UniKey RFC 3001

Title: Verifier DNS Hardening Algorithm

Status: Draft

Category: Standards Track

Version: 1.0

Date: February 2026

Authors: Swoop Engineering Team

Previously published as: RFC-002 (Swoop Series)

Canonical URL: /rfc-3001

Abstract

This document specifies the deterministic DNS validation algorithm that a UniKey verifier MUST execute to discover and trust public keys at internet scale. To mitigate the inherent risks of DNS (spoofing, poisoning, and path-based attacks), this algorithm treats DNS as a probabilistic system where trust is established through multi-resolver correlation and consistency. This protocol ensures that the "Root of Trust" for any UniKey Trust Packet is established without relying on a single point of failure or centralized certificate authority.

---

1. Introduction

The UniKey ecosystem relies on the Domain Name System (DNS) and DKIM-style records to distribute the public keys used to sign Trust Packets. Because standard DNS queries are susceptible to various interception attacks, a naive "query and trust" model is insufficient for high-value agentic commerce. This algorithm defines the "Hardening" steps necessary to promote a DNS response to a "Trusted Key."

2. Scope

This specification covers the discovery of public keys via DNS, the normalization of resolver responses, and the logic for detecting suspicion or cache-poisoning attempts. It does not cover the cryptographic verification of the signatures themselves (see **UniKey RFC 2001**).

3. Terminology

The key words **MUST**, **MUST NOT**, **SHOULD**, **SHOULD NOT**, and **MAY** in this document are to be interpreted as described in RFC 2119.

- **N-Resolver Consensus:** The requirement that multiple independent DNS resolvers return identical records before a key is trusted.
- **Key Selector:** The sub-domain prefix used to locate specific public keys (e.g., selector._unikey.example.com).

4. Requirements (Normative)

1. Verifiers **MUST** query at least **N ≥ 3** independent DNS resolvers.
2. Public key material **MUST** match across all successful resolver responses exactly.
3. Any inconsistency in the public key payload across resolvers **MUST** result in immediate rejection.

5. Specification (The Hardening Algorithm)

5.1 Inputs

The verifier requires three parameters to initiate discovery:

- domain: The target domain (e.g., example.com).
- selector: The specific key version/identifier.
- expected_algorithm: (Optional) The cryptographic family expected for this transaction.

5.2 Step 1: Multi-Resolver Query

The verifier initiates parallel queries for the DKIM-formatted TXT record. Resolvers **SHOULD** be selected from different network paths:

1. **Path A:** Local OS / Corporate Resolver.
2. **Path B:** Global Public Resolver (e.g., 8.8.8.8 or 1.1.1.1).
3. **Path C:** Independent Third-Party or Peer-to-Peer Resolver.

5.3 Step 2: Response Normalization

Before comparison, all responses **MUST** be normalized:

- Remove insignificant whitespace.
- Standardize DKIM tag ordering (e.g., v=; k=; p=;).
- Canonicalize character encoding to UTF-8.

5.4 Step 3: Consistency Check

The verifier compares the normalized strings.

- **Rules:** The p= (public key) field **MUST** be bitwise identical across all responses.
- **Signal:** If 2 out of 3 resolvers return a record but they do not match, the verifier **MUST** trigger a "Suspicion Event" and reject the key.

5.5 Step 4: TTL and Change Analysis

The verifier **SHOULD** evaluate the Time-To-Live (TTL) of the record.

- Sudden drops in TTL (e.g., from 3600 to 60) are a known indicator of an imminent DNS takeover or poisoning attempt and **SHOULD** increase the "Suspicion Score" of the result.

5.6 Step 5: DNSSEC Validation

If DNSSEC signatures are present, the verifier **SHOULD** validate them.

- While DNSSEC success increases confidence, it **MUST NOT** be used as a reason to bypass the N-Resolver Consensus check.

6. Trust Decision Matrix

Result	**Decision**	**Action**
All Resolvers Identical	**Trusted**	Proceed to signature verification.
Minority Conflict (2 match, 1 differs)	**Reject**	Log as potential poisoning attempt.
DNSSEC Failure	**Reject**	Fail closed.
Key Material Match / Tag Mismatch	**Scrutiny**	Re-validate or use strict interpretation.

7. Caching and Operational Guidance

- **Cache Binding:** Validated keys **MUST** be bound to the specific (domain, selector) pair.
- **Revalidation:** Keys **MUST** be re-validated upon TTL expiry.
- **Anomaly Trigger:** A verifier **SHOULD** flush the cache for a specific domain if multiple Trust Packets from that domain fail signature verification in a short window.

8. Error Handling

Code	Condition	Meaning
DS-001	Resolver Inconsistency	Data did not match across N resolvers.
DS-002	Invalid Format	Record found but not a valid UniKey/DKIM format.
DS-003	Algorithm Mismatch	Key found but uses prohibited/weak algorithm.
DS-004	DNSSEC Failure	Cryptographic proof of the DNS record failed.

9. Security Considerations

This algorithm is designed to resist "Man-in-the-Middle" (MitM) attacks at the network layer.

- **Poisoning:** By requiring 3+ independent paths, the cost of a successful poisoning attack is tripled.
- **Privacy:** Verifiers should be aware that querying multiple public resolvers leaks the fact that they are validating a specific domain.

10. IANA Considerations

This document has no IANA actions.

Appendix A: Conformance Invariants

An implementation is conformant with UniKey RFC 3001 if:

1. It never trusts a key derived from a single DNS resolution path.
2. It enforces bitwise equality across all resolver payloads.
3. It fails closed when resolvers provide conflicting data.