Network Working Group
Internet-Draft
Intended status: Informational
Expires: 20 October 2026

S. Jovancevic
SKGO, IKT Support

20 April 2026

Verifiable Identity Claims and Delegation Model (VICDM)
draft-jovancevic-vicdm-04

Status of This Memo

Copyright Notice

Abstract

This document defines a conceptual framework for handling identity assertions in application-layer protocols. It introduces a model in which identity on the Internet is optional, but any asserted identity MUST be verifiable.

It further defines a delegation mechanism that allows entities to authorize third-party infrastructure to act on their behalf in a verifiable and transparent manner.

The goal is to reduce identity misrepresentation while fully preserving the ability for anonymous and pseudonymous interaction. This document does not define a protocol; it defines the principles that protocol specifications SHOULD follow when addressing agent identity.

A concrete protocol implementation of these principles is defined in [SAIP].

Table of Contents

1. Introduction

The Internet is built on a principle of openness: entities may communicate without being required to identify themselves. This property is fundamental and MUST be preserved.

However, modern internet systems increasingly rely on identity assertions for access control, prioritization, rate limiting, and trust evaluation. Automated agents — AI crawlers, IoT devices, backup systems, enterprise automation — routinely assert identities to influence how servers treat their requests.

A critical problem arises when these identity assertions cannot be verified. An agent that claims to be "GoogleBot" or "legitimate-backup-service" without cryptographic proof provides no more assurance than an agent that claims nothing at all. Worse, false identity assertions actively harm the ecosystem by:

- Allowing malicious actors to impersonate trusted services
- Corrupting reputation systems that depend on identity signals
- Undermining filtering mechanisms that would otherwise work
- Creating false trust that delays detection of abuse

The fundamental issue is not the presence or absence of identity claims — it is the verifiability of those claims.

This document defines the VICDM principle:

> Anonymous interaction is permitted.
> Identity assertion is permitted.
> False identity assertion is not.

This seemingly simple principle has significant architectural implications for how protocols handling agent identity should be designed.

A concrete protocol implementing these principles is defined in the Signed Agent Identity Protocol [SAIP].

1.1. Relationship to webbotauth Work

The IETF webbotauth working group is developing mechanisms for bot authentication. Current proposals focus on anonymous attestation models in which a bot proves it is vouched for by a trusted attester without revealing its specific identity.

Anonymous attestation and VICDM are complementary, not competing:

- Anonymous attestation addresses the privacy use case: a bot proves legitimacy without disclosure of its identity.

- VICDM addresses the accountability use case: when a bot DOES

assert an identity, that assertion must be verifiable.

Both models are needed. They serve different threat models and different operational requirements.

2. 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] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Anonymous Client:	A client that does not assert any identity. This is always a valid and permitted mode of interaction.
Claiming Client:	A client that asserts an identity, such as a domain name, organization, or service name.
Verified Client:	A client whose asserted identity has been validated through one or more verifiable mechanisms as defined in Section 5.
Identity Claim:	Any statement or metadata by which a client represents itself as belonging to a specific domain, organization, or service.
Delegated Infrastructure:	Third-party systems that have been explicitly authorized to act on behalf of an entity through verifiable mechanisms.
Attester:	An entity that vouches for the legitimacy of a client without necessarily disclosing the client’s identity. Used in anonymous attestation models.

3. The Core Principle

The VICDM model is founded on a single, minimal principle:

> Systems MUST allow anonymous interaction.
>
> A client asserting an identity MUST provide verifiable linkage to that identity.
>
> Failure to provide verifiable linkage to an asserted identity MUST result in that identity claim being treated as invalid.

This principle is deliberately minimal. It does not require universal identity. It does not prohibit anonymity. It requires only that identity claims, when made, be honest and verifiable.

The distinction between an Anonymous Client and a Claiming Client whose claim is unverifiable is operationally significant:

- An Anonymous Client makes no representation about its origin or affiliation. Servers may apply default anonymous policies.
- An unverified Claiming Client actively misrepresents itself. This is categorically different from anonymity and SHOULD be treated with lower trust than genuine anonymity.

Implementations SHOULD distinguish between these two cases and SHOULD NOT treat unverified identity claims as equivalent to

anonymous interaction.

4. Identity Classes

Under the VICDM model, clients fall into one of four classes:

Class 0 — Anonymous:
The client asserts no identity. No identity claim is present. This is a valid and fully supported mode of interaction. Servers apply default anonymous policies.

Class 1 — Unverifiable Claim:
The client asserts an identity but provides no verifiable linkage. The claim MUST be treated as invalid. Trust SHOULD be lower than for Class 0 (anonymous) clients, as the client is actively misrepresenting itself.

Class 2 — Partially Verified:
The client asserts an identity and provides partial verification (e.g., DNS consistency but no cryptographic proof). Servers MAY grant limited elevated trust based on deployment policy.

Class 3 — Fully Verified:
The client asserts an identity and provides full cryptographic verification of that identity. Servers MAY grant elevated trust, priority handling, or increased rate limits based on deployment policy.

5. Identity Verification

Verification of identity claims MAY use one or more of the following mechanisms. No single mechanism is mandated; systems MAY combine multiple signals to increase confidence.

5.1. DNS-Based Verification

The asserted identity SHOULD be consistent with DNS records for the claimed domain:

- Forward DNS resolution of the claimed domain SHOULD return an address associated with the client’s network.
- Reverse DNS (PTR) resolution of the client’s IP address SHOULD return a hostname within the claimed domain.
- DNS TXT records at a well-known prefix (see Section 6.2) MAY carry cryptographic key material for stronger binding.

DNS-based verification alone provides weak assurance. It SHOULD be combined with cryptographic mechanisms where possible.

5.2. Cryptographic Verification

The client provides a cryptographic signature over a defined canonical string using a private key whose corresponding public key is discoverable and bound to the claimed identity.

This is the strongest form of verification and is RECOMMENDED for all deployments where security is a concern. A concrete implementation is defined in [SAIP].

5.3. Transport-Layer Indicators

Consistency between the identity claim and transport-layer signals (e.g., TLS SNI, TLS client certificate) MAY be used as a supporting signal. These indicators SHOULD NOT be used

as the sole verification mechanism.

5.4. Attester-Based Verification

A trusted third-party Attester MAY vouch for the legitimacy of a client without disclosing the client's specific identity. This is the basis of anonymous attestation models such as those being developed in the IETF webbotauth working group.

Attester-based verification satisfies the VICDM requirement for Class 2 or Class 3 verification only if the Attester itself is verifiable and its delegation is explicit per Section 6.

6. Delegation Model

Large-scale internet deployments commonly use third-party infrastructure — CDN providers, cloud services, proxy networks — to serve content or operate agents on behalf of a domain owner.

This creates an identity delegation problem: the actual client IP address or User-Agent may belong to the infrastructure provider, not the domain owner. Without explicit, verifiable delegation, servers cannot determine whether the infrastructure is legitimately acting on the domain owner's behalf.

6.1. Delegation Requirements

Any infrastructure acting on behalf of an entity and asserting that entity's identity MUST be explicitly authorized through verifiable mechanisms.

Specifically:

- The domain owner MUST publish an explicit authorization for the delegated infrastructure to act on its behalf.
- The authorization MUST be verifiable by any server without requiring prior coordination with the domain owner.
- Absence of verifiable delegation authorization MUST result in the identity claim being treated as invalid (Class 1).
- The delegation authorization SHOULD be bound to specific infrastructure identifiers (IP ranges, ASNs, public keys) rather than being open-ended.

This requirement directly addresses scenarios where large infrastructure providers assert client identities on behalf of multiple customers without explicit per-customer authorization. Such assertions, without verifiable delegation, MUST be treated as invalid identity claims under this model.

6.2. DNS-Based Attestation Discovery

Entities MAY publish delegation authorization using DNS TXT records at the following well-known prefix:

_saip.<domain>.

The record format is:

_saip.<domain>. IN TXT "v=saip1; [parameters]"

The following parameters are defined:

v=saip1 Version indicator. MUST be present.

re=<hostname> Preferred Registration Entity hostname.
MAY be present. Multiple re= parameters
are permitted.

pk=<key> Base64URL-encoded public key for stateless
verification. MAY be present.

asn=<list> Comma-separated list of authorized ASNs for
delegated infrastructure. MAY be present.

ip=<prefix> CIDR prefix of authorized delegated
infrastructure. MAY be present. Multiple
ip= parameters are permitted.

exp=<ts> Unix timestamp after which this record
SHOULD be considered expired. MAY be present.

Example DNS records:

```
; Vendor publishing their SAIP public key
_saip.acme.com. IN TXT "v=saip1; pk=base64urlkey...;
                        re=re1.saip-registry.example"

; Vendor authorizing a CDN provider by ASN
_saip.acme.com. IN TXT "v=saip1; asn=13335,15169;
                        re=re1.saip-registry.example"

; Vendor authorizing specific IP ranges
_saip.acme.com. IN TXT "v=saip1; ip=192.0.2.0/24;
                        ip=2001:db8::/32"
```

DNS record TTL SHOULD be set to a value appropriate for the key rotation policy of the deployment. A TTL of 3600 seconds (1 hour) is RECOMMENDED as a default.

Servers performing DNS-based verification MUST validate DNSSEC signatures where available [RFC4033].

6.3. Cryptographic Delegation Tokens

As an alternative or supplement to DNS-based Attestation Discovery, domain owners MAY issue signed Delegation Tokens to authorized infrastructure:

```
DelegationToken = Sign(MasterKey,
                       infrastructure_id ||
                       valid_from ||
                       valid_until ||
                       scope)
```

Where:

- infrastructure_id identifies the authorized infrastructure (e.g., ASN, IP range, or public key fingerprint)
- valid_from and valid_until define the validity period
- scope limits the actions the infrastructure may take

Infrastructure presenting a valid Delegation Token MAY assert the domain owner’s identity for requests within the token’s scope and validity period.

6.4. DNS-Native Agent Self-Registration (Alternative Trust Model)

This section defines an alternative trust establishment mechanism that replaces the Registration Entity model with the existing DNS

infrastructure. Vendors adopting this model use their DNS zone as the identity registry for their agents, following the same principles established by DKIM [RFC6376].

The DNS-Join trust model rests on three principles: hardware secures the Master Key, DNS publishes it and acts as the trust circuit breaker, and Rolling Keys provide speed and network efficiency.

6.4.1. Model Overview

In traditional agent identity systems, a central registry holds agent public keys. In the DNS-Native model, the vendor's DNS zone serves as the distributed registry. Each agent self-registers by publishing its Master Public Key as a DNS TXT record within the vendor's delegated subdomain.

This is directly analogous to DKIM key publication [RFC6376]:

```
DKIM:      mail._domainkey.example.com  TXT "v=DKIM1; p=<pubkey>"
SAIP-DNS:  agent-id._saip.example.com   TXT "v=saip1; pk=<pubkey>"
```

The vendor delegates the _saip.<domain> subdomain, granting agents the ability to publish and manage their own TXT records within it. The vendor retains control as the DNS zone authority — the ultimate trust circuit breaker.

6.4.2. Key Hierarchy

The DNS-Native model uses a two-tier key structure:

Tier 1 — Master Key (long-term, DNS-published):

- The agent generates a Master keypair on its hardware security module (TPM, HSM, or Secure Enclave).
- The Master Private Key MUST NEVER leave the hardware module.
- The Master Public Key is uploaded once to the DNS TXT record: agent-id._saip.<vendor-domain>. TXT "v=saip1; pk=<MasterPubKey>"
- This record represents the agent's long-term identity.
- The vendor, as DNS zone authority, acts as the trust circuit breaker: deleting this record instantly revokes the agent's identity across the entire ecosystem after DNS TTL expiry.

Tier 2 — Rolling Keys (per-request, locally generated):

- For each request, the agent generates a fresh ephemeral keypair (RollingKeypair).
- The agent signs the Rolling Public Key with its Master Private Key, producing a Rolling Key Certificate (rcert):

```
rcert = Sign(MasterPrivateKey,
             RollingPublicKey ||
             instanceID       ||
             ts               ||
             nonce            ||
             method           ||
             path)
```

- The nonce, ts, method, and path fields bind the rcert to exactly one request, making it non-replayable by design.
- The Rolling Private Key signs the canonical request string.
- Both the Rolling Public Key (rpk=) and rcert= are transmitted in the SAIP header.

6.4.3. Verification Flow

Server verification requires a DNS lookup only once per agent, after which the Master Public Key MAY be cached per DNS TTL:

1. Server receives SAIP header containing rpk= and rcert=.

2. On first encounter with this agent id=, the server performs a DNS TXT lookup:
   agent-id._saip.<vendor-domain>.
   and retrieves the Master Public Key. This result SHOULD be cached according to the record TTL.

3. Server verifies rcert= using the cached Master Public Key. This confirms that the Rolling Public Key was authorized by the legitimate Master Key holder for this exact request.

4. Server verifies the request signature using rpk=. This confirms the request was made by the holder of the Rolling Private Key.

5. Both verifications MUST succeed for the request to be accepted.

DNS is consulted once per agent (or per TTL expiry), while per-request verification is entirely local — providing both strong security and high performance.

6.4.4. Trust Circuit Breaker

Because the Master Public Key is published in DNS under the vendor's authoritative zone, the vendor retains ultimate control:

- To revoke a single agent: delete its _saip TXT record. After DNS TTL expiry, servers will reject its rcert= values as they can no longer verify them against a Master Public Key.

- To revoke all agents: rotate the vendor's zone signing key or remove the _saip subdomain delegation entirely.

- DNS TTL for _saip records SHOULD be set to 300 seconds (5 minutes) to minimize the revocation window while maintaining reasonable DNS query load.

- Servers MUST NOT cache Master Public Keys beyond the DNS TTL of the record from which they were obtained.

6.4.5. Security Properties

Threat	Mitigation
Stolen rcert	Per-request nonce+ts+method+path binding — structurally non-replayable
Stolen rpk+rcert	Valid for exactly one request only
Master Key theft	Hardware module (TPM/HSM) — never leaves hardware
DNS poisoning	DNSSEC + short TTL (300s)
Mass compromise	DNS record deletion — all instances revoked after TTL expiry

This model provides stronger replay protection than TLS Session

Tickets [RFC5077] because rcert= is bound to a single request rather than a time window, while retaining equivalent performance characteristics through DNS caching.

6.4.6. Comparison with RE-Based Model

Property	RKDF + RE Model	DNS-Native Model
Key registry	RE infrastructure	DNS TXT records
Revocation	RE propagation (within 300s)	DNS deletion + TTL (default 300s)
Per-request key	HMAC derivation (deterministic)	Fresh keypair + MasterKey rcert
Replay protection	Nonce + sequence	Nonce bound in rcert
Infrastructure	RE ecosystem	Existing DNS only
Best suited for	Enterprise, regulated envs	Decentralized, IoT, small vendors, KISS

Both models implement VICDM principles. Implementations MAY support both simultaneously. Servers SHOULD apply equivalent trust to successfully verified agents regardless of model used.

6.4.7. MAC-Based Identity Binding (Hardware Identity Layer)

This section defines an optional extension to the DNS-Native model that adds a hardware identity layer through MAC address binding. When present, this creates a verifiable triplet:

```
   MAC address    (hardware identity layer)
 + IP / ASN       (network identity layer)
 + MasterPublicKey (cryptographic identity layer)
 = Complete Agent Identity
```

All three components MUST be consistent for an agent to be classified as Class 3 (Fully Verified) under this extension. Inconsistency in any component MUST result in downgrade to Class 1 (Unverifiable Claim).

6.4.7.1. The Bot as Identity Broker

Because MAC addresses operate at Layer 2 and are not visible beyond the local network segment, direct MAC verification by remote servers is not possible. Instead, the agent acts as a local identity broker:

The agent:
1. Reads the MAC address from its local network interface (Real MAC), OR derives a Virtual MAC (V-MAC) from its MasterPublicKey (see Section 6.4.7.3).
2. Signs the MAC address using its MasterPrivateKey, producing a MAC Attestation Proof (mac_proof).
3. Includes mac= and mac_proof= in the SAIP header.

The server:
1. Retrieves the MasterPublicKey from DNS.
2. Verifies mac_proof using MasterPublicKey.
   This proves the MAC was attested by the legitimate MasterKey holder — not merely asserted.
3. Applies identity binding policy.

The MasterPublicKey in DNS is the root of trust that validates everything — both Real MAC and V-MAC attestations. The DNS record is the single source of truth; the bot is the cryptographic bridge between local hardware and remote verification.

6.4.7.2. Real MAC — Physical Hardware Binding

A Real MAC is the hardware MAC address of the agent's network interface, as reported by the operating system:

```
Linux:   /sys/class/net/<interface>/address
Windows: Get-NetAdapter | Select-Object MacAddress
macOS:   ifconfig en0 | grep ether
```

Real MAC binding is RECOMMENDED for:

- IoT devices with fixed network interfaces
- Physical servers with stable hardware
- High-assurance deployments where hardware binding is a compliance requirement

The MAC Attestation Proof for a Real MAC:

```
mac_proof = Sign(MasterPrivateKey,
                 "real"    ||
                 mac_bytes ||
                 ts        ||
                 nonce)
```

Where mac_bytes is the 6-byte IEEE 802 MAC address in canonical form (big-endian, no separators).

6.4.7.3. Virtual MAC (V-MAC) — Cryptographic Identity

A V-MAC is derived mathematically from the MasterPublicKey, providing a stable, unique, human-readable identifier for virtual agents, cloud instances, and software bots that do not have a meaningful physical MAC address.

V-MAC derivation:

```
raw     = SHA256(MasterPublicKey)[0:6]   (first 6 bytes)
v_mac   = raw with bit 1 of byte 0 set to 1 (LAA bit)
        = (raw[0] | 0x02) || raw[1:6]
```

The LAA (Locally Administered Address) bit distinguishes V-MAC from globally assigned hardware MACs per IEEE 802.

```
Example:
MasterPublicKey → SHA256 → first 6 bytes: AC:CE:7F:3A:B1:09
Set LAA bit:                              AE:CE:7F:3A:B1:09
V-MAC:                                    AE:CE:7F:3A:B1:09
```

The V-MAC is mathematically bound to the MasterPublicKey. Any server can verify:
(SHA256(claimed_MasterPublicKey)[0] | 0x02) == v_mac[0]
AND SHA256(claimed_MasterPublicKey)[1:6] == v_mac[1:6]

The MAC Attestation Proof for a V-MAC:

```
mac_proof = Sign(MasterPrivateKey,
                 "virtual" ||
                 mac_bytes ||
                 ts        ||
                 nonce)
```

V-MAC is RECOMMENDED for:

- Cloud agents and virtual machines
- Software bots without meaningful hardware identity
- Privacy-sensitive deployments (V-MAC reveals no hardware manufacturer information)

6.4.7.4. DNS Record Format with MAC Binding

When MAC binding is used, the DNS TXT record SHOULD include the mac= parameter:

```
; Real MAC binding (IoT device)
wm-bg-4471._saip.manufacturer.com. 300 IN TXT
  "v=saip1;
   mac=B8:27:EB:4A:3C:11;
   mac-type=real;
   pk=<MasterPublicKey>;
   asn=1234"

; V-MAC binding (cloud agent)
crawler-nyc-042._saip.acme.com. 300 IN TXT
  "v=saip1;
   mac=AE:CE:7F:3A:B1:09;
   mac-type=virtual;
   pk=<MasterPublicKey>;
   asn=64496"
```

Defined MAC-related DNS parameters:

mac=	The MAC address (real or virtual) in XX:XX:XX:XX:XX:XX format.
mac-type=	Either "real" or "virtual". MUST be present when mac= is present.

6.4.7.5. Network Identity: IP and ASN Binding

The network identity component of the triplet MAY be expressed as either an IP address, CIDR prefix, or ASN:

```
For stable deployments (physical servers, IoT):
  ip=203.0.113.42          (exact IP)
  ip=203.0.113.0/24        (CIDR prefix)

For dynamic deployments (cloud, autoscaling):
  asn=64496                (Autonomous System Number)
```

IP/ASN binding is OPTIONAL. When present, servers SHOULD verify that the source network matches the registered value. Mismatch SHOULD result in trust downgrade but MUST NOT cause automatic rejection without additional policy context, as legitimate network changes (ISP failover, cloud migration) may cause temporary mismatches.

6.4.7.6. Verification Triplet Summary

Component	Source	Verification Method
MasterPublicKey	DNS TXT pk=	Signature verification
MAC (real)	DNS TXT mac= + hardware	mac_proof verified by MasterPublicKey
MAC (virtual)	DNS TXT mac=	SHA256(pk) derivation check

	+ derivation	+ mac_proof by MasterKey
IP / ASN	DNS TXT ip= / asn=	Source address comparison (advisory, not mandatory)

All cryptographic verifications (MasterPublicKey signature, mac_proof) are REQUIRED for Class 3 classification. IP/ASN verification is RECOMMENDED but OPTIONAL.

6.5. Agent Letter of Intent (ALOI)

This section defines the fourth principle of the VICDM model, extending the core framework with behavioral commitment:

> Anonymous interaction is permitted.
> Identity assertion is permitted if cryptographically proven.
> Delegated infrastructure must be explicitly authorized.
> Declared intent MUST be honored.
> Self-reported deviation is good faith.
> Concealed deviation is fraud.

This principle is analogous to voluntary disclosure in financial regulation: entities that self-report violations to regulators receive reduced penalties compared to those discovered through external audit. SAIP applies this principle cryptographically — an agent that self-reports an ALOI deviation is demonstrably acting in good faith, and the ecosystem SHOULD reward that behavior with reduced sanctions.

6.5.1. Definition and Purpose

An Agent Letter of Intent (ALOI) is an optional, cryptographically bound declaration that an agent publishes at registration time, stating its operational scope, access boundaries, behavioral constraints, and rate limits.

The ALOI serves three purposes:

- Commitment: The agent formally declares what it will do, creating a verifiable behavioral contract.
- Detection: Servers can automatically detect deviation from declared intent without subjective judgment.
- Accountability: Deviation is cryptographically attributable to a specific agent instance, enabling proportional sanctions.

ALOI is OPTIONAL. An agent that does not publish an ALOI operates under standard SAIP identity verification only. An agent that publishes an ALOI is held to its declared constraints.

The ALOI complements robots.txt:

> robots.txt: The server declares what bots MAY access.
> (server-defined policy, unilateral)
>
> ALOI: The bot declares what it WILL do.
> (agent-defined commitment, cryptographic)

These are complementary mechanisms. A well-behaved agent SHOULD respect both robots.txt restrictions and its own ALOI declaration.

6.5.2. ALOI Parameters

The following parameters are defined for ALOI publication in

DNS TXT records. All ALOI parameters use the "aloi-" prefix for namespace isolation:

```
aloi=          Primary intent category. MUST be one of:
               "crawl"   — web crawling and indexing
               "backup"  — data backup and archival
               "monitor" — monitoring and health checks
               "api"     — API integration and data sync
               "smtp"    — mail relay and delivery
               "iot"     — IoT telemetry and control
               "sync"    — file and data synchronization
               Implementations MAY define additional values.

aloi-scope=    URL path scope the agent declares it will
               access. Supports glob patterns:
               /public/*     — all paths under /public/
               /blog/*.html  — HTML files under /blog/
               *             — unrestricted (use with caution)

aloi-rate=     Maximum request rate the agent self-imposes.
               Format: <number>/<unit>
               Examples: 10/sec, 1000/day, 60/min

aloi-exclude= Comma-separated list of paths or categories
               the agent declares it will NOT access:
               /admin, /api, /private, /user-data

aloi-ua=       User-Agent string the agent declares it will
               use. MUST be consistent with SAIP id=.
               Servers SHOULD flag mismatches.

aloi-expires= Unix timestamp after which this ALOI
               declaration should be considered expired.
               Agents SHOULD refresh ALOI before expiry.
```

6.5.3. ALOI Cryptographic Binding

The ALOI declaration MUST be cryptographically bound to the agent’s identity. This binding is achieved in two ways:

First, publication in the agent’s DNS TXT record ties the ALOI to the MasterPublicKey in the same record. Any server can verify that the ALOI was published by the legitimate key holder.

Second, each SAIP request MUST include an ALOI commitment hash:

```
aloi-hash = SHA256(canonical ALOI string from DNS record)
```

The aloi-hash= parameter is included in the SAIP header and MUST be part of the canonical string that is signed:

```
id=<id>;ts=<ts>;nonce=<nonce>;method=<METHOD>;path=<path>;
aloi-hash=<aloi-hash>
```

This cryptographically proves that the agent is operating under the stated ALOI for this specific request. A server that detects deviation has cryptographic proof of the commitment and the violation simultaneously.

6.5.4. DNS Record Format with ALOI

Example DNS TXT records with ALOI declaration:

```
; Web crawler with strict scope declaration
crawler-nyc-042._saip.acme.com. 300 IN TXT
  "v=saip1;
```

```
      pk=<MasterPublicKey>;
      aloi=crawl;
      aloi-scope=/public/*,/blog/*,/sitemap.xml;
      aloi-rate=10/sec;
      aloi-exclude=/api,/admin,/private,/user-data;
      aloi-ua=AcmeBot/2.0 (+https://acme.com/bot)"

   ; Backup agent with narrow scope
   backup-agent-01._saip.corp.com. 300 IN TXT
     "v=saip1;
      pk=<MasterPublicKey>;
      aloi=backup;
      aloi-scope=/exports/*,/snapshots/*;
      aloi-rate=100/min;
      aloi-exclude=/live,/user-sessions"

   ; IoT monitoring device
   sensor-bg-4471._saip.vendor.com. 300 IN TXT
     "v=saip1;
      pk=<MasterPublicKey>;
      aloi=iot;
      aloi-scope=/telemetry/submit,/config/read;
      aloi-rate=1/sec;
      aloi-exclude=/config/write,/admin,/firmware"
```

6.5.5. Server-Side ALOI Enforcement

Servers implementing ALOI enforcement MUST perform the following checks on each request from an ALOI-declaring agent:

1. Retrieve ALOI from DNS cache (same lookup as MasterPublicKey).

2. Verify aloi-hash= in SAIP header matches SHA256 of the DNS ALOI record. Mismatch indicates ALOI substitution attempt and MUST be treated as a Class 1 violation.

3. Check request path against aloi-scope=:
   If path does not match declared scope → scope violation.

4. Check request path against aloi-exclude=:
   If path matches any excluded pattern → exclusion violation.

5. Check request rate against aloi-rate=:
   If rate exceeds declared maximum → rate violation.

6. Check User-Agent against aloi-ua= (if present):
   If User-Agent does not match → identity inconsistency.

Violation detection MUST trigger the sanctions defined in Section 6.5.7.

6.5.6. ALOI Violation Response Header

When a server detects an ALOI violation, it SHOULD include a structured violation signal in the response:

```
SAIP-Violation: <violation-type>;
                declared=<declared-value>;
                attempted=<attempted-value>;
                penalty=<applied-penalty>
```

Example:

```
SAIP-Violation: scope-exceeded;
                declared=/public/*;
                attempted=/api/private-data;
```

penalty=throttle

Defined violation types:

```
scope-exceeded    Request path outside declared aloi-scope=
exclusion-breach  Request path matches aloi-exclude=
rate-exceeded     Request rate exceeds aloi-rate=
ua-mismatch       User-Agent inconsistent with aloi-ua=
aloi-expired      aloi-expires= timestamp has passed
aloi-hash-mismatch aloi-hash= does not match DNS record
```

6.5.7. Violation Sanctions and Scoring

ALOI violations MUST be handled proportionally. The following scoring model is RECOMMENDED:

Violation Count	Penalty	Action
1	Warning	SAIP-Violation header only. Log for audit. No rate change.
2-3	Throttle	Reduce to minimal rate limit. Report to vendor via header.
4-5	Downgrade	Trust downgrade to Class 1. Report to RE (optional).
6-10	Block	Reject all requests. Report to RE for revocation.
>10	Revocation	RE revokes InstanceID. Vendor notified.

Violation scores SHOULD decay over time. A RECOMMENDED decay rate is -1 point per 24 hours of compliant behavior.

Servers MAY report violations to the RE using a standardized violation report:

```
POST <re-endpoint>/violation
Content-Type: application/json

{
  "instance-id": "<agent-id>",
  "violation":   "<violation-type>",
  "declared":    "<declared-value>",
  "attempted":   "<attempted-value>",
  "ts":          <unix-timestamp>,
  "server-sig":  "<MasterKeySig_over_report>"
}
```

6.5.8. Agent Self-Reporting (Good Faith Mechanism)

An agent that detects it has unintentionally violated its own ALOI declaration MAY self-report the violation before the server detects it. This is the Good Faith Mechanism.

This principle is directly analogous to voluntary disclosure in financial and regulatory contexts: entities that proactively report their own violations to authorities receive reduced penalties compared to those discovered through external audit.

Self-reporting is performed by including a self-report signal in the immediately following request:

```
SAIP-Self-Report: violation=<violation-type>;
                  declared=<declared-value>;
                  attempted=<attempted-value>;
                  ts=<unix-timestamp-of-violation>;
                  self-sig=<Sign(MasterPrivateKey,
                               violation-type||declared||
                               attempted||ts||nonce)>
```

The self-sig= MUST be signed by the MasterPrivateKey. This proves the self-report is genuine and cannot be fabricated by a third party.

Self-reported violations MUST result in reduced penalties:

Condition	Server-Detected	Self-Reported
1st violation	Warning	No penalty
2nd-3rd	Throttle	Warning only
4th-5th	Downgrade	Throttle
6th-10th	Block	Downgrade
>10	RE Revocation	Block (no revocation)

Agents that consistently self-report SHOULD receive elevated trust scores over time, as self-reporting demonstrates that the agent’s vendor has implemented responsible operational practices.

The Good Faith Mechanism creates a positive incentive structure: vendors who build self-reporting into their agents benefit from higher trust levels and reduced sanctions, while vendors whose agents conceal violations face progressively harsher penalties.

Class	Identity Status	Trust Level	Recommended Policy
Class 3	Fully Verified	High	Full access, may receive elevated rate limits
Class 2	Partially Verified	Medium	Standard access with monitoring
Class 0	Anonymous	Low	Default anonymous policy
Class 1	Unverifiable Claim	Minimal	Lower than Class 0 log for audit

Note that Class 1 (unverifiable claim) receives LOWER trust than Class 0 (anonymous). This is intentional: an entity that actively misrepresents its identity is more concerning than one that simply does not identify itself.

8. Policy Enforcement

Systems SHOULD NOT rely on binary allow/deny decisions. Instead, they SHOULD apply adaptive trust-based policies proportional to the verified identity class.

Policy actions MAY include:

- Full acceptance and priority handling for Class 3 clients
- Standard acceptance with monitoring for Class 2 clients
- Default anonymous handling for Class 0 clients
- Throttling and logging for Class 1 clients
- Rejection in high-risk or sensitive contexts for Class 1

The specific thresholds and actions are deployment-specific and outside the scope of this document.

9. Relationship to Anonymous Authentication

Anonymous authentication systems (such as Privacy Pass and related mechanisms being developed in the IETF webbotauth working group) allow a client to prove it is vouched for by a trusted Attester without revealing its specific identity.

Anonymous authentication is fully compatible with VICDM:

- A client using anonymous attestation falls under Class 0 (Anonymous) or Class 2 (Partially Verified, via Attester) depending on the Attester’s trust level.

- A client using identified authentication (such as SAIP [SAIP]) falls under Class 3 (Fully Verified) when verification succeeds.

The two models serve different operational requirements:

Anonymous attestation is appropriate when:
- The site’s primary concern is rate limiting and abuse prevention at the Attester level
- Bot privacy is a design requirement
- Per-instance accountability is not needed

Identified authentication is appropriate when:
- Per-instance revocation is required
- Audit trails are a compliance requirement
- Reputation systems depend on persistent agent identity
- IoT or enterprise fleet management is involved

Systems MAY support both models simultaneously and apply different policies to each class of client.

10. Non-Goals

This document does NOT:

- Eliminate or restrict anonymous interaction. Anonymous access MUST remain a first-class mode of operation.

- Require universal identity adoption. Identity remains opt-in.

- Define a mandatory implementation protocol. Protocol specifications implementing these principles are separate documents (see [SAIP]).

- Replace existing standards for authentication, authorization, or access control.

- Define specific rate limits, trust scores, or policy

thresholds. These are deployment-specific.

11. Security Considerations

11.1. False Identity Claims

The primary security concern addressed by this document is false identity assertion — clients that claim to be trusted entities without cryptographic proof.

By defining Class 1 (unverifiable claim) as receiving lower trust than Class 0 (anonymous), this model removes the incentive to make unverifiable identity claims. An agent that cannot prove its identity is better served by not claiming one.

11.2. DNS Security

DNS-based verification (Section 5.1 and 6.2) is subject to DNS spoofing and cache poisoning attacks. Implementations MUST use DNSSEC [RFC4033] where available to validate DNS responses used for identity verification.

11.3. Delegation Scope Creep

Delegation tokens and DNS delegation records SHOULD be scoped as narrowly as possible. Overly broad delegation (e.g., an open-ended authorization for all infrastructure operated by a large provider) defeats the purpose of the delegation model by creating de facto anonymous identity claims under a trusted name.

11.4. Attester Compromise

In anonymous attestation models, compromise of an Attester allows issuance of credentials to malicious clients. This model does not address Attester security directly; that is the responsibility of the attestation protocol specification.

12. Privacy Considerations

This document preserves anonymity as a first-class mode of operation. No client is required to assert an identity.

When a client does assert an identity, the verification mechanisms in Section 5 may disclose information about the client’s network location and infrastructure to the verifying server. Clients that wish to preserve privacy SHOULD use anonymous interaction (Class 0) rather than identity assertion.

DNS-based Attestation Discovery records (Section 6.2) are publicly accessible. Entities publishing such records should be aware that the records disclose information about their infrastructure and delegation relationships.

13. IANA Considerations

This document requests IANA to create the following new registry:

Registry name:	VICDM DNS Delegation Record Parameters
Registration policy:	Specification Required
Initial contents:	v, re, pk, asn, ip, exp as defined in Section 6.2 of this document.

This document has no other IANA actions.

14. References

14.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[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>.

14.2. Informative References

[RFC7208] Kitterman, S., "Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1", RFC 7208, DOI 10.17487/RFC7208, April 2014, <https://www.rfc-editor.org/info/rfc7208>.

[RFC6376] Crocker, D., Ed., Hansen, T., Ed., and M. Kucherawy, Ed., "DomainKeys Identified Mail (DKIM) Signatures", STD 76, RFC 6376, DOI 10.17487/RFC6376, September 2011, <https://www.rfc-editor.org/info/rfc6376>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[SAIP] Jovanevi, S., "SAIP: Signed Agent Identity Protocol", draft-jovancevic-saip-08, April 2026, <https://datatracker.ietf.org/doc/draft-jovancevic-saip/>.

Author’s Address

Srecko Jovancevic
SKGO, IKT Support
Makedonska 22
11000 Belgrade
Serbia

Email: srecko.jovancevic@skgo.org
Email: srecko.jovancevic@gmail.com
URI: https://github.com/sreckojovancevic