

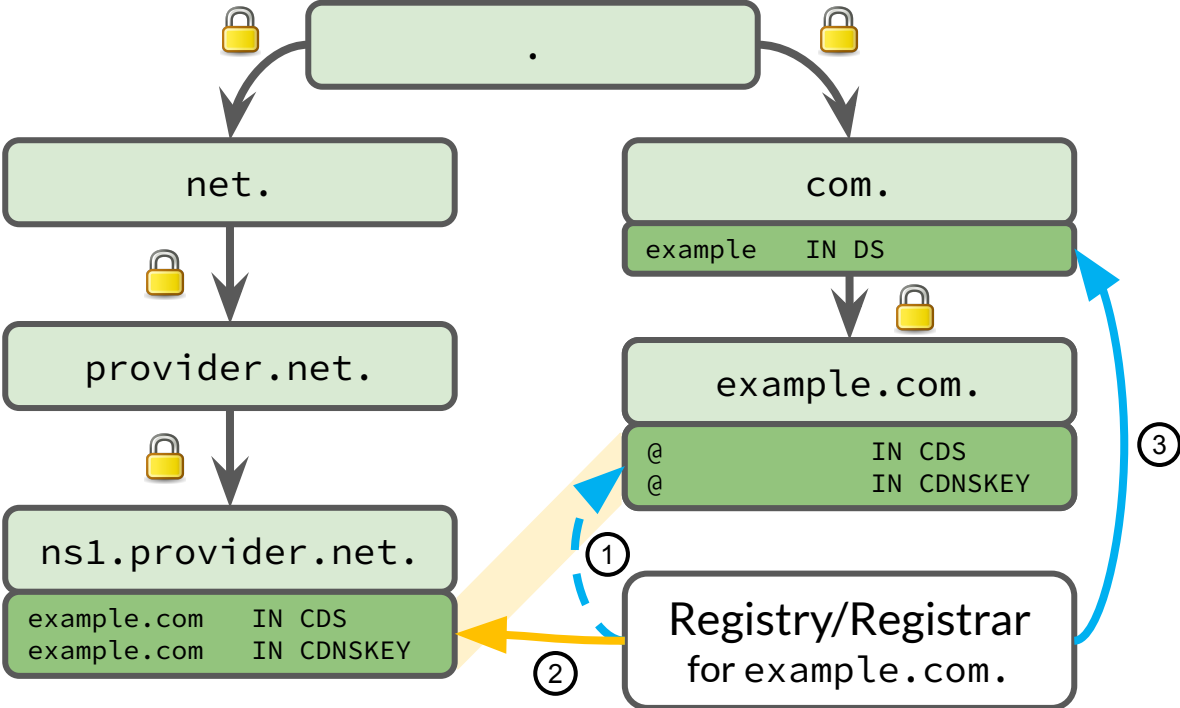
Automatic DNSSEC Bootstrapping using Authenticated Signals from the Zone's Operator

[draft-thomassen-dnsop-dnssec-bootstrapping](#)

ICANN 73 – DNSSEC and Security Workshop
9 March 2022

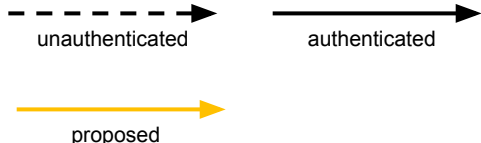
Peter Thomassen (deSEC, Secure Systems Engineering)
[Nils Wisiol](#) (deSEC, Technische Universität Berlin)

CDS Authentication: Co-Publish under Trusted Hostname



💡 Use an established chain of trust (left) to take a detour

- authenticated, immediate
- no active on-wire attacker



Technical Considerations

- No collision with primary use of CDS/CDNSKEY (those are apex-only)
- Add extra label: `example.com._dsboot.ns1.provider.net.`
 - to enable delegation of signaling data to separate zone
 - Update: no hashing of any part of the zone name (enables online generation/signing)
 - allows splitting off DNS operations (e.g. online-signing with different key; delegate by parent)
 - reduces churn on nameserver zone
 - allows discovery of bootstrappable domains using XFR (if allowed)
 - Do you like “_dsboot”?

Is this useful?

Deployment Requirements

- Name server names are in secure zones
- Zone not yet secure

25%

Fulfill requirements in Tranco Top 1M

Current Status

- Implementation
 - Prototype implementation: github.com/desec-io/dsbootstrap
 - CoCCA: implementation underway for 59 ccTLDs
 - GoDaddy: implementation planned after CDS scanning
 - Cloudflare: experimental implementation planned
 - .cl: implementation finished, waiting for internal approval
 - .ch, .cz: interested
- Adoption of [draft](#) by IETF DNS WG in sight
- Post at APNIC Blog to get the word out

Backup

Open Questions

- Should we support sharding, by splitting Signaling Names into several labels?
 - How exactly would that work? Should that be configurable? (How to store configuration?)
- Should the hash(ancestor) label have a PTR record pointing to ancestor?
 - This would allow full enumeration of bootstrappable domains
- For an operator supporting the protocol: is it **REQUIRED** for all domains?
 - Probably no, as it won't work with secondary providers?

Closed Questions (I)

- If a DNS operator deploys DS bootstrapping, parents may like bulk processing. How is that best achieved?
 - allow NSEC walking of signaling zone (thanks to Brian Dickson)
 - allow public AXFR of signaling zone (thanks to John R. Levine)
- Should an extra layer be inserted in the Signaling Name to allow parent-specific bulk processing? (thanks to John R. Levine)
 - Yes
 - compatible with both NSEC walking
 - also compatible with AXFR (but benefit gained only when using subzones for large parents)
- Do we need hash collision mitigation (salt) and/or hash algo upgrade path?
 - No: due to child apex check, collisions don't affect key integrity
 - In case of collision, bootstrapping fails (for this parent) → fallback to conventional DS init
- Do we want hashing at all?
 - No

Closed Questions (II)

- Should the proposal be rephrased as a new mode of operation for RFC 8078?
 - cf. RFC 8078 Section 3.1
 - done
- Drop requirement that all NS responses must agree?
 - No. Otherwise, multihoming with different signers will break the zone.
 - Deployment effort is manageable: 95% of delegations with at least one securely delegated NS target in fact have *all* NS targets securely delegated. Also, dropping this requirement would be inconsistent with requiring records at the child apex to match. It's also unclear what should happen in case of contradictory signaling records, if they are not required to agree.
- Registries/registrars can select which TLDs to trust in the chain. Desirable?
 - No (at least in the spec). One could say that you can't trust a DNS operator anyway if its NS hostnames are not trusted. (That doesn't prevent parents from deciding locally to ignore or reject certain signaling names.)

Discussion Point: Do we want the hashed label?

Do the benefits justify the added complexity?

Pros: ... yes, please, hash please!

- **Helps stay within limits**
 - length / no. of labels → less edge cases
- **Prevents CDS ambiguity** at zone cut
 - What does `foo.bar.net._boot.[...]` mean?
 - It's possible that `bar.net` is not delegated
- **Improves privacy** during discovery
 - must know ancestor to begin NSEC walk
- **Flat structure**
 - simplifies scanning logic
 - facilitates adding prefixes → “properties”
... like: `_cds.example.h(co.uk)._signal.[...]`

Cons: ... no, smash the hash!

- **Complicates implementation**
 - all tooling needs to be able to hash
- **Makes debugging more difficult**
 - standard tools should suffice (dig etc.)
- **Makes synthesis more difficult**
 - How to dynamically associate an incoming query with a target domain?
→ **mapping needed (ancestors only!)**
 - `h(co.uk)._boot` DNAME `co.uk._boot`
(cacheable per parent!)

Securing the example.com delegation (no existing DS)


Assumption: The NS targets (e.g. *ns1.provider.net*) live in securely delegated zones (e.g. *provider.net*).

(I) On the DNS provider side:

Publish example.com's CDS/CDNSKEY records at a “**signaling name**” under the nameserver zone:
example.com.ns1.provider.net

(II) On the registrar / ccTLD registry side:

When receiving a new NS record set,

1. **query** CDS/CDNSKEY records from **DNS provider** (using all NS names):
 - *example.com.ns1.provider.net, ...;*
2. **validate**
 - **DNSSEC signatures** of responses,
 - **sanity check** (consistency with target zone);
3. **publish** example.com's **DS records** in the parent zone → **done!** 

Security Model

- We use an established chain of trust to take a detour
 - authenticated, immediate
 - no active on-wire attacker
- Actors in the chain of trust can undermine the protocol
 - can also undermine CDS / CDNSKEY from insecure
 - but: known point in time / window of opportunity much smaller
- Further mitigations exist, e.g:
 - monitor delegation
 - diversify NS TLDs
 - multiple vantage points

| | BOOTSTRAPPING METHOD | | |
|---------------------------------------|----------------------|------------------------|--------------------------------|
| | MANUAL | CDS/CDNSKEY | PROPOSED |
| BOOTSTRAPPING INVOLVES | | | |
| zone operator Z | ✓ ¹ | ✓ | ✓ |
| domain owner | ✓ | ✗ | ✗ |
| registrar | ✓ | ✗ | ✗ |
| registry | ✓ | ✓ | ✓ |
| ACTORS WHO CAN INITIALIZE KEYS | | | |
| <i>Required parties (trusted)</i> | | | |
| registrar | ✓ | ✓ ² | ✓ ² |
| NS zone operator | ✗ | (✓) | (✓) ³ |
| NS zone ancestors | ✗ | (✓) | (✓) |
| NS zone owner | ✗ | (✓) | (✓) |
| <i>Others parties (untrusted)</i> | | | |
| active on-wire attacker | depends | ✓ ⁴ | ✗ |
| social engineering attacker [1] | ✓ | ✗ | ✗ |
| PROPERTIES | | | |
| Prerequisites | out-of-band channel | MITM attack mitigation | suitable NS zone configuration |
| Authentication | bad in practice [1] | none | cryptographically |
| Duration | varies | days | minutes |

Table 1: Comparison of methods for establishing a new secure delegation, displaying a) entities involved in the bootstrapping of an individual insecure zone, b) attack surface towards trusted and untrusted third parties, and c) prerequisites, key material authentication, and bootstrapping duration. Key initialization within parentheses (✓) requires collusion across all NS zones. ¹ For offline signing, only the signing key holder is involved. ² Registry could refuse deployment through registrar. ³ Requires knowledge of private key. ⁴ Several vantage points and long time must be covered.