Independent Submission
Internet-Draft
Intended status: Standards Track
Expires: 12 October 2026

K. Nyantakyi
Vorim AI

April 2026

Vorim Agent Identity Protocol (VAIP)
draft-nyantakyi-vaip-agent-identity-01

Abstract

This document defines the Vorim Agent Identity Protocol (VAIP), a standard for establishing verifiable cryptographic identity, fine-grained permissions, trust scoring, and tamper-evident audit trails for autonomous AI agents.

As AI agents increasingly perform actions autonomously in enterprise systems, existing identity protocols designed for human users or static services are insufficient. VAIP addresses this gap by providing cryptographic primitives and data structures for issuing, verifying, and managing agent identities in multi-tenant environments.

The protocol uses Ed25519 for agent identity, SHA-256 for audit integrity, and defines seven hierarchical permission scopes with time-bounded grants and rate limiting.

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/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 3 October 2026.

Copyright Notice

Table of Contents

1. Introduction

1.1. Problem Statement

Autonomous AI agents are increasingly deployed to perform real-world tasks: executing code, sending messages, making financial transactions, and accessing sensitive resources. Unlike human users, agents lack a standardised identity layer. There is no widely adopted protocol for answering fundamental questions about an agent:

* Who is this agent? (Identity)

* What is it allowed to do? (Permissions)

* What has it done? (Audit)

* Should I trust it? (Trust)

Existing identity systems (OAuth 2.0, API keys, X.509 certificates) were designed for human users or static services. They do not address the unique requirements of autonomous agents that act independently, accumulate behavioural history, and require dynamic trust assessment.

Non-human identities outnumber human identities by approximately 50:1 in the average enterprise, yet no standardised infrastructure exists for managing them.

1.2. Design Principles

1. Cryptographic verifiability: Every identity claim MUST be verifiable through public key cryptography without relying on a central authority at verification time.

2. Minimal trust: Private keys are generated once and never stored by the issuing system. The agent or its operator is the sole custodian of its private key.

3. Auditability: Every action performed by an agent MUST be logged in a tamper-evident ledger. Audit records MUST support independent integrity verification.

4. Interoperability: The protocol uses widely supported cryptographic standards (Ed25519, SHA-256) and data formats (JSON, PEM) to enable cross-platform verification.

5. Multi-tenancy: All identities, permissions, and audit records are scoped to an organisation. No data leaks across tenants.

1.3. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] when, and only when, they appear in all capitals, as shown here.

Agent An autonomous software entity that performs actions on behalf of an organisation.

Organisation A tenant entity that owns and manages agents.

Operator A human user who registers and configures agents within an organisation.

Scope A named permission that authorises a specific category of action.

Trust Score A numeric assessment (0-100) of an agent’s reliability based on behavioural history.

Audit Event An immutable record of an action performed by or on behalf of an agent.

2. Agent Identity

2.1. Identity Format

Every agent MUST be assigned a unique agent identifier upon registration. The identifier follows this format:

agid_{org_slug}_{random}

Where:

* "agid_" is a fixed prefix identifying the string as a VAIP agent identifier

* {org_slug} is a truncated (up to 8 characters) alphanumeric slug identifying the owning organisation

* {random} is a cryptographically random string (8 characters, derived from UUID v4)

Example: agid_acme_a1b2c3d4

Agent identifiers MUST be globally unique and MUST NOT be reused after revocation.

2.2. Keypair Generation

Each agent MUST be assigned an Ed25519 [RFC8032] keypair at registration time. Ed25519 was chosen for the following properties:

* 128-bit security level

* Fast key generation, signing, and verification

* Small key and signature sizes (32-byte public keys, 64-byte signatures)

* Deterministic signatures (no nonce reuse vulnerabilities)

* Resistance to timing attacks

The keypair MUST be generated as follows:

1. Generate an Ed25519 keypair using the system’s cryptographic random number generator

2. Encode the public key in SPKI/PEM format

3. Encode the private key in PKCS#8/PEM format

4. Compute the key fingerprint

5. Return the private key to the operator exactly once

6. Store the public key and fingerprint; never store the private key

2.3. Key Fingerprint

A key fingerprint is a SHA-256 [RFC6234] hash of the PEM-encoded public key, represented as a hexadecimal string (64 characters):

fingerprint = SHA-256(public_key_pem)[0:64]

The fingerprint serves as a compact, verifiable reference to an agent's identity. It MUST be included in trust verification responses and MAY be used for identity comparison without transmitting the full public key.

2.4. Agent Lifecycle

An agent MUST exist in exactly one of the following states at any time:

* pending: Registered but not yet activated
* active: Fully operational, permissions enforceable
* suspended: Temporarily disabled, can be reactivated
* revoked: Permanently disabled, cannot be reactivated
* expired: Past its expiration timestamp

A revoked agent MUST NOT be reactivated. Its identifier MUST NOT be reused.

2.5. Key Rotation

An agent's keypair MAY be rotated by revoking the current agent and registering a new agent with the same metadata. The new agent MUST receive a new agent_id and keypair.

3. Permission Model

3.1. Scopes

VAIP defines seven permission scopes. Each scope authorises a category of agent actions:

Scope	Identifier	Risk Level
Read	agent:read	Low
Write	agent:write	Medium
Execute	agent:execute	Medium
Transact	agent:transact	High
Communicate	agent:communicate	Medium
Delegate	agent:delegate	High
Elevate	agent:elevate	Critical

Table 1

Scopes are not hierarchical by default. Possessing agent:write does not imply agent:read. Each scope MUST be granted independently.

3.2. Permission Grants

A permission grant binds a scope to an agent with optional constraints including:

* valid_from: When the grant becomes effective
* valid_until: Optional expiration time
* rate_limit: Maximum uses within a time window
* conditions: Additional constraints (e.g., IP allowlists)

3.3. Rate Limits

A rate limit constrains how frequently a scope can be exercised:

```
{ "max": 1000, "window": "1h" }
```

Supported windows: "1m" (minute), "1h" (hour), "1d" (day).

When an agent exceeds its rate limit, permission checks MUST return denied with reason RATE_LIMIT_EXCEEDED.

3.4. Permission Check Flow

When an agent requests to perform an action, the system MUST:

1. Validate agent_id exists and status is active
2. Check cache for recent permission decision
3. If cache miss: query grants, filter by validity and revocation, evaluate rate limits and conditions
4. Cache the decision (RECOMMENDED TTL: 300 seconds)
5. Return the permission check result

Valid denial reasons: AGENT_NOT_FOUND, AGENT_NOT_ACTIVE, SCOPE_NOT_GRANTED, GRANT_EXPIRED, RATE_LIMIT_EXCEEDED, CONDITION_NOT_MET.

4. Audit Trail

4.1. Event Schema

Every auditable action MUST produce an audit event including:

* event_id: Unique, time-sortable identifier (e.g., ULID)
* agent_id: Agent that performed the action
* org_id: Organisation context
* event_type: Category (tool_call, api_request, message_sent, permission_change, status_change, key_rotation)
* action: Specific action performed
* result: Outcome (success, denied, error)
* timestamp: ISO 8601 timestamp

Optional fields include resource, input_hash, output_hash, permission, latency_ms, error_code, signature, and metadata.

4.2. Hash Format

All hashes in audit events MUST use the format:

sha256:{hex_digest}

Where {hex_digest} is the lowercase hexadecimal representation of the SHA-256 hash.

4.3. Event Signatures

Audit events MAY be signed by the agent that produced them. When present, the signature field MUST contain an Ed25519 signature of the canonical JSON [RFC8259] representation of the event (excluding the signature field itself), encoded as:

ed25519:{base64_signature}

4.4. Immutability

Audit events MUST be append-only. Once written, an event MUST NOT be modified or deleted except by automated retention policies.

4.5. Signed Audit Bundles

Conforming systems MUST support exporting audit events as signed bundles. A bundle contains the events array, event count, time range, organisation identifier, and a manifest field containing a SHA-256 hash of the JSON-serialised events array.

Any modification to any event produces a different manifest hash, making tampering detectable.

5. Trust Scoring

5.1. Overview

Trust scoring provides a quantitative assessment of an agent’s reliability. The score is a number from 0 to 100, computed from multiple behavioural and configuration factors. Trust scores are dynamic, public, and composite.

5.2. Scoring Algorithm

The trust score is computed as:

```
score = base_score + status_factor + age_factor
        + success_rate_factor - denial_penalty
        - scope_breadth_penalty
```

Base score: 50. Status factor: active (+10), suspended (-20), revoked (score fixed at 0). Age factor: >90 days (+15), >30 days (+10), >7 days (+5). Success rate factor: round(success_rate * 15) over trailing 30-day window. Denial penalty: -10 if denied_count exceeds 10% of total events. Scope breadth penalty: -5 if agent holds more than 5 active scopes. Final score clamped to [0, 100].

5.3. Trust Verification

Conforming implementations MUST provide a public, unauthenticated endpoint for trust verification. The response includes agent_id, verification status, trust score, active scopes, key fingerprint, and revocation status.

5.4. Trust Badges

Conforming implementations SHOULD provide embeddable SVG trust badges displaying protocol name, verification status, and trust score with colour coding: green (80-100), amber (50-79), red (0-49).

6. Conformance Levels

Implementations MAY conform to one or more levels:

Level 1 - Identity Agent registration with Ed25519 keypairs, key fingerprints, lifecycle management.

Level 2 - Permissions Level 1 plus seven permission scopes, time-bounded grants, rate limiting, cached permission checks.

Level 3 - Audit Level 2 plus audit event ingestion, schema compliance, SHA-256 hashing, event query with filtering.

Level 4 - Trust Level 3 plus trust score computation, public verification endpoint, embeddable trust badges.

Level 5 - Full Conformance Level 4 plus signed audit bundle export, Ed25519 event signatures, multi-format export, API key management.

7. Credential Delegation (Future Extension)

This section outlines a future extension to VAIP for secure credential delegation across multi-agent systems. This work is complementary to draft-sweeney-wimse-credential-delegation-00, which defines the wire protocol for OAuth token delegation. VAIP's contribution is binding credential delegation to verified agent identities, permission scopes, and audit trails.

7.1. Overview

When an agent needs to access a third-party service (e.g., Google Drive, GitHub, Salesforce) on behalf of a user, it requires OAuth credentials. Credential delegation defines how an agent receives scoped, time-limited credentials without exposing raw refresh tokens or long-lived secrets.

A conforming credential delegation system SHOULD:

* Verify the requesting agent's VAIP identity before issuing any credential
* Check the agent's permission scope (agent:read, agent:execute, etc.) against the requested OAuth scope
* Issue short-lived access tokens only; refresh tokens MUST remain in the delegation server's encrypted vault
* Record every credential issuance as an audit event in the agent's VAIP audit trail
* Support immediate revocation that propagates across delegation chains

7.2. Multi-Hop Delegation Chains

In multi-agent systems, credentials may flow through delegation chains: User to Agent A to Agent B to Agent C. At each hop, credentials SHOULD be attenuated — narrowing the scope, reducing the time-to-live, and lowering the rate limit.

Each delegation hop MUST:

* Verify the delegating agent has the agent:delegate permission in VAIP
* Produce a signed delegation receipt (Ed25519 JWS) linking the delegator's agent_id to the delegate's agent_id
* Record the delegation as an audit event with event_type "credential_delegation"
* Enforce that the delegated credential's scope is a subset of the delegator's scope

7.3. Ephemeral Agent Identity

For short-lived agents that do not require persistent identity registration, a future extension MAY support ephemeral identity using W3C did:key format. Ephemeral agents would:

* Bootstrap identity on instantiation without pre-registration
* Receive a temporary VAIP agent_id linked to the did:key
* Have a maximum lifetime (RECOMMENDED: 24 hours)
* Still produce audit events attributable to the ephemeral identity
* Be subject to all VAIP permission checks

Ephemeral identity is OPTIONAL and does not replace persistent Ed25519 identity for production agents requiring long-term accountability.

7.4. Integration with Credential Delegation Protocols

VAIP is designed to be complementary to credential delegation protocols such as draft-sweeney-wimse-credential-delegation-00. The integration model is:

* VAIP provides the agent identity layer (who is this agent?)
* VAIP provides the permission layer (what is it allowed to do?)
* VAIP provides the audit layer (what did it do?)
* A credential delegation protocol provides the credential flow (how does the agent safely access third-party services?)

Together, these two layers cover the full agent security surface: identity, permissions, credentials, audit, and trust.

8. Security Considerations

8.1. Cryptographic Choices

Primitive	Algorithm	Rationale
Agent identity	Ed25519	Fast, compact, timing-attack resistant

Fingerprints	SHA-256	Widely supported, collision resistant
Audit integrity	SHA-256	Industry standard for data integrity
Password hashing	bcrypt (cost 12)	Memory-hard, adjustable work factor
API key storage	SHA-256	One-way hashing for stored secrets

Table 2

8.2. Private Key Handling

Conforming implementations:

* MUST generate private keys server-side using a cryptographic random number generator

* MUST return the private key to the operator exactly once

* MUST NOT store, log, or persist the private key after returning it

* MUST NOT transmit private keys over unencrypted channels

8.3. Multi-Tenant Isolation

All database queries MUST be scoped to the authenticated organisation's identifier. No agent, permission, or audit record from one organisation may be accessible by another.

8.4. Transport Security

All API communications MUST use TLS 1.2 or higher. TLS 1.3 is RECOMMENDED.

8.5. Threat Model

Threat	Mitigation
Key compromise	Immediate revocation, new registration
Audit tampering	SHA-256 manifest, append-only storage

Cross-tenant access	Organisation-scoped queries
Replay attacks	Time-sortable event IDs, timestamp validation
Brute-force auth	bcrypt hashing, API key entropy (192 bits)
Trust manipulation	Score from audited behaviour, not self-reported

Table 3

9. IANA Considerations

This document requests the registration of the following media type:

Type name application

Subtype name vaip+json

Required parameters none

Optional parameters none

Encoding considerations binary (UTF-8 JSON)

Security considerations See the Security Considerations section of this document.

Interoperability considerations Implementations MUST support JSON encoding as defined in RFC 8259.

Published specification This document.

Applications that use this media type AI agent identity management systems, audit trail systems, trust verification services.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

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

[RFC6234] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[RFC8259] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

10.2. Informative References

[RFC6749] Hardt, D., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[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, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[I-D.sweeney-wimse-credential-delegation] Sweeney, D., "Credential Delegation Protocol for AI Agents", Work in Progress, Internet-Draft, draft-sweeney-wimse-credential-delegation-00, 2026, <https://datatracker.ietf.org/doc/html/draft-sweeney-wimse-credential-delegation-00>.

Appendix A. Reference Implementation

A reference implementation of VAIP is available at:

* TypeScript SDK: https://www.npmjs.com/package/@vorim/sdk
* Python SDK: https://pypi.org/project/vorim/
* Protocol specification: https://github.com/Kzino/vorim-protocol
* MCP Server: https://www.npmjs.com/package/@vorim/mcp-server

Appendix B. Relationship to Existing Standards

VAIP is designed to complement, not replace, existing identity and security standards:

OAuth 2.0 (RFC 6749) OAuth focuses on delegated authorisation for human users. VAIP extends the concept to autonomous agents that act independently without human-in-the-loop authorisation.

W3C Decentralized Identifiers (DID) VAIP agent identifiers could be extended to DID format for decentralised resolution. The current format prioritises simplicity and readability.

SPIFFE SPIFFE provides workload identity attestation. VAIP adds trust scoring, audit trails, and permission models beyond identity.

X.509 (RFC 5280) VAIP uses Ed25519 directly rather than certificate chains, prioritising simplicity and agent-specific semantics over PKI hierarchies.

Credential Delegation Protocol (draft-sweeney-wimse-credential-delegation-00) The Credential Delegation Protocol defines secure OAuth token delegation across multi-agent systems. VAIP is complementary: VAIP provides the identity, permission, and audit layers while the Credential Delegation Protocol provides the credential flow mechanics. Together they cover the full agent security surface. See Section 8 of this document for the integration model.

WIMSE (Workload Identity in Multi-System Environments) The IETF WIMSE working group addresses non-human identity in distributed systems. VAIP's agent identity model is a specific application of workload identity to autonomous AI agents, with the addition of trust scoring, behavioural audit trails, and AI-specific permission scopes.

Appendix C. Regulatory Alignment

VAIP's identity, permission, and audit features support compliance with emerging AI governance frameworks including:

* EU AI Act (enforced August 2025): Traceability and audit trail requirements for high-risk AI systems.

* US Executive Order on AI (EO 14110): Risk management, transparency, and accountability for AI systems.

* Colorado AI Act: Automated decision-making documentation requirements.

* NIST AI Risk Management Framework: AI system governance and accountability controls.

Appendix D. Acknowledgements

The author thanks the open-source community and early adopters of the Vorim Agent Identity Protocol for their feedback and contributions to this specification.

Author's Address

Kwame Nyantakyi
Vorim AI
Email: kwame@vorim.ai
URI: https://vorim.ai