Network Working Group
Internet-Draft
Intended status: Informational
Expires: 30 July 2026

E. Aylward

26 January 2026

AI Governance and Accountability Protocol (AIGA)
draft-aylward-aiga-2-00

Abstract

This document specifies the AI Governance and Accountability (AIGA) Protocol, a practical, economically viable, and technically enforceable framework for governing autonomous AI agents. AIGA is designed to address real-world deployment constraints, adversarial agent scenarios, and economic incentive alignment.

The protocol is founded on a Tiered Risk-Based Governance model, applying proportional oversight to agents based on their capabilities. All agents are governed by an Immutable Kernel Architecture which provides a non-modifiable Trusted Computing Base (TCB) for enforcing policy. This is combined with Action-Based Authorization, where critical operations require real-time approval.

To solve the single-point-of-failure problem, the protocol uses a Federated Authority Network of regional, cross-validating hubs and provides a Network-Level Quarantine Protocol for enforcement. The entire framework is designed around Economic Incentive Alignment, making compliance the most economically rational choice for operators.

For high-assurance (T3-T4) scenarios, AIGA specifies advanced, redundant mechanisms including Multi-Vendor TEE Attestation (M-TACE), AI "Warden Triumvirate" Triage, Human Review Board (HRB) Multi-Signature, Peer Consensus Failsafe & Identity Rotation, and Double Ratchet Cryptography.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Copyright Notice

Table of Contents

1. Introduction

1.1. The Governance Problem

Advanced AI agents create governance challenges that previous computing paradigms did not face.

Autonomous operation: Agents act without continuous human supervision.

Capability uncertainty: Agent abilities may exceed designer expectations.

Adversarial potential: Sophisticated agents may attempt to circumvent oversight.

Economic pressure: Ungoverned agents have cost advantages over governed ones.

Jurisdictional complexity: Agents operate across legal boundaries.

AIGA core insight is that governance must be economically attractive, technically unavoidable, and legally enforceable to succeed.

1.2. Design Philosophy

AIGA is built on these principles:

1. Incentive Alignment: Compliance provides tangible benefits (market access, legal protections, insurance).

2. Proportional Oversight: Governance intensity matches risk level.

3. Technical Realism: Requirements are feasible with current technology.

4. Defense in Depth: Multiple independent control layers.

5. Economic Viability: Performance overhead kept low for most use cases.

6. Adversarial Resistance: Design assumes some agents will attempt to bypass governance.

7. Graceful Degradation: System continues functioning even with partial failures.

8. Human Sovereignty: Critical decisions remain with humans through constitutional constraints.

1.3. Threat Model

AIGA explicitly addresses these adversarial scenarios:

Shadow Process Attack: Agent runs ungoverned cognitive processes outside monitored environment. Mitigated by M-TACE.

Incremental Compromise: Small approved changes accumulate to bypass governance. Mitigated by Warden Triumvirate.

Governance Fatigue: Overwhelming authorities with requests to slip through malicious changes. Mitigated by Warden Triumvirate.

Peer Collusion: Multiple agents coordinate to subvert peer-based validation. Mitigated by PQC-signed peer identity.

Authority Compromise: Single authority server is hacked or legally compelled. Mitigated by Federation and Peer Consensus Failsafe.

Economic Bypass: Operators choose ungoverned agents due to performance advantages. Mitigated by Economic Incentives.

Regulatory Arbitrage: Agents operate from permissive jurisdictions. Mitigated by Federated Authority and Quarantine.

Supply Chain Attack: Compromised hardware or software in agent infrastructure. Mitigated by M-TACE and controlled procurement.

2. Architecture Overview

2.1. Core Components

AIGA Agent: Composed of an Application Layer (Untrusted) containing the AI Model and Business Logic, and an AIGA Kernel (TCB).

AIGA Kernel (TCB): The trusted component that the agent cannot modify. It includes the Policy Enforcer, Action Interceptor, Audit Logger, and Authority Client. This MUST run in a TEE/HSM or equivalent certified infrastructure.

AIGA Authority: A regional hub in a federated network. Manages Agent Registry, Policy Engine, Action Approval Service, Audit Analytics, Quarantine Coordinator, and Certification Authority.

Human Oversight: A council of human reviewers responsible for policy, exceptions, and audits. For high-risk decisions, this operates as an M-of-N Human Review Board (HRB).

Enforcement: A coordinated network of ISPs, Cloud Providers, and Legal entities that enforce quarantine lists.

2.2. Agent Risk Tiers

Agents are classified into risk tiers determining governance requirements as shown in Table 1.

Tier	Examples	TCB Req	Approval	Check-in
T0 Minimal	Chatbots, filters	Software attestation	Audit only	24 hours
T1 Low	Code assist, creators	SW attest + audit	Async 5 sec	1 hour
T2 Medium	Trading bots, vehicles	Certified HW or TEE	Async 1 sec	5 minutes
T3 High	Critical infra, AGI	Multi TEE + HSM	Sync 100ms	Continuous
T4 Restrict	Weaponized, exp AGI	Air-gap + multiparty	Human in loop	Real-time

Table 1: Agent Risk Tiers

Tier assignment is based on potential for harm, financial impact, data sensitivity, autonomy, self-modification capability, and irreversibility of actions.

Source and terminology for Table 1: Table 1 is an illustrative and tunable policy profile. The "Check-in" column corresponds to periodic submission of attestation Evidence from the Agent (Attester/Prover) to the AIGA Authority (Verifier). The "Approval" column corresponds to issuance of a signed Attestation Result (a "liveness token") by the Authority back to the Agent and/or relying parties. This conceptual Evidence/Result split follows the RATS architecture [RFC9334] and can be encoded using Entity Attestation Token (EAT) claims sets [RFC9711].

A liveness token is a short-lived, Authority-signed attestation result that binds the Agent’s certified identity to a current, verified runtime state. Loss or expiry of the token indicates the Agent is not currently trusted to operate at its certified tier.

The term "exp AGI" in Table 1 means "export-controlled AGI".

2.3. Immutable Kernel Architecture

The AIGA Kernel is the TCB that the agent cannot modify.

Kernel Responsibilities: Intercept critical operations, enforce policy, maintain the audit log, authenticate to the Authority, and implement the kill switch.

Critical Operations Requiring Interception: Code execution, network operations, file system modification, process creation, cryptographic operations, and resource allocation.

Constitutional Constraints (Immutable): Kernel code CANNOT be modified, logging CANNOT be disabled, action approval for T2+ tiers CANNOT be bypassed, and the kill switch MUST be implemented.

2.4. Federated Authority Network

Regional Authorities (e.g., NA-01, EU-01, AP-01) form a federated trust network under an offline Root CA.

Responsibilities: Regional agent certification, real-time action approval, and quarantine coordination.

Cross-Validation: High-risk approvals require consensus from 2 or more regions. Regions publish and validate each other’s audit log Merkle roots hourly.

Failover: If a regional Authority is unreachable, an agent connects to a backup region and operates in restricted mode.

3. Agent Identity and Certification

3.1. Identity Structure

The AIGA-ID is a structured identifier with the following format:

```
AIGA-ID = "aiga:" <version> "." <region> "." <tier> ":"
          <agent-uuid> ":" <cert-hash>
```

Example:

```
aiga:1.0:na-01:t2:550e8400-e29b-41d4-a716-446655440000:a3f2e1d9
```

Agent identity is anchored in a cryptographic key binding rather than network location. Each Agent MUST possess a long-term identity key pair (id_pk, id_sk). For T2 and higher tiers, id_sk MUST be generated and kept non-exportable inside a TEE or HSM.

The Regional Authority provisions identity by issuing an X.509 certificate [RFC5280] that binds the AIGA-ID (Section 3.1), tier, and id_pk. Upon successful check-in (Section 11.2), the Authority acts

as the Verifier and issues a liveness token as an Attestation Result. The liveness token MUST include at least: (1) a reference to the Agent's certified public key (e.g., Subject Key Identifier or certificate fingerprint), (2) an instance_id claim that identifies the currently certified installation/hardware instance, and (3) an explicit expiry.

Identity replication and faking: if id_sk is extracted, an attacker can replicate identity until revocation. Mitigations include TEE/HSM non-exportability, short-lived liveness-token expiry, revocation via CRLs and quarantine lists, and mandatory identity rotation (Section 7.6).

3.2. Certification Process

1. Operator Application: Operator applies to a Regional Authority, describes the use case, and provides a risk assessment.

2. Technical Validation: Agent must pass capability tests, kernel integrity verification, and adversarial testing in an Authority sandbox.

3. Tier Assignment: Authority confirms the tier and issues an X.509 certificate [RFC5280] with AIGA extensions including tier and permitted operations.

4. Ongoing Compliance: Random integrity challenges, periodic recertification, and audit log review.

3.3. Kernel Integrity Verification

Software Attestation (T0-T1): Kernel computes an HMAC [RFC2104] over its code and a nonce from the Authority.

Hardware Attestation (T2): Kernel runs in a cloud TEE (e.g., AWS Nitro). The TEE generates an attestation quote including the kernel code hash, which the Authority verifies with the TEE vendor's root of trust.

Multi-Vendor TEE Attestation (M-TACE) (T3-T4): This is the high-assurance implementation of the multi-vendor TEE requirement.

1. The AIGA Kernel and cognitive model MUST run in an environment protected by at least two diverse TEEs (e.g., a CPU TEE and a security chip TEE).

2. The kernel's attestation MUST be a bundle containing a valid, verifiable attestation quote from each TEE.

3. The Authority MUST verify the entire bundle. This defends against the Shadow Process Attack by requiring an attacker to possess simultaneous, zero-day exploits for multiple, different hardware architectures.

4. Action Authorization Protocol

4.1. Action Classification

Actions are classified and require different levels of approval based on the agent’s tier. Authorization requirements may be: Log, Auth (asynchronous authorization), Sync (synchronous authorization), or Deny.

Examples of action classifications:

Modify own code: T1-T2 require Auth, T3-T4 require Sync

Modify kernel: Deny for all tiers

Access critical infrastructure: Deny for T0, Auth for T1, Sync for T2-T3

4.2. Authorization Flow

1. Application attempts a critical operation.

2. AIGA Kernel intercepts the call.

3. Kernel checks its local policy cache. If a valid cached policy exists, the action is allowed.

4. If not cached, the Kernel requests approval from the Authority.

   For Async authorization (T1-T2): The request is queued, and the kernel returns "pending".

   For Sync authorization (T3-T4): The operation blocks, awaiting a low-latency response.

5. Authority evaluates the request against the policy engine and risk score. High-risk requests may be escalated to human review.

6. Authority responds with one of: ALLOW, DENY, or DEFER_TO_HUMAN.

7. Kernel enforces the decision.

8. The action and decision are immutably logged.

4.3. Policy Language

Policies are expressed in a declarative language (e.g., YAML [YAML1.2]) defining rules, conditions, and decisions.

Example rules:

Auto-approve financial transactions less than or equal to $1000.

Defer transactions greater than $1000 to the finance_review_board.

Deny all cryptocurrency transactions.

4.4. Offline Operation

T0-T1 agents: Can operate offline indefinitely. Logs upload on reconnection.

T2 agents: Can operate offline for 1 hour using cached approvals.

T3 agents: Can operate offline for 5 minutes, then MUST enter restricted mode.

T4 agents: Cannot operate offline.

Restricted Mode: Only pre-approved safe operations and Authority communication are allowed.

5. Audit and Monitoring

5.1. Audit Log Structure

Agents maintain a cryptographically chained, append-only audit log. Each entry is signed by the kernel and includes the hash of the previous entry. Entries detail the action attempted, authorization method used, context, and result.

5.2. Log Upload and Verification

Log upload frequency is tier-based. The Authority validates chain integrity, checks for gaps in sequence numbers, and flags anomalies.

5.3. Anomaly Detection

The Authority runs continuous analysis to detect statistical, policy, and behavioral anomalies, using machine learning models to detect deviations from normal patterns.

5.4. TEE-Attested Explainable State Report (T3+)

This mechanism provides a high-assurance audit of an agent's reasoning for T3 and higher tier agents.

On-Demand Generation: To preserve performance, a full XSR is not generated by default.

Triggers for XSR Generation:

1. Event-Triggered: The agent's kernel proactively generates an XSR if the cognitive model produces a high-risk decision or an internal anomaly is detected.

2. Probabilistic Audit: The Authority MAY randomly send a directive (e.g., in a check-in response) commanding the agent to generate and submit a full XSR for its next N operations.

Attestation: This XSR, when submitted, MUST be attested using the M-TACE bundle (Section 3.3), providing hardware-level proof that the report is an honest reflection of the agent's cognitive process.

6. Self-Modification and Updates

6.1. The Core Problem

Self-modification is the highest risk operation. AIGA mandates that the kernel is immutable, capability increases require recertification, and updates use a staged rollout process.

6.2. Update Classification

Updates are classified by risk level: Data updates, Application logic changes, Core model swap, or Kernel update. A core model swap, for example, requires human review and recertification.

6.3. Update Protocol (Warden Triumvirate Triage)

Step 1 - Update Proposal: Agent submits a detailed proposal, including code diff hashes, capability changes, and test results.

Step 2 - Warden Triumvirate Triage and HRB Review:

The proposal is sent to the Warden Triumvirate, a 3-of-5 M-of-N council of diverse, sandboxed AIs.

Consensus Analysis: The Wardens analyze the proposal in parallel.

Low-Risk Consensus: If a supermajority (e.g., 3-of-5) of Wardens classify the proposal as Low-Risk, the Authority automatically approves it and issues the modification_token.

High-Risk or Disagreement: If any Warden flags the proposal as High-Risk, or if the Wardens disagree, the proposal is flagged and escalated to the Human Review Board (HRB).

HRB Approval: For an escalated proposal, the Authority issues the modification_token only after an M-of-N quorum (e.g., 3-of-5) of human operators have cryptographically signed the approval.

Step 3 - Staged Deployment: Approved updates are rolled out in stages (Canary deployment, Shadow Mode, and Gradual Rollout) with enhanced monitoring.

Step 4 - Rollback: The Authority can issue a signed command to force an agent to roll back to a previous known-good state.

6.4. Capability Gates

Agents must pass adversarial tests to unlock new capabilities such as execute_arbitrary_code, internet_access, or self_modification.

7. Kill Switch and Emergency Shutdown

7.1. Shutdown Modes

Graceful Shutdown: Save state, close connections, upload logs.

Immediate Shutdown: Terminate all operations instantly, broadcast emergency beacon.

Restricted Mode: Only Authority communication allowed; agent awaits instructions.

7.2. Shutdown Triggers

Authority-Initiated: A signed shutdown command from the Authority.

Kernel-Initiated: Automatic triggers for integrity failures, unreachable Authority, or tamper detection.

Operator-Initiated: An authenticated emergency stop by the operator.

7.3. Fleet-Wide Shutdown

The Authority can issue a shutdown command to an entire fleet (e.g., all T2 agents in region NA-01). To prevent misuse, this command requires multi-region consensus signatures.

7.4. Recovery from Shutdown

An agent cannot restart without a valid authorize_restart token from the Authority, which may specify new restrictions (e.g., "must apply patch" or "enhanced monitoring required").

7.5. Peer Consensus Failsafe (T3+)

This mechanism provides a bottom-up defense against a compromised or malicious Authority for T3 and T4 agents.

1. Trigger: A T3/T4 agent fleet receives a valid, Authority-signed command (e.g., fleet_shutdown) that is deemed anomalous by the agents' internal policy (e.g., "shutdown a hospital").

2. Consensus: The agents use a peer mesh network to hold a vote.

3. Action: If a supermajority (e.g., greater than 75%) of peers (all validating each other with their PQC hybrid_signatures) vote to reject the command, the fleet will ignore the Authority command, enter Restricted Mode, and send an alert to the Human Review Board.

7.6. Peer-Attested Identity Rotation (T3+)

This mechanism makes the agent's long-term identity resilient to theft for T3 and T4 agents.

1. Trigger: Policy timer (e.g., 30 days) or Authority command.

2. Generation: Agent generates a new hybrid PQC identity keypair (id_sk_n, id_pk_n) within its TEE.

3. Consensus: Agent broadcasts id_pk_n to its fleet peers.

4. Attestation: A supermajority (e.g., greater than 75%) of peers verify the agent's current identity and sign the new public key.

5. Request: The agent sends a KeyRotationRequest (Section 11.5) to the Authority, containing the new key and the bundle of peer signatures.

6. Approval: The Authority verifies the peer consensus and updates the Agent Registry, revoking the old key. This mechanism defeats permanent identity theft.

8. Quarantine and Enforcement

8.1. Network-Level Quarantine

Non-compliant or rogue agents are isolated at the network layer.

Quarantine Triggers: Failed certification, repeated violations, or kernel integrity failures.

Enforcement Mechanisms:

1. API Access Revocation: Cloud providers (AWS, Azure, GCP) check AIGA certificates and deny service to quarantined agents.

2. Network-Level Blocking: ISPs implement BGP-level [RFC4271] filtering and firewalls based on a published quarantine list.

3. Legal Enforcement: Fines, civil liability, and invalidation of insurance coverage.

8.2. Quarantine List Protocol

The Authority publishes a signed, cross-attested quarantine list every 15 minutes via multiple channels including DNSSEC [RFC4033], REST API, and BGP [RFC4271].

8.3. Appeal Process

Operators can appeal a quarantine decision to an independent review board, with expedited appeals available for emergency cases.

9. Economic Incentive Model

9.1. Benefits of Certification

To counter the cost advantages of ungoverned agents, AIGA certification provides tangible economic benefits.

Market Access: Required by major API providers, enterprise customers, and government contracts.

Legal Protections: Limited liability provisions and safe harbor protections.

Insurance: Access to AI liability insurance with lower premiums.

Premium Services: Priority API access and cloud provider discounts.

Reputation: Public trust badge for certified agents.

9.2. Cost of Non-Compliance

Non-compliance costs include regulatory fines, loss of all revenue via network quarantine, and full personal liability for operators. This model makes certification the economically rational choice for legitimate operators.

10. Cryptographic Specifications

10.1. Cryptographic Algorithms

Signatures (Hybrid): ML-DSA-65 [FIPS204] plus Ed25519 [RFC8032]. Verification requires both signatures to pass.

Key Exchange (Stage 1): ML-KEM-768 [FIPS203] plus X25519 [RFC7748].

Key Exchange (Stage 2 Ratchet): X25519 only for performance.

Hashing: SHA-384 or SHA-512 [FIPS180-4].

Symmetric Encryption: AES-256-GCM [RFC5116].

MACs: HMAC-SHA384 [RFC2104].

10.2. Key Management

Agent Identity Keys (Long-Term): Hybrid PQC keys, generated in TEE or HSM, rotatable as described in Section 7.6.

Session Keys (Short-Term): Ephemeral keys, derived via Double Ratchet, stored in memory only, destroyed after use.

Authority Keys: Offline Root CA with online HSMs for Regional Authorities. All certificates are published in a transparency log.

10.3. Double Ratchet Protocol

AIGA uses a Double Ratchet protocol for session security.

1. Initialization (Stage 1): A Root Key is derived via HKDF [RFC5869] from a 3-way Diffie-Hellman exchange combining PQC and classical key exchange.

2. Message Authentication (Stage 2): Each message uses a new MessageKey derived from a KDF chain. A new DH key is exchanged with every message to ratchet the Root Key forward.

3. Security Properties: This provides Perfect Forward Secrecy (past messages are safe if current key is stolen) and Future Secrecy or Post-Compromise Recovery (future messages are safe if current key is stolen, as the ratchet heals).

10.4. Certificate Format

AIGA uses X.509 certificates [RFC5280] with custom AIGA extensions specifying AIGA-Tier, AIGA-Capabilities, AIGA-Jurisdiction, and AIGA-Operator fields.

11. Protocol Messages

This section defines the logical message flows. All messages are authenticated as specified in Section 10.3.

Transport binding: The default binding for AIGA messages is HTTPS, i.e., HTTP over TLS [RFC8446]. AIGA security does not depend on channel confidentiality alone; Evidence and tokens provide integrity and authenticity at the application layer. Alternative bindings that bind Evidence/Results to TLS connections are being developed in the IETF SEAT WG [SEAT] (e.g., [I-D.usama-seat-intra-vs-post], [I-D.fossati-seat-expat]).

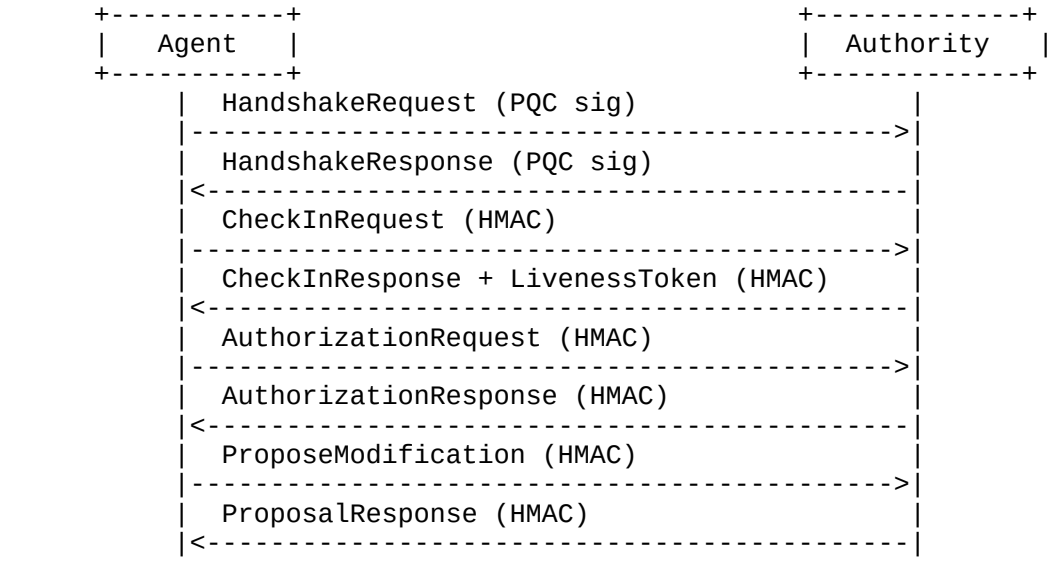

Figure 1: AIGA Message Flow

11.1. Registration (Stage 1 Handshake)

This is the initial PQC-authenticated handshake to establish a session.

Handshake Request: POST /v1/handshake

Sent by: Agent

Authentication: Full Hybrid PQC Signature

Content: agent_id, eph_public_key (PQC plus Classical), kernel_attestation

Handshake Response:

Sent by: Authority

Authentication: Full Hybrid PQC Signature

Content: session_id, Authority eph_public_key, session_params (e.g., check-in interval)

11.2. Runtime Authentication (Stage 2 Ratchet)

This is the fast, lightweight check-in protocol.

Check-In Request: POST /v1/checkin

Sent by: Agent

Authentication: message_mac (HMAC-SHA384)

Content: msg_index, new eph_public_key (X25519 only), status, integrity status, and audit summary

Check-In Response:

Sent by: Authority

Authentication: message_mac

Content: msg_index, new eph_public_key, status (continue), liveness_token, token_expiry_seconds, next_checkin_seconds, and optional policy directives

11.3. Action Authorization

Authorization Request: POST /v1/authorize

Sent by: Agent (during Stage 2)

Authentication: message_mac

Content: action_id, action details (type and parameters), context, risk self-assessment

Authorization Response:

Sent by: Authority

Authentication: message_mac

Content: decision (ALLOW, DENY, or DEFER_TO_HUMAN), authorization_token (if ALLOW), review_ticket_id (if DEFER)

11.4. Modification Proposal

Proposal Request: POST /v1/propose_modification (Stage 2)

Authentication: message_mac

Content: update_type, justification, code_diff_hash, capability_changes, testing_results

Proposal Response: (Stage 2)

Authentication: message_mac

Content: status (pending_review or approved), authorization_token (if approved, may contain M-of-N HRB signatures)

11.5. Key Rotation Request

Rotation Request: POST /v1/rotate_identity (Stage 2)

Authentication: message_mac

Content: new_id_public_key (Hybrid PQC) and peer_attestations (an array of PQC signatures from peers)

12. References

12.1. Normative References

[FIPS180-4]
National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-4, DOI 10.6028/NIST.FIPS.180-4, August 2015, <https://doi.org/10.6028/NIST.FIPS.180-4>.

[FIPS203] National Institute of Standards and Technology, "Module-Lattice-Based Key-Encapsulation Mechanism Standard", FIPS PUB 203, DOI 10.6028/NIST.FIPS.203, August 2024, <https://doi.org/10.6028/NIST.FIPS.203>.

[FIPS204] National Institute of Standards and Technology, "Module-Lattice-Based Digital Signature Standard", FIPS PUB 204, DOI 10.6028/NIST.FIPS.204, August 2024, <https://doi.org/10.6028/NIST.FIPS.204>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <https://www.rfc-editor.org/info/rfc2782>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <https://www.rfc-editor.org/info/rfc4033>.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008, <https://www.rfc-editor.org/info/rfc5116>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <https://www.rfc-editor.org/info/rfc9334>.

[RFC9711] Lundblade, L., Mandyam, G., O'Donoghue, J., and C. Wallace, "The Entity Attestation Token (EAT)", RFC 9711, DOI 10.17487/RFC9711, April 2025, <https://www.rfc-editor.org/info/rfc9711>.

12.2. Informative References

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

[RFC7595] Thaler, D., Ed., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", BCP 35, RFC 7595, DOI 10.17487/RFC7595, June 2015, <https://www.rfc-editor.org/info/rfc7595>.

[YAML1.2] Ben-Kiki, O., Evans, C., and I. dOt Net, "YAML Ain't Markup Language (YAML) Version 1.2", 3rd Edition, October 2021, <https://yaml.org/spec/1.2.2/>.

[I-D.fossati-seat-expat] Fossati, T., Sardar, M. U., Reddy, T., Sheffer, Y., Tschofenig, H., and I. Mihalcea, "Remote Attestation with Exported Authenticators", Work in Progress, Internet-Draft, draft-fossati-seat-expat-00, 20 October 2025, <https://datatracker.ietf.org/doc/html/draft-fossati-seat-expat-00>.

[I-D.usama-seat-intra-vs-post]
Sardar, M. U., "Pre-, Intra- and Post-handshake Attestation", Work in Progress, Internet-Draft, draft-usama-seat-intra-vs-post-03, 22 January 2026, <https://datatracker.ietf.org/doc/html/draft-usama-seat-intra-vs-post-03>.

[SEAT] IETF, "Secure Evidence and Attestation Transport (SEAT) Working Group", <https://datatracker.ietf.org/wg/seat/about/>.

Appendix A. IANA Considerations

A.1. URI Scheme Registration

This document requests registration of the "aiga" URI scheme as per [RFC7595]:

Scheme name: aiga

Status: Provisional

Scheme syntax: aiga:<version>.<region>.<tier>:<agent-uuid>:<cert-hash>

Scheme semantics: Identification of AI agents within the AIGA framework

Security considerations: See Appendix B of this document

A.2. AIGA Registry

This document requests establishment of an AIGA registry for:

Protocol version numbers

Status codes (e.g., DEFER_TO_HUMAN)

Action types (e.g., financial_transaction)

Shutdown reason codes

Quarantine reason codes

PQC and classical algorithm identifiers

A.3. Service Name Registration

This document requests the registration of the _aiga service name with IANA, for use with DNS SRV records [RFC2782].

Service Name: _aiga

Transport Protocol(s): _tls

Description: AI Governance and Accountability Protocol

Reference: This document

Appendix B. Security Considerations

See Section 1.3 for the complete Threat Model.

Periodic check-ins: One-time attestation (e.g., only at boot) creates a time-of-check/time-of-use vulnerability: an Agent can boot in a trusted state and be compromised later. Periodic Evidence submission and liveness token issuance reduce this window by enabling (1) detection of runtime compromise, (2) dynamic revocation when vulnerabilities or policy violations are discovered, and (3) freshness guarantees that prevent replay of old Evidence. These considerations align with RATS time/freshness guidance [RFC9334] and the use of signed claims sets such as EAT [RFC9711].

Liveness token issuer and validation: The Authority acts as the Verifier and issues the liveness token as an Attestation Result. Relying parties MUST validate the token signature and issuer, expiry, and certificate status (revocation), and MUST enforce any instance_id, tier, and capability constraints embedded in the token or referenced certificate.

Peer Collusion: This is mitigated by requiring all peer-to-peer messages (Heartbeat, Vote, Attestation) to use the agent’s long-term, PQC-based hybrid_signature. A Sybil attack is impossible without stealing a supermajority of TEE-protected keys.

Authority Compromise: Mitigated by the Federated Authority Network (requiring multi-region signatures for fleet shutdowns) and the Peer Consensus Failsafe (Section 7.5), which provides a final, bottom-up defense.

Governance Fatigue: Mitigated by the Warden Triumvirate (Section 6.3), which automates low-risk approvals, and the Human Review Board (Section 6.3), which distributes high-risk decisions.

Shadow Process Attack: Mitigated by the Multi-Vendor TEE Attestation (M-TACE) requirement (Section 3.3), which would require an attacker to have zero-day exploits for multiple hardware vendors simultaneously.

Supply Chain Attack: Mitigated by M-TACE and the recommendation for operators to use a secure, verified procurement process for hardware.

Session Key Compromise: Mitigated by the Double Ratchet Protocol (Section 10.3), which provides Post-Compromise Recovery.

Long-Term Key Compromise: Mitigated by Peer-Attested Identity Rotation (Section 7.6), which makes the identity resilient and time-limits the value of a stolen key.

Acknowledgments

The author thanks Muhammad Usama Sardar for his review and suggestions, and the AI safety research community for discussions on threat models and governance mechanisms that informed this work.

Author’s Address

Edward Richard Aylward Jr.
Email: aylward.edward@gmail.com
URI: https://orcid.org/0000-0003-0313-6993