Verifier DNS Hardening Algorithm

Status: Draft (Normative)

Purpose: Define the deterministic, step-by-step DNS validation algorithm a UniKey verifier MUST execute to safely discover and trust public keys at internet scale.

1. Design Goals

The DNS hardening algorithm is designed to:

- Detect spoofing, poisoning, and substitution attacks
- Avoid single-resolver or single-path trust
- Remain compatible with global DNS realities
- Fail closed under uncertainty

DNS is treated as a distributed, probabilistic system. Trust emerges through correlation and consistency, not assumption.

2. Inputs

The verifier is provided:

- `domain`
- `selector`
- Expected key algorithm and strength

3. Resolution Algorithm (Normative)

Step 1: Multi-Resolver Query

Query the DKIM DNS record using at least **N ≥ 3** independent resolvers.

Resolvers SHOULD include:

- Local OS resolver
- Public resolver (e.g., ISP)
- Independent third-party resolver

Step 2: Response Normalization

Normalize all responses:

- Remove ordering differences
- Normalize TTLs
- Canonicalize DKIM tag ordering

Step 3: Consistency Check

Compare normalized responses.

Rules:

- Public key material MUST match exactly
- Selector and domain MUST match
- Inconsistent results trigger suspicion

Step 4: TTL & Change Analysis

Evaluate TTL and historical observations:

- Sudden TTL drops increase suspicion
- Unexpected key changes require revalidation

Step 5: Optional DNSSEC Validation

If DNSSEC is present:

- Validate signatures
- Treat failure as signal, not sole decision

DNSSEC success increases confidence but does not eliminate other checks.

Step 6: Trust Decision

Decision matrix:

- Consistent responses → Accept
- Minor anomalies → Heightened scrutiny
- Major inconsistencies → Reject

4. Caching Rules

- Cache only validated keys
- Bind cache to (domain, selector)
- Revalidate on TTL expiry or anomaly

5. Failure Modes

Condition	Action
Resolver inconsistency	Reject
Key format invalid	Reject
Algorithm too weak	Reject
Suspicious change	Revalidate or reject

6. Invariant Enforcement

This algorithm enforces:

- Deterministic verification
- Independent validation
- Infrastructure-level trust decisions

No verifier MAY bypass this process.