Signalling Authoritative DoT support in DS records, with key pinning

Peter van Dijk, Robin Geuze, Manu Bretelle

IETF 108 DPRIVE

July 2020

draft-vandijk-dprive-ds-dot-signal-and-pin-01

Praise

" It is hacky but has very nice properties which are not explicitly described in the document:

- It does not require any extra round-trips to determine if DoT is supported on the auth side.
- Is downgrade-resistant.
- Potentially closes all cleartext leaks (if parent domains are also secured).
- DS "compatibility" makes it deployable in practice.
- CDS/CDNSKEY makes management from auth side easier. "

-- *Petr Špaček*

Other comments

"This is a straight up hack, but it’s tasteful, and I hope it’s got legs."

-- *twitter.com/andrewtj*

"this signalling and verification scheme sounds clever and neat."

-- *Tony Finch*

- "This is protocol abuse"
- "You should be using TLSA"

-- *several people*

History

- DSPKI draft from Manu at IETF 103 DPRIVE
  - no additional roundtrips for a resolver
  - downgrade resistant
  - simple protocol, no indirections
- neat, but not deployable as a new type
- now folded into a DNSKEY/DS deployable form

Resolver protocol

Resolver protocol, step 1

RFC 4034, 2.1. DNSKEY RDATA Wire Format

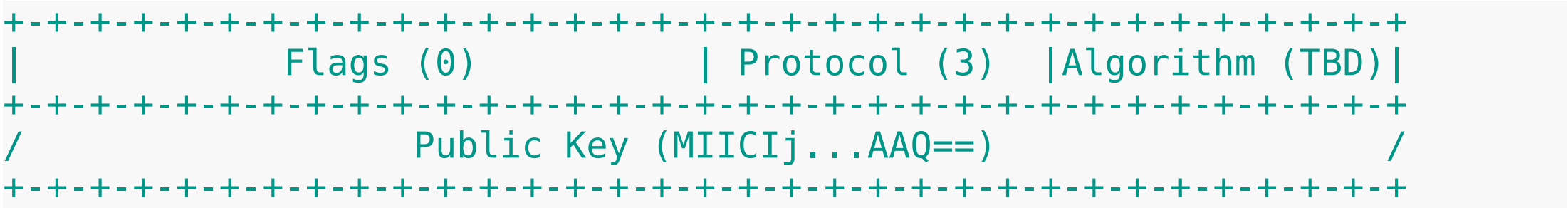
Resolver receives delegation for `example.com.` from `com.`:

```
example.com. NS ns1.example.com.
example.com. NS ns2.example.com.
example.com. DS 7573 TBD 2 fcb6...c26c
```

(TBD: To Be Decided pending IANA assignment)

Resolver protocol, step 2

RFC 4034, 2.1. DNSKEY RDATA Wire Format




```
example.com. NS ns1.example.com.
```

Resolver connects to `ns1.example.com.:853`, negotiates TLS without any key/certificate checking. Resolver receives the auth's pubkey (DER SPKI) and puts it in an in-memory pseudo DNSKEY record. The other 3 DNSKEY fields are filled with constants.

Resolver protocol, step 3

RFC 4034, 2.1. DNSKEY RDATA Wire Format

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Flags (0)           | Protocol (3)  |Algorithm (TBD)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                    Public Key (MIICIj...AAQ==)                    /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
example.com. DS 7573 TBD 2 fcb6...c26c
```

For each DS record in the delegation, the pseudo DNSKEY is hashed with the digest type given by that DS (in this case, with digest 2, SHA256). If we're lucky, that yields a DS we've seen.

Resolver protocol, step 4

RFC 4034, 2.1. DNSKEY RDATA Wire Format

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Flags (0)             | Protocol (3)  |Algorithm (TBD)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                    Public Key (MIICIj...AAQ==)                    /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
example.com. DS 7573 TBD 2 fcb6...c26c
```

If we match a DS in step 3, we are done! We are now securely connected to an authoritative server for `example.com`, with no risk of downgrade. If no DS was matched, we can try another server. If we run out of servers, we are under attack (or misconfiguration) and end up unable to resolve anything inside `example.com.`

Resolver protocol, step 5

RFC 4034, 2.1. DNSKEY RDATA Wire Format

We can start issuing queries, safe in the knowledge that any observer will only know the name (and IP) of the name server we connected to (at least until Encrypted SNI becomes a thing).

Open questions

Q: DNSKEY flags?

- 0 makes semantic sense - these are not ZONE/SEP keys
  - but these are not Zone Signing keys, so those bits mean nothing anyway?
- 257 is suspected to make registry deployment easier
  - need more data on this (both current and future!)

```
+-----------------------------------+
| Number | Description              |
|  7     | ZONE                     |
|  8     | REVOKE                   |
| 15     | Secure Entry Point (SEP) |
+-----------------------------------+
```

Q: What do non-DNSSEC DSes even mean?

- right now, registries say 'DNSSEC: Yes' when you have any DS
- CDS/CDNSKEY 'continuity' rules assume your CDS/CDNSKEY are for DNSSEC
  - current rules in 7344 4.1 also prevent DoT CDS/CDNSKEY for unsigned zones

```
+--------------+-----------------+
| Number       | TBD             |
| Description  | DoT signal+pin  |
| Mnemonic     | DOTPIN          |
| Zone signing | N               |
| Trans sec.   | N               |
| Reference    | RFC TBD2        |
+--------------+-----------------+
```

Implementation status

Text from draft:

- Some Proof of Concept code showing the generation of the (C)DNSKEY, and the subsequent hashing by a client (which should match one of the DS records with algo TBD), in Python and Go, is available at https://github.com/PowerDNS/parent-signals-dot/tree/master/poc

Also, C++ (using OpenSSL's C interface and existing PowerDNS code infrastructure) example code at https://github.com/PowerDNS/pdns/compare/master...Habbie:sdig-dot-pin

Questions, comments?