DNSSEC Multi-Algorithm Problem Statement

Peter Thomassen <peter@desec.io>

ICANN78 – DNSSEC and Security Workshop
October 25, 2023

Situation

– – –

- There is an **increasing demand for DNSSEC-enabled multi-provider setups**
  - as evidenced by existing support at Cloudflare/deSEC, and implementation work at UltraDNS
- Each signing provider added to trust chain, by referencing its key in DS RRset
- DS RRset governs expectations on presence of RRSIG signatures
  → Providers **choosing different signing algorithms is a spec violation:**

> There MUST be an **RRSIG for each RRset** using at least one DNSKEY **of each algorithm** in the zone apex DNSKEY RRset. The apex DNSKEY RRset itself **MUST be signed by each algorithm appearing in the DS RRset** located at the delegating parent (if any).

RFC 4035 Section 2.2

Complication

– – –

- Providers always using the same algorithm(s) for a zone **not a realistic premise**
  1. Would prevent migrating away from legacy algorithms (e.g., 5→13), unless providers coordinate
  2. Would prevent adding additional algorithms, such as for testing, or to support old+new
     → **Negatively impacts gaining experience / implementation maturity / cryptographic hygiene**
- However: **Violating the spec works fine** in practice (e.g., with algos 8+13)

  → **Expect this to happen sooner or later.** (Perhaps it already has.)

- Ongoing uncertainty → **impediment to predictable multi-provider behavior**
  - and perhaps to multi-provider DNSSEC itself
- **A specification for dealing with this is needed.**
  - Given a multi-algo DS RRset, is there a safe way to skip RRSIGs for some? When, and why?

Use Cases for Forgoing RRSIGs with Advertised Algorithms

1. **Algorithm Rollovers**

   Current specifications effectively **require double-signing** the zone with both old and new algorithm before adding/removing algorithms in DS RRset

   Operational Risks:

   - Larger zone files
   - Larger responses → increased potential for amplification attacks
   - More frequent TCP fallback → higher resource usage and latency
   - For online signers: additional real-time computational overhead

   Double-signing may also introduce financial burden.

Use Cases for Forgoing RRSIGs with Advertised Algorithms

2. **Root Zone Algorithm Rollover**

   Requires updating resolvers' trust anchors. Old algorithm's signatures need to remain in place while resolver operators are adding the new trust anchor

   Operational Risks as described before, for a potentially long time period:

   - Larger zone files/responses; increased potential for amplification attacks
   - More frequent TCP fallback → higher resource usage and latency
   - Financial burden

   Benefit of **avoiding double-signing** described in "Draft Report of the Root Zone DNSSEC Algorithm Rollover Study" → now in Public Comment period

Use Cases for Forgoing RRSIGs with Advertised Algorithms

3. **Permanent Multi-Provider Configurations (for *Redundancy*)**

   Using several independent DNS providers provides resilience against full outage of either
   - Example: github.com uses both Amazon Route 53 and NS1
   - **Very high availability**

   When using DNSSEC, each provider may use their own keys. Even when using distinct signing keys, they can cooperatively serve the same DNS zone.
   - “Multi-signer setup”, RFC 8901 Section 2.1.2

   Under current specifications, these methods are illegal if the providers involved employ different DNSSEC algorithms.

Use Cases for Forgoing RRSIGs with Advertised Algorithms

– – –

4. **Temporary Multi-Provider Configurations (for *Provider Change*)**

   Multi-signer can also be used for transferring a signed zone from one provider to another, each of which uses their own signing keys
   - Without loss of resolution or validation (“non-disruptively”)

   If the providers do not use the same signing algorithms, **such transfer cannot be done legally under current specifications**.

   Note: there’s no reason to assume that both providers support the same set of algorithms
   - In particular when the transfer is due to outdated operations at the former provider

Current Resolver Behavior

> A signed zone MUST include a DNSKEY for each algorithm present in the zone's **DS RRset** and **expected trust anchors** for the zone.
>
> [...] This requirement **applies to servers, not validators**. Validators SHOULD accept any single valid path. They SHOULD NOT insist that all algorithms signaled in the DS RRset work, and they MUST NOT insist that all algorithms signaled in the DNSKEY RRset work.

RFC 6840 Section 5.11

- Colloquially: **SHOULD do logical OR, MUST NOT do logical AND** across paths

Current Resolver Behavior

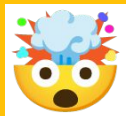
– – –

- Assume a delegation with a DS RRset **advertising two algorithms A and B**
  - ➢ If **resolver supports both algorithms**, sending **just one RRSIG works***
  - ➢ If **resolver supports only algorithm A**, sending **RRSIG only for algorithm B** causes **SERVFAIL**

→ Generally, **not safe to drop signatures** for algorithms advertised in DS RRset
  - Result dependent on the set of algorithms supported by the resolver
  - Authoritative server can't predict that in the general case
    → runs risk of provoking SERVFAIL, unless serving signatures for all advertised algorithms

- Note: **when resolver supports none** of the advertised algorithms
  → zone is **treated as insecure** (no risk of SERVFAIL)

* unless e.g. Unbound's `harden-algo-downgrade` setting is enabled



… what now?

Challenge

– – –

- From a signer's perspective, **just “going ahead” is a reasonable thing to do**
  - For many common cases, serving just one RRSIG will work (e.g., algorithms 8+13 in DS RRset)
- **Things break down** as soon as one of the involved algorithms is no longer supported everywhere (such as with last year’s Red Hat Linux release)
  - When DS RRset advertises algorithms 5 and 8, this should have worked a year ago
  - Today, resolvers on Red Hat do no longer support algorithm 5 (due to a system-wide policy)
  - When receiving a response with an algo-7 signature only, **they will respond with SERVFAIL**
- Why? – The auth cannot reliably predict which algorithms a resolver supports
  - Even harder to make predictions over time.
- Leaving this problem space without guidance **is dangerous**.
  - It’s time to start thinking how to make these use cases work

Solution Approaches

1. **Just loosen the rules, “any RRSIG is fine”**

   By implication, if RRSIG isn't understood, return insecure data (no SERVFAIL)

✅ Serving only one RRSIG for any algorithm advertised via DS works well
- given that advertised algorithms are generally supported

❌ **Trivial downgrade-to-insecure attack** via unsupported algorithms
- Red Hat resolvers: when 5+8 in DS, downgrade is achieved by always setting 5 on the wire
- **Also affects single-provider setup with rare algorithms**, even when the operator serves all!
  - Stalls experiments/early-adopter deployments, severely hindering cryptographic agility
- Any resolver not supporting one of the algorithms can be **caused to return unvalidated data**, despite the other algorithm being supported
  - Attacker just needs to change the RRSIG's algorithm number to the unsupported one
- Auth can’t predict for which algorithms this is safe

Solution Approaches

— — —

2. **Serve all RRSIGs**

   By implication, if RRSIG isn't understood, return insecure data (no SERVFAIL)

✅ Predictable, secure validation behavior

❌ **Not feasible for multi-provider setups** because

- Providers may have different algorithms
- Even if you try to force them to use the same,
  - There will always be some that won't (e.g. **HSM constraint**)
  - Will **stall** all attempts for major providers to transition to future algorithm
  - … unless all introduce at the same time, which does not seem realistic

Solution Approaches

3. **Retry elsewhere when an expected signature is missing**

   If a signature for a supported algorithm is missing (and only unsupported ones are present): **retry other NS**

✅ Accommodates desire for responses with missing RRSIGs
- Provides fallback in case of incompatibility

❌ adds **extra latency**

❌ **renders multi-provider setups useless:**
- If one provider's RRSIG is not supported, **resolution depends on other provider**
- Such **fate sharing is contradictory** to the objectives of a multi-provider setup
  - If that were OK, why have multiple providers?

Solution Approaches

4. **DS hacks to signal RRSIG presence rules**

   Signal RRSIG presence policy directly in the delegation; proposed at ICANN 73

✅ Allows domain owner to say: **“I know what I’m doing, missing RRSIG OK”**
   - On a per-domain basis

❌ **not backward-compatible** (requires all resolvers to upgrade)

❌ **doesn't work with all registries**
   - DNSKEY-style records and compute DS themselves

❌ **can of worms**
   - e.g.: How does this interact with publication of CDS records in multi-provider setups?

Summary

- There is a **need for more fine-grained requirements on RRSIG presence**
- Discussed approaches seem **uncompromising**, and are **not sufficient**:
  - Structural modifications to protocol (DS hack etc.) generally are problematic
  - Otherwise,
    - Retrying is not practical (fate sharing contradicts multi-provider idea)
    - Serving all RRSIGs is not practical (stalls algo transitions, ignores HSM policies)
    - Just relaxing the rules is not practical (insecure)
- Is it conceivable to **design a more measured solution** by combining aspects?
  - E.g.: Relax requirements for widely supported algorithms, keeping tighter rules otherwise?
  - Combine this with a well-considered retry mechanism?
- Or another direction nobody has thought of?

Thank you!

… also to our sponsors:

RIPE NCC
RIPE NETWORK COORDINATION CENTRE

SSE

Questions?