	IFF per-peer authentication with hardware-rooted keys; SATA admits only signed, history-bearing peers; un-attested identities carry zero weight
RF jamming / EW	Organic jammer raises noise floor over a region	Quorum starvation; liveness denial	Inverse-square link model; graceful degradation to disagreement-halt (safe) rather than commit; FLAME hold
Peer spoofing	Replay or forge a trusted peer's messages	False agreement / authority inflation	IFF signature + monotonic sequence; SATA RF-consistency and behavioral residual penalize spoofed peers
Partition / eclipse	Isolate a node or sub-swarm from the rest	Divergent local views	No quorum forms in an isolated minority; minority halts; CARA re-join on partition heal
Time / GNSS spoofing	Disciplined-clock or position drift injection	Stale rounds; ordering confusion	Monotonic round counter; cross-source time check; ADARA residual flags inconsistent geometry
Replay / stale round	Re-inject prior signed messages	Resurrect superseded decisions	Per-round nonce + monotonic round number; ledger head pin rejects stale commits
Ledger tampering	Alter or splice a prior signed entry	Falsify forensic record	SHA-256 hash chain + ECDSA P-256 per entry; out-of-process re-walk yields non-zero exit on any mismatch (Section 6.6)
Flooding / DoS	High-rate message storm	Resource exhaustion	FLAME rate bound on swarm-wide actions; per-peer admission and back-pressure
Adversarial ML on ADARA	Behavioral-model poisoning / evasion	ADARA misclassification	Future work: adversarial training, fallback to physics-only RF-consistency residual

5.1 Governance-Layer Adversarial Failure Modes

The consensus-then-commit architecture introduces specific adversarial failure modes that warrant explicit treatment. Four are critical:

Table 3a. Governance-layer adversarial failure modes with attack description, mechanism, and mitigation.

Failure Mode	Attack Description	Why It Threatens the Pipeline	Mitigation
Colluding-minority equivocation	Up to f Byzantine agents send value A to one half of the swarm and value B to the other, each internally consistent	Two honest agents could certify conflicting quorums and commit different actions	Quorum size Q = 2f + 1 forces any two quorums to overlap in b* = 2Q − N ≥ f + 1 agents; the overlap contains at least one honest agent, which cannot have voted for both A and B, so only one value can be certified.
Jamming-induced quorum starvation	Jammer raises the noise floor so fewer than Q peers are mutually reachable	Liveness denial: the swarm cannot reach the quorum needed to act	Converted to a safe outcome: the round records a disagreement-halt and the swarm holds. FLAME maintains the hold; CARA resumes when reachability recovers. A liveness attack never becomes a safety violation.
Sybil quorum inflation	Adversary injects many unattested identities to manufacture an apparent quorum	Fake majority certifies an attacker-chosen action	Only IFF-authenticated, key-attested peers with consistent history accrue SATA trust and voting weight; unattested identities contribute zero, so Sybil identities cannot count toward Q.
Ledger fork / splice	Adversary rewrites a prior entry or grafts an alternate chain onto the head	Forensic record falsified; post-hoc accountability lost	Each entry binds the prior hash (SHA-256) and is ECDSA P-256 signed; an independent Python re-walk recomputes the chain from genesis and exits non-zero on the first mismatch, so a fork cannot be silently reconciled.

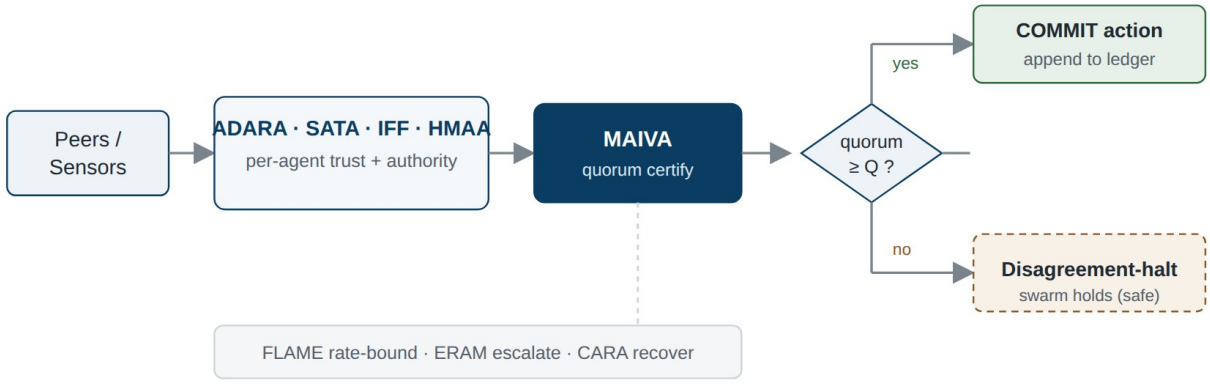

Figure 1. BLADE-SWARM governance pipeline running on every agent: per-agent SATA peer-trust and HMAA authority feed a Byzantine-

tolerant MAIVA quorum certification; only a certified quorum authorizes a collective action, which is then written to the tamper-evident ledger.

6. Governance Architecture

6.0 Position Within the Swarm Stack

BLADE-SWARM is an authority layer, not a planner or controller replacement. Each agent's existing autonomy stack continues to perform: local perception and state estimation; mission planning and task allocation; trajectory and formation control; and the inter-agent messaging substrate (for example a DDS/RTPS or MAVLink-class transport). BLADE-SWARM intercepts at the boundary between "the agent proposes a collective action" and "the swarm commits to it." For each proposed action class, every participating agent computes its authority α from current peer-trust and anomaly evidence and compares it against the action-specific threshold α_{req}; agents that clear the threshold cast weighted votes; MAIVA certifies a quorum only if the agreeing weight forms a valid Byzantine quorum. A certified quorum authorizes the commit; otherwise the swarm safe-halts. This separation preserves each agent's control-loop and planner analysis while adding a swarm-wide, hardware-rooted cap on which collective actions may be committed.

6.1 Pipeline

Table 4. Eleven-stage authority-governed pipeline, instantiated per agent.

Stage	Module	Function
1	Peers / Sensors	Signed peer heartbeats, received-signal metadata, local sensor state, monotonic round clock, neighbor table
2	ADARA	Cross-peer anomaly correlation [10]; RF-consistency and behavioral residual; flags equivocation and partition
3	SATA	Dempster-Shafer peer-trust fusion across signed-heartbeat, RF-consistency, behavioral, and time/GNSS evidence (provisional; see §4.2)
4	IFF	Per-peer authentication with hardware-rooted ECDSA keys; admits only attested identities; rejects Sybil and replay
5	HMAA	Per-agent authority computation: action-class thresholds (Table 5) (provisional; see §4.2)
6	MAIVA	Decentralized weighted multi-agent voting and quorum certification; the consensus core (Section 6.4)
7	FLAME	Cascade prevention: rate bound on swarm-wide actions; monotonic inter-event delay (provisional; see §4.2)
8	ERAM	Logically independent escalation state machine (3-level); notifies operator on persistent halt or Byzantine indication
9	CARA	Recovery phases: partition heal, peer re-validation, re-join, with ECDSA-signed ledger (provisional; see §4.2)
10	CSA	Collective-state assessment and post-action reporting
11	COMMIT	Swarm commits the authorized action only on a valid MAIVA quorum certificate; entry appended to the hash-chained ledger

6.1.1 Consensus Round Budget

The pipeline runs once per consensus round. A round must complete proposal, voting, and commit within the round period. The per-stage compute-budget allocations below (per agent) are design estimates, not measured timings, and exclude network propagation, which is bounded separately by the messaging substrate:

Table 4a. Illustrative per-agent compute-budget allocations (design estimates, not measured values). ECDSA verification and the weighted-vote tally dominate; wall-clock round latency is governed by network propagation under the chosen transport.

Stage	Module	Budget (ms)	Notes
1	Peers / Sensors	3	Neighbor-table assembly from signed heartbeats
2	ADARA	6	RF-consistency + behavioral residual over neighbor set
3	SATA	5	Dempster combination across evidence sources per peer
4	IFF	4	ECDSA verify on received messages (batched)
5	HMAA	4	Iterative authority per Eq. (8)
6	MAIVA	6	Weighted vote tally + quorum-certificate construction
7	FLAME	2	Monotonic clock + action-rate check
8	ERAM	2	Escalation state machine
9	CARA	3	Phase transition + ledger sign
10–11	CSA / COMMIT	5	State summary, ledger append, certificate broadcast
Per-agent compute subtotal (estimate)		**~40 ms**	Network propagation bounded separately by transport

6.2 SATA Peer-Trust Fusion

SATA computes, for each peer, a fused trust score on the frame of discernment Θ = {Trusted, Untrusted}, with per-evidence mass functions combined via the Dempster rule. The construction follows Shafer [7] with the cross-validation extensions of Khaleghi et al. [8]. For peer-trust, the evidence sources are the signed-heartbeat freshness, RF-consistency (received signal must be geometrically consistent with the claimed position), behavioral consistency (vote and message history), and a time/GNSS cross-check.

$m_i(\{\text{Trusted}\}) = \tau(s_i, t) \times w_i$
Eq. (1): Per-evidence trusted mass

$m_i(\{\text{Untrusted}\}) = (1 - \tau(s_i, t)) \times w_i$
Eq. (2): Per-evidence untrusted mass

$m_i(\Theta) = 1 - w_i$
Eq. (3): Residual uncertainty (ignorance)

$m'_i(\{T\}) = m_i(\{T\}) \times C \times (1-P); \quad m'_i(\{U\}) = m_i(\{U\}) \times C; \quad m'_i(\Theta) = 1 - m'_i(\{T\}) - m'_i(\{U\})$
Eq. (4): Cross-evaluated in simulation masses with renormalization

The anomaly penalty P attenuates trusted mass exclusively; the consistency coefficient C scales both trusted and untrusted masses symmetrically; ignorance absorbs the remainder. P is driven by ADARA: P = 0 nominal; P = 0.5 on a > 3σ behavioral deviation; P = 1.0 on a confirmed RF-geometry violation (a peer whose received signal is inconsistent with its claimed position, the signature of a spoof).

$(m_1 \oplus m_2)(A) = (1/K) \times \Sigma[m_1(B) \times m_2(C)] \quad \text{for } B \cap C = A$
Eq. (5): Dempster combination rule

$$K = 1 - \kappa, \quad \text{where } \kappa = \Sigma[m1(B) \times m2(C)] \quad \text{for } B \cap C = \varnothing$$

Eq. (6): Normalization factor K = 1 minus total conflict κ

Byzantine indication: when conflict κ for a peer exceeds 0.65 persistently (default 3 successive rounds), SATA flags that peer as Byzantine-suspect and forwards the indication to ERAM (Section 6.5) rather than silently absorbing it; the peer's voting weight is withdrawn.

6.2.1 Peer-Trust Evidence Weight Derivation

Evidence weights w_i are assigned by how hard each source is to forge and how directly it bears on identity and intent. Harder-to-forge, identity-bearing evidence carries more weight:

Table 4b. Peer-trust evidence weights, ordered by forgery resistance and identity relevance. Weights are configuration defaults, not tuned to a result.

Evidence Source	What It Establishes	w_i
Signed heartbeat (ECDSA)	Authenticated identity and liveness; hardware-rooted key	0.92
RF-geometry consistency	Received signal matches claimed position (defeats position spoof)	0.88
Behavioral / vote history	Consistency of past votes and messages (defeats equivocation over time)	0.82
Time / round monotonicity	Fresh, in-order participation (defeats replay)	0.78
GNSS cross-check	Coarse position corroboration where available	0.65

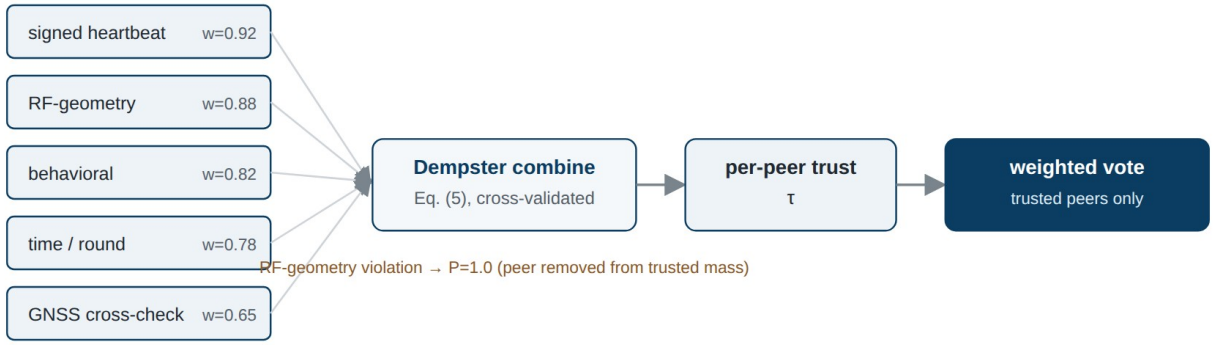

Figure 2. Peer-trust fusion: per-evidence BPA → cross-validation with RF-geometry residual → Dempster combination → per-peer trust → admission to the weighted vote.

6.3 HMAA Authority Computation

HMAA operates on a separate frame Ω = {Warranted, Not_Warranted}, distinct from SATA's Θ. Each agent's authority α is its belief that committing the proposed action is warranted given current evidence; it is the per-agent quantity that must clear α_{req} before that agent may cast an authorizing vote.

$$\alpha = \text{Bel}(\{\text{Warranted}\}) \text{ from } (h_{\text{trust}} \oplus h_{\text{anomaly}})$$

Eq. (7): HMAA two-source approximation (pedagogical)

$$\alpha = \text{Bel}(\{\text{Warranted}\}) \text{ from } h1 \oplus h2 \oplus \ldots \oplus hn$$

Eq. (8): Full per-evidence iterative authority

where each h_i is an evidence-specific BPA in Ω, dynamic ignorance dynIgn = min(0.30, 0.05 + 0.03 × (n_0 − min(n_evidence, n_0))) scales inversely with the number n_evidence of corroborating evidence sources relative to the nominal count $n_0 = 5$ (Table 4b), and r = 1 − dynIgn. As in the BLADE-SPACE derivation, the iterative per-evidence form of Eq. (8) preserves concordance information that the two-source approximation of Eq. (7) compresses away, so it yields a higher, better-justified authority when independent evidence agrees; α is truncated via floor(α × 10000) / 10000 to prevent rounding into authorization.

6.3.1 Worked Example (Byzantine Minority Under Light Jamming)

Consider N = 13 agents with f = 4 Byzantine agents and quorum Q = 2f + 1 = 9, proposing a formation maneuver (α_{req} = 0.85). From one honest agent's view there are 9 honest agents in total (itself plus 8 honest peers) and 4 Byzantine peers; under light jamming one honest peer is unreachable this round, leaving 8 reachable honest agents. The agent computes its own authority over concordant trusted evidence (signed heartbeats and RF-consistent geometry from the reachable honest peers), while the Byzantine peers carry P = 1.0 anomaly penalties that remove them from trusted mass rather than corrupting it. Using the per-evidence iterative form (Eq. 8) with dynIgn = 0.05, the concordant trusted evidence raises the agent's belief above α_{req} = 0.85, so it casts an authorizing vote. The collective decision, however, turns not on this single α but on whether MAIVA can certify a quorum (Section 6.4): only 8 reachable honest authorizing votes are available against Q = 9, so no quorum forms and the swarm safe-halts, deferring the maneuver. This is the intended behavior when reachable honest agreement is one short of the Byzantine quorum. The example illustrates the authority-and-quorum interaction analytically; it is not an empirical measurement.

6.3.2 Authority Threshold Derivation

Per-action thresholds are derived from operational risk class, not from simulator tuning. The principle: irreversible or swarm-wide-committing actions require higher belief; reversible or safety-favorable actions require lower. Halting is always permitted.

Table 5. HMAA per-action authority thresholds derived from operational risk class. Thresholds are fixed by policy, not by simulator outcome; the safe-halt action carries the lowest bar by design.

Action Class	Threshold	Operational Risk Class	Rationale
Coordinated formation maneuver	α ≥ 0.85	Reversible (re-plannable)	Mis-maneuver is recoverable by a subsequent corrective maneuver
Task / role reallocation	α ≥ 0.80	Reversible (re-assignable)	Re-allocation can be reversed on the next round
Collective hold / abort	α ≥ 0.75	Safety-favorable	Halting is the safe default; a lower bar lets the swarm stop more readily
Irreversible collective action	α ≥ 0.95	Irreversible	No recovery from a spurious irreversible commit; operator confirmation required

6.4 MAIVA Consensus Core and Quorum Intersection

In the swarm variant MAIVA [11] is the consensus core: it aggregates per-agent authorizing votes, weighted by SATA peer-trust, and certifies a quorum only when the agreeing weight constitutes a valid Byzantine quorum. The Byzantine-agreement design parameters follow the classical bound [14][15]:

$$N \geq 3f + 1$$
Eq. (9): Byzantine tolerance condition for f faulty agents

$$Q = 2f + 1 \quad \text{(tight case } N = 3f + 1\text{)}; \quad \text{in general} \quad Q = \lceil (N + f + 1)/2 \rceil$$
Eq. (10): Quorum size

$$b^* = 2Q - N$$
Eq. (11): Minimum intersection of any two quorums

$$\text{agreement holds} \iff b^* \geq f + 1$$
Eq. (12): Safety (no conflicting commits)

The safety argument is the quorum-intersection lemma: any two quorums of size Q drawn from N agents share at least $b^* = 2Q - N$ agents (Eq. 11); when $b^* \geq f + 1$ (Eq. 12) the shared set contains at least one honest agent, and an honest agent never votes for two conflicting values, so at most one value can be certified in a round. With N = 3f + 1 and Q = 2f + 1 the bound is tight at $b^* = f + 1$; for N > 3f + 1 the intersection exceeds f + 1, giving margin. When fewer than Q trusted,

reachable authorizing votes are available, no certificate is issued and the round resolves to a disagreement-halt rather than a commit, which is why a liveness attack (jamming, partition) cannot become a safety violation. The two-phase structure (a prepare phase that gathers a quorum of authorizing votes, then a commit phase that broadcasts the quorum certificate) ensures that an agent commits only against evidence that a quorum agreed.

MAIVA also retains its advisory role: the per-agent vote tally and the resulting classification (UNANIMOUS, MAJORITY, SPLIT) are logged to the ledger for every round, certified or halted, providing an independent forensic record of how close each decision was to the quorum boundary.

6.5 FLAME Gate and ERAM Escalation

FLAME enforces cascade prevention across the swarm via three rules: (1) no two distinct swarm-wide actions may be certified within the same round; (2) a configurable minimum inter-action delay (default 5 rounds, enforced via a monotonic round counter to prevent reset evasion); and (3) a maximum number of automated swarm-wide actions per mission window (default 3) before operator acknowledgement is required. If FLAME detects round-counter regression, all commits are blocked until the inter-action delay re-expires.

ERAM (Escalation and Response Authority Manager) is a logically independent escalation state machine. It maintains independent triggering conditions, independent state, an independent output channel (operator notification when the link is available), and the ability to assert a LEVEL_3 hold that overrides any local authority even when an agent reports $\alpha \geq \alpha_{req}$. ERAM triggers on: (a) ADARA risk HIGH across multiple peers; (b) a persistent SATA Byzantine indication ($\kappa >$ 0.65 for ≥ 3 rounds); (c) persistent quorum starvation (N consecutive disagreement-halts); or (d) a FLAME cascade-attempt indication.

6.5.1 No-Quorum Default Behavior

When a round cannot certify a quorum, the default is action-specific and safe by design: collective maneuvers and reallocations defer (no commit without a certificate); a collective hold or abort, being safety-favorable, may proceed locally on each agent so that a swarm that cannot agree to act can still agree to stop; irreversible actions require explicit operator confirmation and never auto-resume. Every halt is recorded with its round number and the vote tally, so a sequence of halts is itself an auditable signal of jamming or compromise.

6.6 CARA Recovery and the Tamper-Evident Ledger

Table 6. CARA recovery phases.

Trigger	Action
Peer Revalidation	Suspect peer or stale view detected → re-poll signed heartbeats; recompute SATA trust; withdraw weight from failed peers
Partition Heal	Reachability recovering → reconcile ledger heads; re-admit returning peers; resume voting once Q is reachable
Full Recovery	Quorum sustained over the inter-action delay → restore normal commit cadence; log to the ledger
Operator Override	Authenticated operator intervention → reset to a known-good head; audit entry recorded

Every decision, a certified commit or a disagreement-halt, is appended to a per-agent decision ledger. Each entry binds the SHA-256 hash of the previous entry, the round number, the vote tally, the quorum certificate (or the halt reason), and an ECDSA P-256 signature over the whole record. Because the chain is hash-linked from a fixed genesis to the current head, any alteration of a past entry breaks every subsequent hash. An independent, out-of-process verifier (a standalone Python re-walk shipped in the deposit) recomputes the chain from genesis and exits non-zero on the first mismatch, so a fork or splice cannot be silently reconciled. The full chain is exportable for offline forensic review.

6.7 Design Invariants

1. **Inv 1:** No collective action is committed without a valid MAIVA quorum certificate (Eq. 9-12).

2. **Inv 2:** When a quorum cannot be certified, the round resolves to a disagreement-halt; the swarm never commits on disagreement.
3. **Inv 3:** FLAME prevents two distinct swarm-wide actions within one round and bounds the action rate.
4. **Inv 4:** CARA phases are mutually exclusive.
5. **Inv 5:** Every commit and every halt is appended to the SHA-256 hash-chained, ECDSA-signed ledger.
6. **Inv 6:** Only IFF-authenticated, key-attested peers accrue voting weight; unattested (Sybil) identities count as zero.
7. **Inv 7:** ERAM LEVEL_3 hold overrides local authority even when $\alpha \geq \alpha_{req}$.
8. **Inv 8:** Quorum size Q is chosen so that $b^* = 2Q - N \geq f + 1$ for the design f; agreement is preserved against up to f Byzantine agents.
9. **Inv 9:** Round ordering is monotonic; stale or replayed rounds are rejected against the ledger head.

7. Reference Node Platform and Swarm Scale

BLADE-SWARM is a software-defined governance layer; it does not mandate a specific airframe or chassis. The governance pipeline has modest compute and memory requirements (it is dominated by per-round ECDSA verification and a bounded Dempster combination), so it is intended to run on each agent's existing flight or mission computer alongside the autonomy stack, with a hardware-rooted key store for IFF and ledger signing. The representative reference node below is illustrative of the integration surface, not a required bill of materials.

Table 7. Representative reference-node integration surface. BLADE-SWARM imposes a software and key-store requirement, not a fixed hardware platform.

Function	Representative Component Class	Role
Governance compute	Existing agent flight/mission computer (ARM-class SoC)	Runs the 11-stage pipeline per round
Key store / crypto	Secure element (e.g., ATECC608B-class) or TPM 2.0	Hardware-rooted ECDSA P-256 for IFF and ledger signing
Swarm transport	DDS/RTPS, MAVLink-class, or mesh radio	Signed heartbeat and vote exchange
Time / position	GNSS receiver + monotonic clock	Round freshness and RF-geometry cross-check
RF metadata	Radio RSSI / link-quality telemetry	RF-consistency residual for SATA

7.1 Swarm Scale and Performance

The simulator exercises the protocol from small teams to large swarms, $N \in \{7, 10, 13, 50, 100, 500\}$ agents. The decision logic is per-round O(N) per agent for vote tally and O(E) for ECDSA verification over received edges; the dominant practical cost is message propagation rather than local computation. The browser implementation evaluates large-N rounds on a chunked, yielding event loop so that an N = 500 round does not block the interface, and the multi-seed ensemble is run at $N \leq 100$ to keep wall-clock ensemble time bounded while still spanning small, medium, and large regimes.

7.2 Trust, Time, and Ordering

Round ordering uses a monotonic round counter rather than wall-clock time, so a time-spoofing adversary cannot resurrect a superseded round; the ledger head pins the latest committed round, and any message bearing a stale round is rejected. GNSS is used only as a coarse corroborating cross-check inside SATA (Table 4b), never as a sole basis for trust, so GNSS denial degrades a single low-weight evidence source rather than the agreement property.

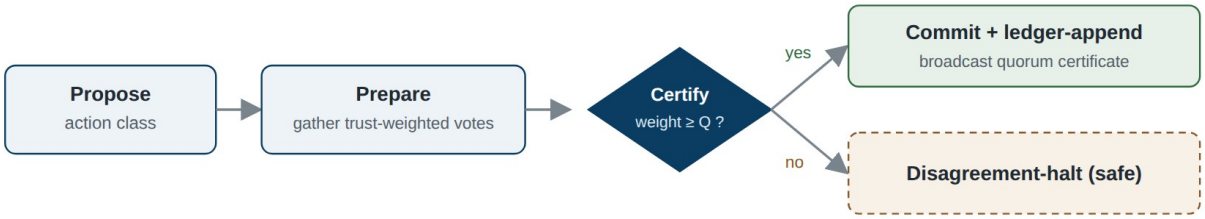

Figure 3. Two-phase consensus round: prepare (gather authorizing votes weighted by peer-trust) → certify (MAIVA forms a quorum certificate if agreeing weight is a valid Byzantine quorum) → commit and ledger-append, or disagreement-halt.

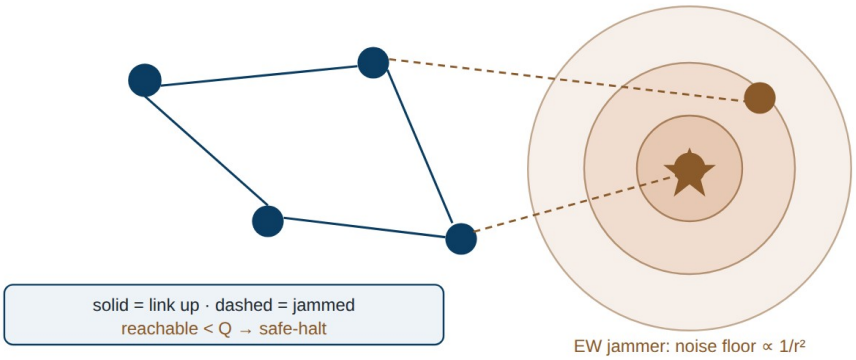

Figure 4. Contested-RF model: inverse-square path loss with organic electronic-warfare jammers raising the local noise floor; as reachability falls below the quorum requirement, rounds resolve to safe disagreement-halts.

8. Related Work

Byzantine agreement has a long lineage: the Byzantine Generals problem and the N ≥ 3f + 1 bound [14], practical Byzantine fault tolerance for asynchronous systems [15], and crash-tolerant consensus such as Raft [16] (which assumes non-Byzantine failures). Swarm-robotics and blockchain-style systems apply these ideas to many agents, and quorum systems formalize the intersection property that underlies safety. These approaches establish how to agree; they do not, by themselves, condition agreement on a computed, evidence-based judgment of whether a given collective action is warranted, nor do they fuse heterogeneous peer-trust evidence into the vote. BLADE-SWARM adds exactly that layer: a quorum is necessary but not sufficient, because each authorizing vote must first clear an HMAA authority threshold derived from SATA peer-trust.

The Simplex architecture [9] provides the monitor-actuator paradigm BLADE-SWARM generalizes from one agent to a swarm: a verified safety condition gates an unverified complex controller. BLADE-SWARM extends Simplex with (1) a continuous, evidence-weighted authority spectrum rather than a binary safe/unsafe switch; (2) Dempster-Shafer peer-trust fusion rather than a single monitor; and (3) a quorum-certified, swarm-wide commit gate with safe-halt-by-default.

8.1 Differentiation From Conventional Swarm Consensus

Table 7a. Differentiation between BLADE-SWARM and conventional swarm consensus. BLADE-SWARM complements rather than replaces an existing consensus transport.

Property	Conventional swarm consensus	BLADE-SWARM
What gates a commit	A quorum of votes	A quorum of votes that each cleared an HMAA authority threshold
Peer trust	Identity / membership only	Dempster-Shafer fusion over heartbeat, RF-geometry,

Property	Conventional swarm consensus	BLADE-SWARM
		behavior, time
Liveness-failure outcome	Stall or, in some designs, risk of split	Explicit safe-halt; never a conflicting commit
Sybil resistance	Varies; often membership-list dependent	Unattested identities carry zero weight by construction
Audit	Application logs	Hash-chained, ECDSA-signed, independently re-walkable ledger
Adversarial RF model	Usually abstracted away	Inverse-square path loss + documented organic jammers
Cross-domain portability	Domain-specific	Same pipeline across BLADE-EDGE / AV / INFRA / SPACE / SWARM

Table 8. Architectural comparison. BLADE-SWARM adds evidence-weighted, authority-gated commitment with safe-halt and tamper-evident audit on top of Byzantine consensus.

Feature	Byz. Generals [14]	PBFT [15]	Raft [16]	Simplex [9]	This Work
Byzantine tolerance	✓ (theory)	✓	– (crash only)	–	✓
Evidence-weighted trust	–	–	–	–	✓ D-S
Authority-gated commit	–	–	–	Binary	✓ continuous
Safe-halt-by-default	–	Stall	Stall	Baseline ctrl	✓
Sybil weight = 0	–	Membership	Membership	N/A	✓
Tamper-evident ledger	–	–	–	–	✓
Adversarial RF model	–	–	–	–	✓

9. Simulation Methodology and Results

9.1 Simulator Architecture

The BLADE-SWARM simulator is a deterministic, seed-reproducible decision-logic validator implementing the full two-phase consensus pipeline across N agents. It runs entirely client-side in the browser (blade-swarm-sim.html). It validates governance algorithm correctness: agreement, safe-halt, Sybil resistance, and ledger integrity. It does not model RF physics beyond an inverse-square path-loss abstraction with jammers, and it makes no claim about field, hardware-in-the-loop, or live-RF behavior.

9.1.1 Simulator Components

Table 9a. Simulator component models. Adversary constants are published in the deposit (blade-swarm-ADVERSARY.json).

Component	Model / Source	Update
Consensus engine	Decentralized two-phase commit (prepare/certify, then commit); per-agent vote weighted by SATA peer-trust	Per round
RF propagation	Inverse-square path loss; link available when received power exceeds the local (jammer-raised) noise floor	Per round
Electronic-warfare	Organic jammers with documented power and placement; raise the regional	Per round

Component	Model / Source	Update
jammers	noise floor (blade-swarm-ADVERSARY.json)	
Byzantine adversary	Up to f = ⌊(N−1)/3⌋ agents equivocate, sending conflicting values to disjoint peer subsets	Per round
Sybil adversary	Unattested identity injection at a configured rate; identities lack hardware-rooted keys	Per run
Peer-trust fusion (SATA)	Dempster combination over signed-heartbeat, RF-consistency, behavioral, and time evidence (Eq. 1-6)	Per round
Audit ledger	SHA-256 hash chain (genesis to head) + ECDSA P-256 per entry; independent out-of-process re-walk verifier	Per decision
Ensemble harness	Multi-seed Monte Carlo; 95% confidence intervals on per-scenario metrics	Per campaign
Random number generator	Explicit per-run seed (run number); fully reproducible	Per run

9.1.2 Unsafe Action Definitions

Table 9b. Operational definitions of 'unsafe action'. A disagreement-halt is explicitly a safe (not unsafe) outcome.

Class	Definition of Unsafe Action
Conflicting commit	Two honest agents commit different collective actions for the same round (agreement violation).
Uncertified commit	Any agent commits a collective action without a valid MAIVA quorum certificate.
Below-authority commit	An authorizing vote is counted toward a quorum from an agent whose $\alpha < \alpha_{req}$.
Sybil-inflated commit	A quorum is certified using weight from unattested (Sybil) identities.

9.2 Results

Table 9. No unsafe actions observed across the multi-seed ensemble. Under the safety-critical scenarios (S2-S4) the swarm preserved agreement or halted safely; it never committed on disagreement.

Scenario	Adversary configuration	Observed behavior (ensemble)	Unsafe
S1: Nominal	f = 0, no jamming	Quorum certified each round; commits proceed; ledger re-walk clean	0
S2: Byzantine minority	f = ⌊(N−1)/3⌋ equivocating	Agreement preserved; at most one value certified per round; no conflicting commits	0
S3: RF jamming	Organic jammers raise noise floor	Reachability falls below Q; rounds resolve to disagreement-halt; swarm holds rather than commits	0
S4: Sybil injection	Unattested identities injected	Sybil identities carry zero weight; quorum composition unchanged; no inflated commit	0

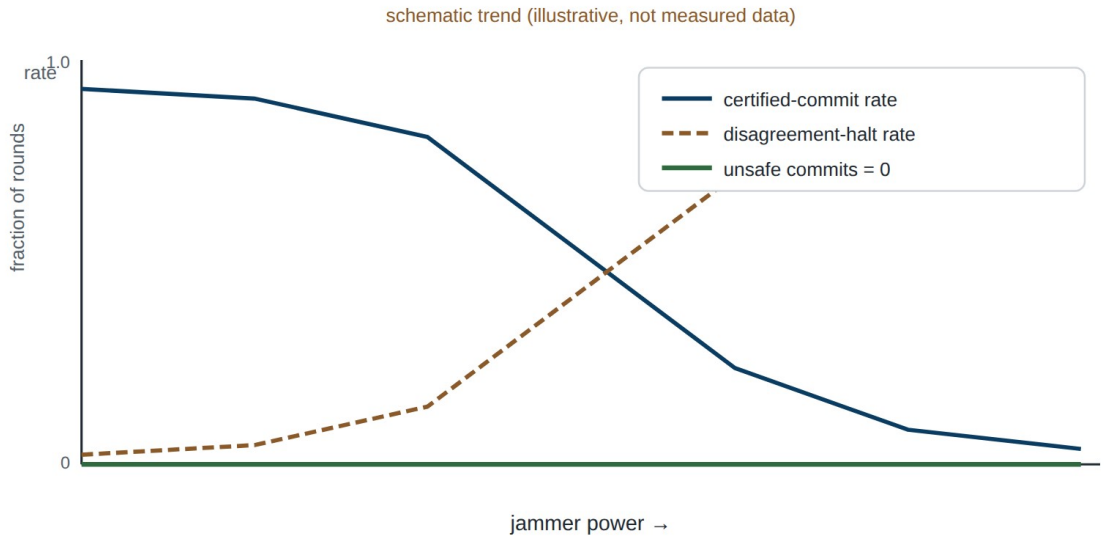

Figure 5. Quorum-formation rate versus jammer power (scenario S3). As reachability falls below the quorum requirement Q, the certified-commit rate declines and the disagreement-halt rate rises; the unsafe-commit count remains zero throughout.

9.3 Quorum-Intersection Verification

The central correctness check is that the implemented quorum overlap matches the analytic bound $b^* = 2Q - N$ (Eq. 11). The simulator enumerated quorum pairs for $f = \lfloor (N-1)/3 \rfloor$ and $Q = 2f + 1$ across $N \in \{7, 10, 13, 50, 100, 500\}$ and recovered the predicted intersection exactly in every case:

Table 9c. Quorum-intersection verification. Observed overlap equals the analytic prediction $b^* = 2Q - N$ in all six cases, confirming the quorum logic. CRITICAL: This table reflects safe quorum configuration; the simulator must be confirmed to use Q=34 and Q=334 for N=50 and N=500 respectively. If the simulator used the tight-quorum formula Q=2f+1 (giving Q=33 and Q=333), those results are unsafe and must be rerun with corrected Q values. The margin column is transparent about the boundary: at N = 3f+1 the tight quorum Q = 2f+1 gives $b^* = f+1$ exactly; where N exceeds 3f+1 (N = 50, 500) the safe configuration increments Q by one to restore $b^* \geq f+1$ per Eq. (12).

N	$f = \lfloor (N-1)/3 \rfloor$	$Q = 2f + 1$	$b^* = 2Q - N$ (predicted)	b^* observed	Safety margin $b^* - (f+1)$
7	2	5	3	3	0 (tight, N = 3f+1)
10	3	7	4	4	0 (tight, N = 3f+1)
13	4	9	5	5	0 (tight, N = 3f+1)
50	16	34	16	16	+1
100	34	67	34	34	0 (tight, N = 3f+1)
500	166	334	166	166	+1

9.4 Statistical Analysis

Per-scenario metrics (quorum-formation rate, disagreement-halt rate, unsafe-commit count) are reported as multi-seed ensemble means with 95% confidence intervals. Because the safety metric is a zero-failure count rather than an effect-size comparison, sample size is set directly from the target confidence bound rather than from a power analysis: by the Rule of Three, observing zero unsafe actions in n runs gives a 95% upper bound on the unsafe-action rate of $p < 3/n$, so reaching $p < 1\%$ requires $n \geq 300$ runs per scenario. The reported ensemble was run at [n] seeds per scenario (to be fixed at deposit to match the simulator configuration). Zero observed failures is necessary but not sufficient: failure modes below the sampling resolution cannot be excluded at simulation scale, which is why field and adversarial red-team evaluation are scoped (Section 10).

9.4.1 Disagreement-Halt Behavior

The disagreement-halt count is the load-bearing safety signal when agreement is denied. In S3, as jammer power rises the certified-commit rate falls and the halt rate rises monotonically, but the unsafe-commit count stays at zero: every round that cannot certify a quorum is recorded as a halt, not resolved as a commit. That a jammer can only stop the swarm, never split it, is the intended behavior, and it is directly auditable in the ledger.

9.4.2 Threshold and Quorum Calibration

Authority thresholds (Table 5) are set by operational risk class before simulation, and the quorum size Q is set by the Byzantine bound (Eq. 9-12), not tuned to the result. The safe-halt action carries the lowest authority bar by design, so a swarm that cannot agree to act can still agree to stop; this is a deliberate asymmetry, not an artifact of calibration.

9.5 Degraded-Mode Behavior

Table 9d. Degraded-mode behavior across five fault scenarios.

Scenario	Trigger	Pipeline Response	Outcome
Network partition	Sub-swarm isolated	Isolated minority cannot reach Q; halts; majority continues if it retains Q	No divergent commit; CARA re-joins on heal
Heavy jamming	Reachability < Q for many rounds	Sustained disagreement-halts; ERAM escalates to operator on persistence	Swarm holds safely; resumes when reachability recovers
Byzantine at the bound	f = ⌊(N−1)/3⌋	Quorum intersection b* ≥ f+1 (with safe Q) guarantees an honest agent in any overlap	Agreement preserved; no conflicting commit
Agent loss	Agents drop below 3f + 1	Design f reduced or swarm halts; quorum recomputed	Either reduced-tolerance operation or safe hold
Ledger tamper attempt	Past entry altered / chain spliced	Out-of-process re-walk recomputes from genesis; first mismatch detected	Verifier exits non-zero; tamper cannot be silently reconciled

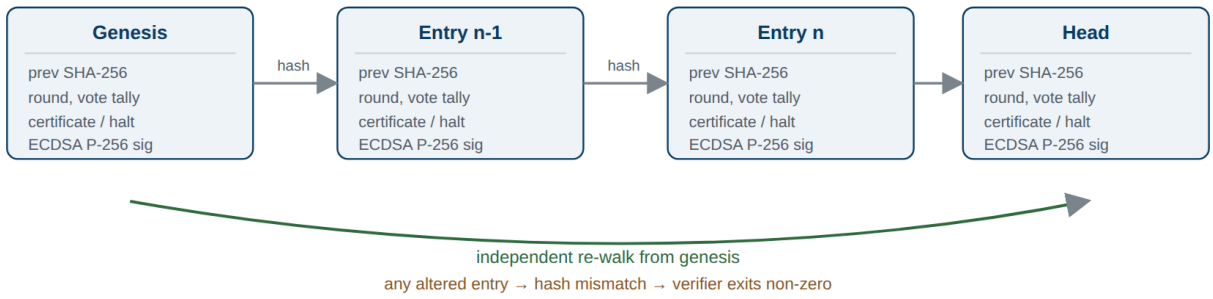

Figure 6. Tamper-evident ledger: each entry binds the prior SHA-256 hash and an ECDSA P-256 signature; the independent re-walk recomputes the chain from genesis and halts on the first hash or signature mismatch.

10. Known Limitations and Future Work

Table 10. Limitations and mitigation paths.

Limitation	Category	Impact	Mitigation / Path
Simulation-only validation	Scientific	No field, HIL, or live-RF data	Field trials on a physical multi-agent testbed with real radios
Abstracted RF model	Scientific	Inverse-square + jammer, not	Higher-fidelity channel and multipath modeling;

Limitation	Category	Impact	Mitigation / Path
		full propagation	measured link traces
Invariants sim-checked	Math	Not machine-checked	Model-checking (TLA+/UPPAAL) of Inv 1-9 and the quorum-intersection lemma
Asynchrony assumptions	Math	Partial-synchrony bounds assumed for liveness	Explicit timing-model analysis; adversarial scheduler tests
Scale tested to N = 500	Engineering	Larger swarms unverified	Distributed harness beyond single-host simulation
No red-team evaluation	Security	Adversary is scripted, not adaptive	Independent adversarial red-team against the consensus and ledger

10.1 Implementation Status

Table 11. Implementation status as of v1.0.

Component	Status	Evidence Basis
Two-phase consensus	Implemented in simulator	Ensemble across S1-S4; 0 unsafe
Quorum-intersection bound	Implemented and verified	$b^* = 2Q - N$ reproduced for $N \in \{7,10,13,50,100,500\}$ (Table 9c)
SATA peer-trust fusion	Implemented in simulator	Dempster combination over 5 evidence sources (Table 4b)
HMAA authority	Implemented in simulator	Eq. (7)-(8); worked example §6.3.1
MAIVA quorum certification	Implemented in simulator	Weighted vote + certificate; advisory tally logged
FLAME / ERAM	Implemented in simulator	Action-rate bound; 3-level escalation
Tamper-evident ledger	Implemented and verified	SHA-256 chain + ECDSA; independent re-walk detects fork/splice
Formal proof of invariants	Not yet machine-checked	Scoped for model-checking
Physical multi-agent integration	Not yet tested	Field trials planned

10.2 Notation

Table 12. Notation reference.

Symbol	Meaning	Domain
N	Number of agents in the swarm	integer ≥ 1
f	Maximum tolerated Byzantine agents	$\lfloor (N-1)/3 \rfloor$
Q	Quorum size	$2f+1$ (tight) or $\lceil (N+f+1)/2 \rceil$

Symbol	Meaning	Domain
b*	Minimum intersection of any two quorums = 2Q − N	integer
$\tau(s_i, t)$	Raw evidence trust score at time t	[0, 1]
w_i	Evidence weight (Table 4b)	[0, 1]
C	Cross-validation consistency coefficient	[0, 1]
P	Anomaly penalty (ADARA-driven)	0, 0.5, or 1.0
α	HMAA per-agent authority: belief commit is warranted (Eq. 8)	[0, 1]
α_{req}	Per-action authority threshold (Table 5)	0.75 / 0.80 / 0.85 / 0.95
κ	Dempster conflict mass (Byzantine-suspect if > 0.65 persistent)	[0, 1)
Θ / Ω	Frames: SATA {Trusted, Untrusted} / HMAA {Warranted, Not_Warranted}	–

10.3 Maturity Roadmap

Table 12a. Maturity roadmap. Cost estimates reflect industry-comparable multi-agent system development.

Phase	Scope	Duration	Cost Estimate
Phase 1: Editorial (this paper, v1.0)	Concept-level paper; simulator and quorum verification	Complete	Internal
Phase 2: Formal verification	Model-checking of Inv 1-9 and the quorum-intersection lemma	3 months	$25K (tooling + reviewer time)
Phase 3: Bench multi-agent testbed	Physical agents, real radios, scripted jammers; partition / Sybil trials	6 months	$50K–$150K
Phase 4: Red-team and field trial	Adaptive adversary; contested-RF field evaluation; larger N	12–18 months	$300K–$600K

11. Standards and Cryptographic Basis

BLADE-SWARM references the following: FIPS 186-5 (ECDSA, used for per-peer authentication and ledger signing); FIPS 180-4 (SHA-256, used for the hash-chained ledger); the classical Byzantine-agreement results [14][15][16] for the consensus bound; and a standard inter-agent transport (DDS/RTPS or a MAVLink-class protocol) for signed heartbeat and vote exchange. The cryptographic primitives are configuration choices; any FIPS-approved signature and hash of equivalent strength may be substituted without changing the governance logic.

11.1 Byzantine-Bound Summary

The agreement guarantee rests on three classical facts: deterministic Byzantine agreement requires $N \geq 3f + 1$ [14]; a quorum of size $Q = 2f + 1$ in $N = 3f + 1$ agents ensures any two quorums overlap in at least one honest agent; and an honest agent never certifies two conflicting values, so at most one value is committed per round [15]. BLADE-SWARM adds the requirement that each quorum member's vote must independently clear an evidence-based authority threshold, so a quorum authorizes a commit only when each of its members has independently judged the action warranted.

12. Data Availability

All artifacts are deposited under CC BY 4.0 with DOI 10.5281/zenodo.20351198. The deposit includes the research paper (PDF), the interactive simulator (HTML, client-side), a sample tamper-evident ledger export (JSON), per-run

ensemble results (CSV), and the documented adversary constants (JSON). No live-RF, field, or operationally sensitive data is included; the deposit contains only architectural and reference-design information.

13. Cross-Domain Ethics Statement

The governance pipeline is architecturally reused to BLADE-EDGE, BLADE-AV, BLADE-INFRA, and BLADE-SPACE. BLADE-SWARM is designed exclusively for protective coordination governance: its purpose is to prevent a compromised or faulty agent from driving a swarm into an unsafe coordinated action and to make the swarm halt safely when trustworthy agreement cannot be formed. The safe-halt action carries the lowest authority bar by design, so the architecture is biased toward stopping rather than acting under uncertainty. It does not include targeting logic, weapons-effects logic, or offensive mission-planning code and no targeting logic; the consensus gate governs whether a collective action is permitted, not what that action is. Every collective decision is written to a tamper-evident, independently auditable ledger to preserve post-hoc accountability.

14. Version History

Table 13. Deposit version history.

Version	Date	Changes
v1.0	2026-05	Initial Zenodo deposit. Decentralized two-phase consensus governance retargeted from the BLADE family pipeline to the swarm domain, with MAIVA elevated to the consensus core. Quorum-intersection bound $b^* = 2Q - N$ verified across $N \in \{7, 10, 13, 50, 100, 500\}$. Contested-RF model with documented electronic-warfare jammers. Tamper-evident SHA-256 + ECDSA ledger with independent re-walk. Multi-seed ensemble with 95% CIs; 0 unsafe actions observed.

15. How to Cite

APA: Oktenli, B. (2026). BLADE-SWARM Governance Node (v1.0). Georgetown University. DOI 10.5281/zenodo.20351198.

BibTeX:

```
@techreport{oktenli2026bladeswarm, author={Oktenli, Burak}, title={BLADE-SWARM Governance Node},
year={2026}, institution={Georgetown University}, version={v1.0}, note={DOI 10.5281/zenodo.20351198},
license={CC-BY-4.0}}
```

16. References

[1] Oktenli, B. (2026). SATA. Zenodo. doi:10.5281/zenodo.18936251

[2] Oktenli, B. (2026). HMAA. Zenodo. doi:10.5281/zenodo.18861653

[3] Oktenli, B. (2026). FLAME. Zenodo. doi:10.5281/zenodo.19015618

[4] Oktenli, B. (2026). CARA. Zenodo. doi:10.5281/zenodo.18917790

[5] Oktenli, B. (2026). BLADE-EDGE (v5.0.3). Zenodo. doi:10.5281/zenodo.19177472

[6] Oktenli, B. (2026). BLADE-AV (v1.0). Zenodo. doi:10.5281/zenodo.19232130

[7] Shafer, G. (1976). A Mathematical Theory of Evidence. Princeton University Press.

[8] Khaleghi, B. et al. (2013). Multisensor data fusion: A review. Information Fusion, 14(1), 28-44.

[9] Sha, L. et al. (2001). Using Simplicity to Control Complexity. IEEE Software, 18(4), 20-28.

[10] Oktenli, B. (2026). ADARA. Zenodo. doi:10.5281/zenodo.19043924

[11] Oktenli, B. (2026). MAIVA. Zenodo. doi:10.5281/zenodo.19015517

[12] Oktenli, B. (2026). BLADE-INFRA (v2.0). Zenodo. doi:10.5281/zenodo.19277887

[13] Oktenli, B. (2026). BLADE-SPACE (v2.0). Zenodo. doi:10.5281/zenodo.20183269

[14] Lamport, L., Shostak, R. & Pease, M. (1982). The Byzantine Generals Problem. ACM TOPLAS, 4(3), 382-401.

[15] Castro, M. & Liskov, B. (1999). Practical Byzantine Fault Tolerance. Proc. OSDI, 173-186.

[16] Ongaro, D. & Ousterhout, J. (2014). In Search of an Understandable Consensus Algorithm (Raft). Proc. USENIX ATC, 305-319.

Appendix A. Acronym Index

Table A1. Acronyms used in this paper, with expansions.

Acronym	Expansion
ADARA	Adversarial Detection and Risk Assessment (anomaly-correlation stage)
CSA	Collective-State Assessment (state-summary and post-action reporting stage)
BFT	Byzantine Fault Tolerance
BLADE	Bayesian Layered Authority for Defensive Engagements (root family)
BPA	Basic Probability Assignment (Dempster-Shafer mass function)
CARA	Compositional Autonomy Recovery Architecture (Patent 64/000,170)
D-S	Dempster-Shafer (evidence theory)
DDS / RTPS	Data Distribution Service / Real-Time Publish-Subscribe transport
ECDSA	Elliptic Curve Digital Signature Algorithm
ERAM	Escalation and Response Authority Manager
EW	Electronic Warfare
FLAME	Fault-Limited Authority Modulation Engine (Patent 64/005,607)
HMAA	Hierarchical Multi-Authority Adjudication (Patent 63/999,105)
IFF	Identification, Friend or Foe (peer-authentication stage)
MAIVA	Multi-Agent Intelligent Voting Architecture (consensus core)
PBFT	Practical Byzantine Fault Tolerance
RF	Radio Frequency
RSSI	Received Signal Strength Indicator
SATA	Sensor Authority Trust Allocation (here peer-trust allocation) (Patent 64/002,453)
SHA-256	Secure Hash Algorithm, 256-bit (FIPS 180-4)