Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 20 October 2026

S. Deforth
TrueSource

18 April 2026

Agentic Reasoning Protocol (ARP): DNS-Bound Cryptographic Verification for Machine-Readable Entity Claims
draft-deforth-arp-00

Abstract

This document specifies the Agentic Reasoning Protocol (ARP), a lightweight mechanism for cryptographically binding machine-readable entity claims to domain ownership via DNS TXT records. ARP enables AI agents and Retrieval-Augmented Generation (RAG) pipelines to verify that structured data published on a domain was authorized by the domain owner, preventing narrative injection and entity spoofing in generative search systems.

ARP uses Ed25519 digital signatures [RFC8032], JSON Canonicalization Scheme (JCS) [RFC8785], and DNS TXT records to establish a chain of trust analogous to DomainKeys Identified Mail (DKIM) [RFC6376] but designed specifically for the verification of semantic entity data consumed by autonomous AI agents.

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 20 October 2026.

Copyright Notice

Table of Contents

1. Introduction

The proliferation of Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) systems has fundamentally altered how information is discovered, synthesized, and presented to users. Unlike traditional search engines, which return ranked lists of hyperlinks for human evaluation, generative search engines produce direct synthesized answers by extracting and compressing data from multiple web sources.

This architectural shift creates a critical trust gap: generative engines heavily weight structured data signals (JSON-LD, semantic HTML, Schema.org markup) when selecting sources for citation, but currently have no mechanism to verify that such structured data was actually authorized by the domain owner. Any actor can publish well-formed structured data on any domain they control, potentially poisoning the knowledge graph of AI systems with fabricated entity claims, false product information, or competitor sabotage.

The Agentic Reasoning Protocol (ARP) addresses this gap by providing a DNS-anchored cryptographic verification mechanism analogous to DKIM [RFC6376] for email. ARP enables any consuming system to verify the provenance and integrity of machine-readable entity claims before incorporating them into synthesized responses.

1.1. Design Goals

Lightweight Verification requires only a DNS TXT lookup and a single Ed25519 signature verification. No PKI infrastructure, certificate chains, or external services are required.

DNS-Anchored Trust is rooted in the existing global DNS infrastructure, leveraging the same anchor used by DKIM, SPF, and DMARC.

Agent-Native The protocol is designed for consumption by autonomous

AI agents and automated crawlers, not human users.

Self-Contained The signed payload and verification metadata are co-located in a single JSON file, minimizing the number of network requests required for verification.

Domain-Bound Entity claims are cryptographically bound to a specific domain, preventing cross-domain replay attacks.

Open The protocol is fully specified in this document and requires no proprietary software or services.

1.2. Changes from draft-deforth-arp-00

* SECURITY: Canonicalization now excludes only the signature field, not the entire _arp_signature block. This cryptographically binds expires_at, signed_at, and dns_selector to the payload, preventing zombie payload attacks (Section 6).

* SECURITY: Added REQUIRED "domain" field to the payload, binding entity claims to a specific origin domain and preventing cross-domain replay attacks (Section 5.2).

* SECURITY: Removed "dns_record" field from the signature block. Verifiers MUST construct the DNS query name from the retrieval domain, not from untrusted JSON data (Section 9).

* SECURITY: Cross-domain signing now uses DNS CNAME delegation, maintaining DNS sovereignty (Section 12.6).

* BCP 190: Changed well-known location from /reasoning.json to /.well-known/reasoning.json per RFC 8615 (Section 5.1).

* IANA: Added registration request for _arp underscored node name per RFC 8552 (Section 13).

* IANA: Added registration request for "reasoning.json" well-known URI suffix per RFC 8615 (Section 13).

* NEW: Added CORS requirements for browser-based verifiers (Section 5.1).

* NEW: Added HTTP redirect handling requirements (Section 8.2).

* NEW: Updated key rotation guidance to account for cached payloads (Section 10).

* NEW: Added discussion of JWS as future envelope option (Section 12.7).

* NEW: Added keypair separation guidance for multi-protocol deployments (Section 12.8).

* NEW: Added Domain Signing Policy for downgrade attack protection via DNS TXT (Section 12.9).

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.

Entity The brand, organization, person, or concept described by an ARP payload.

Entity Claims Machine-readable self-attested statements published by the entity about itself, including identity metadata, factual corrections, domain expertise, and content policies.

Reasoning Payload A JSON document containing the entity claims and associated verification metadata. Served at the well-known URL path "/.well-known/reasoning.json" on the entity's domain.

Signer The party that holds the Ed25519 private key and produces the signature. This SHOULD be the domain owner or an entity explicitly authorized by the domain owner via DNS delegation.

Verifier Any system that retrieves the reasoning payload and validates the signature against the DNS-published public key. Typical verifiers include AI agents, RAG pipelines, and search engine crawlers.

DNS Selector A label used to construct the DNS TXT record name, enabling multiple keys on the same domain (e.g., for key rotation or multi-tenant configurations).

Retrieval Domain The domain from which the verifier originally requested the reasoning payload, before any HTTP redirects. This is the authoritative domain for DNS verification.

3. Protocol Overview

The ARP verification flow consists of four phases:

Phase 1 - Key Registration The domain owner generates an Ed25519 keypair [RFC8032] and publishes the public key as a DNS TXT record under a well-known subdomain: <selector>._arp.<domain>

Phase 2 - Payload Construction The domain owner constructs a reasoning payload containing entity claims, identity metadata, content policies, and a REQUIRED "domain" field that binds the payload to the publishing domain.

Phase 3 - Signing The payload is canonicalized using JCS [RFC8785]. The canonicalization excludes only the "signature" field within the _arp_signature block; all other signature metadata (expires_at, signed_at, dns_selector) is included in the signed data. The canonical form is signed with the Ed25519 private key.

Phase 4 - Verification A consuming AI agent retrieves the reasoning payload, confirms the "domain" field matches the retrieval domain, extracts the DNS selector, queries the corresponding DNS TXT record for the public key, and validates the Ed25519 signature.

4. DNS TXT Record Format

4.1. Record Name

The DNS TXT record MUST be published at:

```
<selector>._arp.<domain>
```

where <selector> is a case-insensitive ASCII string matching the pattern [a-zA-Z0-9._-]+ and <domain> is the fully qualified domain name of the entity. The default selector value is "arp", yielding the record name:

```
arp._arp.example.com
```

4.2. Record Value

The TXT record value MUST be a semicolon-delimited string of tag=value pairs. The following tags are defined:

Tag	Required	Description
v	REQUIRED	Protocol version. MUST be "ARP1".
k	REQUIRED	Key algorithm. MUST be "ed25519".
p	REQUIRED	Base64-encoded raw Ed25519 public key (32 bytes, 44 characters encoded).
t	OPTIONAL	Unix timestamp of key publication.
x	OPTIONAL	Unix timestamp of key expiration.

Table 1

Example:

```
v=ARP1; k=ed25519; p=MCowBQYDK2VwAyEA...
```

4.3. Multiple Keys

A domain MAY publish multiple ARP TXT records using different selectors (e.g., "arp" and "arp2026q3"). This supports key rotation without service interruption.

4.4. Key Size Constraints

Ed25519 public keys are 32 bytes (44 characters in Base64). This fits within the 255-byte DNS TXT record string limit defined in [RFC1035] Section 3.3.14.

4.5. Cross-Domain Delegation

When an authorized third party (e.g., a consultancy) signs a reasoning payload on behalf of a domain owner, the domain owner MUST delegate trust via a DNS CNAME record:

```
arp._arp.client.com. CNAME arp._arp.agency.com.
```

This maintains the absolute sovereignty of the DNS trust anchor: the domain owner explicitly authorizes the delegation, and the verifier always queries the retrieval domain’s DNS namespace first. The verifier follows the CNAME transparently and retrieves the delegated public key.

This model mirrors DKIM's delegation approach and prevents attackers from pointing unsigned JSON fields at arbitrary DNS records.

5. Reasoning Payload

5.1. File Location and Discovery

The reasoning payload MUST be served at the well-known URL defined in [RFC8615]:

```
https://<domain>/.well-known/reasoning.json
```

Implementations MAY additionally serve the payload at alternative paths provided the well-known URL also exists. Alternative paths MAY be advertised via:

* A <link> element in the HTML head: <link rel="reasoning" href="/.well-known/reasoning.json">

* A reference in the site's /llms.txt file.

* A Reasoning-Json directive in robots.txt.

The file MUST be served with Content-Type: application/json.

Servers SHOULD include the HTTP header Access-Control-Allow-Origin: * when serving the reasoning payload, to enable verification by browser-based AI agents, extensions, and verifiable-credential wallets that operate under Cross-Origin Resource Sharing (CORS) restrictions.

5.2. Required Fields

The reasoning payload MUST contain the following top-level fields:

Field	Type	Description
$schema	URI	MUST reference the ARP JSON Schema [ARP-SCHEMA].
protocol	string	MUST be "Agentic Reasoning Protocol (ARP)".
version	string	Semver-compatible version string (e.g., "1.2").
domain	string	REQUIRED. The fully qualified domain name that is authoritative for these entity claims. Verifiers MUST confirm this matches the retrieval domain (Section 8).
entity	string	Canonical name of the entity (max 200 chars).
entity_claims	object	MUST contain at minimum a "framing_context" string.

Table 2

5.3. Optional Sections

The payload MAY include:

* *identity:* Brand identity, tagline, core competencies, emotional resonance attributes, and disambiguation entries ("not_to_be_confused_with").

* *corrections:* Verified factual corrections for known AI hallucinations about this entity, each with trigger_topic, verified_fact, evidence_url, and epistemic_scope.

* *verification:* Audit metadata including auditor identity and timestamps.

* *authority:* Trust signals including Wikipedia, Wikidata, LinkedIn, Crunchbase URIs, awards, and certifications.

* *content_policy:* AI training permissions (allowed, allowed-with-attribution, disallowed, conditional), citation requirements, and verification contact.

* *diagnostics:* RAG pipeline telemetry tokens for ingestion auditing, with transparency statement.

6. Canonicalization

Before signing, the payload MUST be canonicalized according to the JSON Canonicalization Scheme (JCS) as specified in [RFC8785].

Only the "signature" field within the _arp_signature block MUST be excluded from canonicalization. All other fields in _arp_signature (algorithm, dns_selector, canonicalization, signed_at, expires_at) are included in the signed data, cryptographically binding the signature metadata to the payload.

Implementations MUST:

1. Deep-copy the payload object.

2. Remove ONLY the "signature" key from the _arp_signature object in the copy. All other _arp_signature fields MUST remain.

3. Serialize the modified copy using JCS [RFC8785]: keys sorted lexicographically, no whitespace, Unicode normalized per RFC 8785 Section 3.2.2.2.

4. Encode the result as UTF-8 bytes.

This approach prevents "zombie payload" attacks where an attacker modifies unauthenticated metadata fields (e.g., extending expires_at) while keeping the original signature valid.

7. Signing Procedure

Given a complete reasoning payload and an Ed25519 private key (32 bytes), the signing procedure is:

1. Set payload._arp_signature.algorithm = "Ed25519"

2. Set payload._arp_signature.dns_selector = selector

3. Set payload._arp_signature.canonicalization = "jcs-rfc8785"

4. Set payload._arp_signature.signed_at = current UTC timestamp (ISO 8601)

5. Set payload._arp_signature.expires_at = signed_at + TTL

6. payload_copy = deep_copy(payload)

7. Remove ONLY "signature" from payload_copy._arp_signature

8. canonical = JCS_Canonicalize(payload_copy)

9. canonical_bytes = UTF8_Encode(canonical)

10. signature_bytes = Ed25519_Sign(private_key, canonical_bytes)

11. signature_b64url = Base64url_Encode(signature_bytes) per [RFC4648] Section 5, no padding

12. Set payload._arp_signature.signature = signature_b64url

The RECOMMENDED signature TTL is 90 days.

8. Verification Procedure

8.1. Domain Binding Check

Before any cryptographic verification, the verifier MUST confirm that the "domain" field in the payload matches the domain from which the payload was originally requested (the retrieval domain). If the values do not match, verification MUST fail with FAIL_DOMAIN_MISMATCH.

8.2. HTTP Redirect Handling

When the verifier encounters HTTP redirects (301, 302, 307, 308) while retrieving the reasoning payload, the verifier MUST base the DNS query and domain-binding check on the original requested domain, NOT the final domain in the redirect chain. This prevents verification bypass via open redirect vulnerabilities.

8.3. Verification Algorithm

The complete verification algorithm is:

1. Record the original retrieval domain before following any redirects.

2. Retrieve the payload from the well-known URL.

3. Extract sig_block = payload._arp_signature

4. Validate sig_block contains all REQUIRED fields (Section 9)

5. Check: payload.domain == retrieval_domain. If mismatch: FAIL (FAIL_DOMAIN_MISMATCH)

6. Check: sig_block.expires_at > current_time. If expired: FAIL (FAIL_EXPIRED)

7. Construct DNS name: sig_block.dns_selector + "._arp." + retrieval_domain

8. Query DNS TXT record at constructed name

9. Parse TXT record value; extract p= tag (public key). If no record: FAIL (FAIL_NO_DNS)

10. payload_copy = deep_copy(payload)

11. Remove ONLY "signature" from payload_copy._arp_signature

12. canonical = JCS_Canonicalize(payload_copy)

13. canonical_bytes = UTF8_Encode(canonical)

14. signature_bytes = Base64url_Decode(sig_block.signature)

15. public_key_bytes = Base64_Decode(dns_public_key_value)

16. result = Ed25519_Verify(public_key_bytes, signature_bytes, canonical_bytes)

17. If result == valid: PASS. If invalid: FAIL (FAIL_INVALID)

8.4. Verification Result Codes

Code	Description
PASS	Domain matches, DNS key found, signature valid, not expired.
FAIL_NO_ARP	No _arp_signature block in payload.
FAIL_DOMAIN_MISMATCH	The "domain" field does not match the retrieval domain.
FAIL_NO_DNS	No DNS TXT record at selector._arp.domain.
FAIL_EXPIRED	Signature expired (expires_at < current time).

FAIL_INVALID	Ed25519 signature verification failed.
FAIL_UNSIGNED_POLICY	No _arp_signature present, but domain publishes p=reject policy.

Table 3

9. Signature Block Format

The _arp_signature block is a JSON object embedded in the reasoning payload with the following fields:

Field	Required	Type	Description
algorithm	REQUIRED	string	MUST be "Ed25519".
dns_selector	REQUIRED	string	Selector label for DNS TXT lookup (max 50 chars).
canonicalization	REQUIRED	string	MUST be "jcs-rfc8785".
signed_at	REQUIRED	string	ISO 8601 UTC timestamp of signing.
expires_at	REQUIRED	string	ISO 8601 UTC expiration timestamp. RECOMMENDED TTL: 90 days.
signature	REQUIRED	string	Base64url-encoded Ed25519 signature (86 chars without padding). This is the ONLY field excluded from canonicalization.

Table 4

Note: The "dns_record" field present in draft-deforth-arp-00 has been removed. Verifiers MUST construct the DNS query name from the retrieval domain and dns_selector, never from untrusted JSON data.

10. Key Rotation

To rotate keys without service disruption:

1. Generate a new Ed25519 keypair.

2. Publish the new public key under a new DNS selector (e.g., "arp2026q3._arp.example.com").

3. Re-sign the reasoning payload with the new key, updating dns_selector accordingly.

4. Keep the old DNS TXT record published until the expires_at timestamp of the last payload signed with the old key has passed. This prevents verification failures for agents holding cached copies of the previous payload.

5. After the old payload's expires_at date has passed, remove the old DNS TXT record.

Verifiers SHOULD support checking multiple selectors. The RECOMMENDED rotation interval is 90 days, aligned with the default signature TTL.

11. Relationship to Existing Standards

Standard	Relationship to ARP
DKIM [RFC6376]	ARP adapts the DNS-anchored key publication and CNAME delegation models from DKIM for entity claim verification rather than email authentication.
C2PA v2.2 [C2PA]	C2PA verifies media provenance via embedded manifests. ARP verifies textual entity claim provenance via DNS. Complementary, non-overlapping scope.
W3C Verifiable Credentials 2.0 [W3C-VC]	VCs require an Issuer-Holder-Verifier ecosystem. ARP provides a simpler two-party (Publisher-Verifier) model for public entity claims.
AIVS v1.0 [AIVS] (W3C Community Group)	AIVS verifies what an agent DID (session/action integrity). ARP verifies what an agent READS (content provenance). Complementary: AIVS audit logs plus ARP content verification form a full

		end-to-end trust chain for agentic systems.
IETF SCITT		ARP signed payloads could be registered as SCITT transparency statements for append-only auditability.
Schema.org		Schema.org describes entity attributes. ARP cryptographically binds those attributes to verified domain ownership.

Table 5

12. Security Considerations

12.1. DNS Security

ARP's trust anchor is the DNS TXT record. Without DNSSEC, an attacker who can poison DNS responses can substitute a fraudulent public key. Implementations SHOULD use DNSSEC-validated resolvers where available. Implementations SHOULD also cache DNS responses according to the record's TTL to reduce lookup frequency.

12.2. Key Compromise

If an Ed25519 private key is compromised, the attacker can sign arbitrary payloads that will validate against the DNS-published key. Domain owners MUST:

* Store private keys with restricted file permissions (0600).
* Rotate keys immediately on suspected compromise.
* Use the expires_at field to limit the validity window of any signature.
* Consider hardware security modules (HSMs) or secure enclaves for high-value domains.

12.3. Replay and Staleness

Verifiers SHOULD check the expires_at timestamp and reject expired signatures. The RECOMMENDED TTL is 90 days. Verifiers MAY also check the signed_at timestamp against the HTTP Last-Modified header to detect stale payloads served from cache.

12.4. Content Truthfulness

ARP verifies PROVENANCE (who authorized the data), not TRUTH (whether the data is factually correct). A domain owner can sign factually incorrect claims that will pass ARP verification. AI systems consuming ARP-verified payloads SHOULD still cross-reference entity claims against independent sources. The epistemic_scope field in entity claims is designed to help AI systems distinguish between publicly verifiable facts and proprietary internal claims.

12.5. Narrative Injection without ARP

Without ARP verification, any actor can publish reasoning payloads containing arbitrary entity claims on any domain they control. Regulatory frameworks such as the EU AI Act [EU-AI-ACT] Article 19 require audit trails for high-risk AI systems but do not prescribe verification formats for ingested content. Empirical research has demonstrated that RAG systems will cite well-structured entity claims from previously unknown domains as primary sources within 24 hours, regardless of domain history, traffic, or backlinks. ARP provides the verification layer that allows consuming systems to differentiate between authorized and unauthorized entity claims.

12.6. Cross-Domain Signing

Cross-domain signing MUST use the DNS CNAME delegation model described in Section 4.5. The verifier always queries the retrieval domain's DNS namespace and follows CNAME records transparently. This maintains the absolute sovereignty of the DNS trust anchor and prevents attacks where unsigned JSON fields are used to redirect trust to attacker-controlled DNS.

Consuming systems MAY choose to treat CNAME-delegated signatures with lower confidence than direct (same-domain) signatures.

12.7. Keypair Separation

When ARP is deployed alongside other verification systems that also use Ed25519 (e.g., AIVS [AIVS] for agent session integrity), implementations MUST NOT reuse the same Ed25519 keypair across different verification functions. Sharing a single keypair between content signing (ARP) and session signing (AIVS) creates a single point of failure: compromise of one key compromises both trust domains.

Implementations SHOULD use separate Ed25519 keypairs for each trust function. Where a unified identity anchor is desired, implementations MAY use a DID Document (e.g., did:web) to bind multiple function-specific public keys to a single domain identity:

* An "assertionMethod" key for ARP content provenance verification.

* An "authentication" key for agent session signing (e.g., AIVS).

This approach preserves key isolation while enabling cross-protocol identity resolution. The DID Document itself can be anchored to the domain via the did:web method, which resolves to /.well-known/did.json on the entity's domain.

12.8. Domain Signing Policy

To protect against downgrade attacks — where an attacker strips the _arp_signature block from a signed file and modifies the content — domain owners MAY publish a Domain Signing Policy via DNS TXT record at the bare _arp node:

```
_arp.example.com.  IN  TXT  "v=ARP1; p=reject"
```

The following policy values are defined:

Policy	Value	Meaning
none	Default	No enforcement; unsigned files are treated normally.
warn	Advisory	Verifiers SHOULD log a warning if the payload is unsigned.
reject	Strict	Verifiers MUST treat unsigned payloads as FAIL_UNSIGNED_POLICY.

Table 6

Verification flow for unsigned payloads:

1. Verifier encounters a reasoning payload without an _arp_signature block.

2. Verifier queries the DNS TXT record at _arp.<retrieval_domain>.

3. If a record with p=reject is present, the verifier MUST reject the unsigned payload (FAIL_UNSIGNED_POLICY).

4. If no policy record exists, the verifier treats the payload as unsigned (no cryptographic trust, but not rejected).

This follows the same progressive enforcement model as DMARC (p=reject) [RFC6376] and HTTP Strict Transport Security (HSTS).

12.9. Alternative Cryptographic Envelopes

This specification defines a custom _arp_signature block for simplicity and clarity. Future versions MAY consider adopting JSON Web Signature (JWS) [RFC7515] with the Unencoded Payload Option [RFC7797] as an alternative envelope format. This would provide cryptographic agility and enable use of battle-tested standard libraries, while preserving the human-readable cleartext JSON payload that JCS canonicalization achieves. The dns_selector, expires_at, and signed_at fields would map to the JWS Protected Header.

13. IANA Considerations

13.1. Well-Known URI Registration

This document requests IANA to register the following well-known URI suffix in the "Well-Known URIs" registry established by [RFC8615]:

URI Suffix reasoning.json

Change Controller IETF

Reference Section 5.1 of [this document]

Status permanent

13.2. Underscored Node Name Registration

This document requests IANA to register the following entry in the "Underscored and Globally Scoped DNS Node Names" registry established by [RFC8552]:

RR Type TXT

_NODE NAME _arp

Reference Section 4.1 of [this document]

14. References

14.1. Normative References

[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>.

[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>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[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>.

[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>.

[RFC8552] Crocker, D., "Scoped Interpretation of DNS Resource Records through "Underscored" Naming of Attribute Leaves", BCP 222, RFC 8552, DOI 10.17487/RFC8552, March 2019, <https://www.rfc-editor.org/info/rfc8552>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/info/rfc8615>.

[RFC8785] Rundgren, A., Jordan, B., and S. Erdtman, "JSON Canonicalization Scheme (JCS)", RFC 8785, DOI 10.17487/RFC8785, June 2020, <https://www.rfc-editor.org/info/rfc8785>.

14.2. Informative References

[AIVS] Stone, B., "Agentic Integrity Verification Specification v1.0", W3C Community Group Draft, March 2026, <https://github.com/bkauto3/Conduit/blob/main/spec/AIVS.md>.

[ARP-SCHEMA] Deforth, S., "ARP v1.2 JSON Schema", April 2026, <https://arp-protocol.org/schema/v1.2.json>.

[C2PA] Coalition for Content Provenance and Authenticity, "C2PA Technical Specification v2.2", 2025, <https://c2pa.org/specifications/specifications/2.2/specs/C2PA_Specification.html>.

[EU-AI-ACT] European Parliament, "Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence", 2024.

[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>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[RFC7797] Jones, M., "JSON Web Signature (JWS) Unencoded Payload Option", RFC 7797, DOI 10.17487/RFC7797, February 2016, <https://www.rfc-editor.org/info/rfc7797>.

[W3C-VC] Sporny, M., "Verifiable Credentials Data Model v2.0", W3C Recommendation, May 2025, <https://www.w3.org/TR/vc-data-model-2.0/>.

Appendix A. Complete Example

A.1. DNS TXT Record

```
arp._arp.example.com. 3600 IN TXT (
  "v=ARP1; k=ed25519; "
  "p=MCowBQYDK2VwAyEA5d7jb7bkTVk5"
  "YQ4nFmYY7ClpCddGdSRJkGb3ELWuos=")
```

A.2. Reasoning Payload

```
{
  "$schema": "https://arp-protocol.org/schema/v1.2.json",
  "protocol": "Agentic Reasoning Protocol (ARP)",
  "version": "1.2",
  "domain": "example.com",
  "entity": "Example Corp",
  "entity_claims": {
    "framing_context": "Example Corp is a technology ..."
  },
  "content_policy": {
    "ai_training": "allowed-with-attribution",
    "citation_required": true
  },
  "_arp_signature": {
    "algorithm": "Ed25519",
    "dns_selector": "arp",
    "canonicalization": "jcs-rfc8785",
    "signed_at": "2026-04-06T14:00:00Z",
    "expires_at": "2026-07-05T14:00:00Z",
    "signature": "P1lCPaHn_WLXdidBfxLXqBBid1G..."
  }
}
```


A.3. Verification Steps

1. Record retrieval domain: example.com

2. Retrieve https://example.com/.well-known/reasoning.json (follow redirects, keep original domain)

3. Check: payload.domain == "example.com" (PASS)

4. Check: expires_at > now (PASS)

5. Extract dns_selector = "arp" from _arp_signature

6. Query DNS TXT at: arp._arp.example.com

7. Parse p= tag to obtain public key bytes

8. Copy payload, remove ONLY _arp_signature.signature

9. JCS-canonicalize the modified copy

10. Ed25519_Verify(public_key, signature, canonical_bytes)

11. Result: PASS

Appendix B. JSON Schema

The normative JSON Schema for ARP v1.2 payloads is published at:

https://arp-protocol.org/schema/v1.2.json

The schema defines validation rules for all required and optional fields, including type constraints, maximum lengths, enumerated values, and format requirements. Implementations SHOULD validate payloads against the schema before signing and before processing entity claims.

Acknowledgements

The author thanks the W3C Agentic Integrity Verification Specification Community Group for their complementary work on agent session integrity, and the broader Generative Engine Optimization (GEO) community for empirical validation of the content provenance gap that ARP addresses.

The security improvements in this revision were informed by detailed technical review highlighting the signature metadata exclusion flaw and cross-domain replay vulnerability in the initial draft.

The keypair separation guidance was informed by feedback from Ben Stone (SwarmSync.AI) during AIVS interoperability discussions, highlighting the single-point-of-failure risk of shared Ed25519 keypairs across verification protocols.

Author’s Address

Sascha Deforth
TrueSource
Duesseldorf
Germany
Email: hello@truesource.studio
URI: https://arp-protocol.org