Workgroup: dprive
Internet-Draft:
draft-vandijk-dprive-ds-dot-signal-and-pin-01
Published: 13 July 2020
Intended Status: Standards Track
Expires: 14 January 2021
Authors: P. van Dijk R. Geuze E. Bretelle
PowerDNS TransIP Facebook

Signalling Authoritative DoT support in DS records, with key pinning

Abstract

This document specifies a way to signal the usage of DoT, and the pinned keys for that DoT usage, in authoritative servers. This signal lives on the parent side of delegations, in DS records. To ensure easy deployment, the signal is defined in terms of (C)DNSKEY.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 14 January 2021.

Table of Contents

1. Introduction

Even quite recently, DNS was a completely unencrypted protocol, with no protection against snooping. In the past few years, this landscape has shifted. The connections between stubs and resolvers are now often protected by DoT, DoH, or other protocols that provide privacy.

This document introduces a way to signal, from the parent side of a delegation, that the name servers hosting the delegated zone support DoT, and with which TLS/X.509 keys. This proposal does not require any changes in authoritative name servers, other than (possibly through an external process) actually offering DoT on port 853 [RFC7858]. DNS registry operators (such as TLD operators) also need to make no changes, unless they filter uploaded DNSKEY/DS records on acceptable DNSKEY algorithms, in which case they would need to add algorithm TBD to that list.

This document was inspired by, and borrows heavily from, [I-D.bretelle-dprive-dot-for-insecure-delegations].

2. Document work

This document lives on GitHub; proposed text and editorial changes are very much welcomed there, but any functional changes should always first be discussed on the IETF DPRIVE WG (dns-privacy) mailing list.

3. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

CDNSKEY record as defined in [RFC7344][RFC8078]

DS record as defined in [RFC4034]

DNSKEY record as defined in [RFC4034]

4. Summary

To enable the signaling of DoT a new DNSKEY algorithm type TBD is added. If a resolver with support for TBD encounters a DS record with the DNSKEY algorithm type TBD it MUST connect to the authoritative servers for this domain via DoT. It MUST use the hashes attached to the DS records with DNSKEY algorithm type TBD to check whether the public key supplied by the authoritative nameserver during the TLS handshake is valid. If the DoT connection is unsuccessful or the public key supplied by the server does not match any of the DS digests, the resolver MUST NOT fall back to unencrypted Do53. For resolvers that are willing to probe for protocol support (DNS over HTTPS, DNS over QUIC), a fallback to other encrypted protocols is allowed if they can satisfy the key pin. This means that if a DS for algo TBD is present, and no name servers satisfy the pin requirement, the response returned to the client is SERVFAIL because no name servers for the domain were available to answer the questions.

A domain MAY have more than one DS record with DNSKEY algorithm TBD. A resolver with support for TBD should then try to verify the public key supplied by the authoritative nameserver against every supplied DS record. Multiple records can be used to support multiple DS digest types, multiple TLS key algorithms, different keys for each authoritative, and for key rollovers. In case of an algorithm or key rollover the new DS record should be added to all served domains before the new key is deployed on the authoritatives. To allow for emergency rollovers, having a standby DS record for all domains with a private key securely stored offline can be a valid strategy.

The pseudo DNSKEY record (when considered in wire format) MUST contain the ([RFC4648] 4.) DER SubjectPublicKeyInfo as defined in [RFC5280] 4.1.2.7. Since the cert provided by the TLS server over the wire is already DER encoded this makes for easy validation. (In the DNSKEY presentation format, the Public Key field contains the Base64 encoding of the DER SPKI, which is equivalent to the SPKI in PEM format minus the header and footer.) The pseudo DNSKEY algorithm type TBD is algorithm agnostic, like the TLSA record, since the DER encoded data already contains information about the used algorithm.

Algorithm support SHOULD be handled at the TLS handshake level, which means a DNS application SHOULD NOT need to be aware of the algorithm used by its TLS library. The pseudo DNSKEY record MUST NOT be present in the zone. The procedure for hashing the pseudo DNSKEY record is the same as for a normal DNSKEY as defined in RFC4034 5.1.4.

As DNSKEY algorithm TBD is not meant to be used for Zone Signing, the existing ZONE and SEP flags do not mean anything. This specification statically defines the flags value as 257 for optimal compatibility with existing registry operations.

The pseudo DNSKEY type can be used in CDNSKEY and CDS (as defined in [RFC7344]) records. These records MAY be present in the zone.

For those familiar with TLSA ([RFC6698]), key matching for this protocol is identical to that provided by TLSA 3 1 0 for (C)DNSKEY. For the DS case, key matching is similar to TLSA 3 1 x where x is not zero, except that the rest of the (C)DNSKEY, including the owner name, gets prepended before hashing.

5. Example

This section will take you through the various parts of this specification, by example.

We assume that we are working with a domain example.com. with one name server, ns.example.com..

5.1. Generating and placing the (C)DNSKEY/DS records

[NOTE: this section uses '225' instead of 'TBD' because otherwise the code does not work. We need to fix this before publication.]

We will walk you through the CDNSKEY/DS generation, demonstrating it in terms of basic shell scripting and some common tools.

First, we extract the SubjectPublicKeyInfo:

```
openssl s_client -connect ns.example.com:853 < /dev/null \
  | openssl x509 -noout -pubkey > pubkey.pem
```

This gives us a file pubkey.pem that looks like this (abridged):

```
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAxH2a6NxIcw5527b04kKy
...
71AWASNoX2GQh7eaQPDD9i8CAwEAAQ==
-----END PUBLIC KEY-----
```

To turns this into a CDNSKEY:

1. remove the header and footer

2. remove all newlines

In other words:

```
openssl s_client -connect ns.example.com:853 </dev/null \
  | openssl x509 -noout -pubkey \
  | sed '1d;$d' \
  | tr -d '\n'
```

Then we prepend

```
example.com. IN CDNSKEY 257 3 225
```

so that we end up with

```
example.com. IN CDNSKEY 257 3 225 MIICIj...AAQ==
```

If your registry accepts CDNSKEY, or DNSKEY via EPP, you are done - you can get your DS placed.

To generate the DS, do something like this:

```
echo example.com. IN DNSKEY 257 3 225 MIICIj...AAQ== \
  | ldns-key2ds -f -n -2 /dev/stdin
example.com.    3600    IN      DS      7573 225 2 fcb6...c26c
```

6. Implementation

The subsection titles in this section attempt to follow the terminology from [RFC8499] in as far as it has suitable terms. 'Implementation' is understood to mean both 'code changes' and 'operational changes' here.

6.1. Authoritative server changes

This specification defines no changes to query processing in authoritative servers.

If DoT-signaling DS records are published for a zone, all name servers for the zone (from both the parent-side and child-side NS RRsets) SHOULD offer DoT service on port 853, and when they do, they SHOULD do so using keys present in the DS RRset. However, there are potential cases where this is not possible, like having multiple DNS providers. In this case the name servers that do not support DoT MUST respond with a RST response or similar on the port tcp/853 to prevent name resolution slowdowns.

6.2. Validating resolver changes

If a resolver successfully uses DoT with a nameserver as specified in this document for one domain, it MAY assume DoT is always available from that nameserver for questions for another domain. However, it

MUST NOT assume that the connection is properly pinned for that other domain unless there is a DS record available for that other domain it is currently resolving.

A validating resolver that supports this draft will perform the following actions when a DS record with algorithm TBD is encountered:

1. Connects to the name server on port 853.

2. During TLS handshake, the resolver will extract the SubjectPublicKeyInfo from the certificate.

3. Construct an in-memory DNSKEY record [RFC4034] section 2 with its fields set as follow:

   *Flags: 257

   *Protocol: 3

   *Algorithm: TBD

   *Public Key: The wire-format SubjectPublicKeyInfo

4. Get the list of Digest Type for DS records obtained from the parent with algorithm TBD

5. For each digest type from the list, compute the DS record of the previously computed DNSKEY, its fields are set as follow:

   *Key Tag: computed from DNS key using [RFC4034] appendix B

   *Algorithm: TBD

   *Digest Type: the current Digest Type we are computing the DS for.

   *Digest: Following [RFC4034] section 5.1.4, compute the digest of owner name | previously computed DNSKEY's RDATA.

6. Test the computed DS record against all the supplied DS records until a match is encountered.

7. If any computed DS record matches a DS record in the DS record set we got from the parent, the connection is successfully authenticated.

6.3. Stub resolver changes

This specification defines no changes to stub resolvers.

6.4. Zone validator changes

This section covers both the 'online' type of zone validator, such as Zonemaster, and the 'offline full zone' type, such as validns and dnssec-verify.

Checks for child DNSKEY records based on parent DS records algorithms, and checks for zone RRSIG algorithms based on DNSKEY algorithms, MUST not be applied to algorithm TBD. [NOTE: rephrase this in terms of the Zone Signing column at https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml ?]

DNSKEY validity checks MAY verify correct DER syntax in DNSKEY Public Key content when algorithm is TBD.

6.5. Domain registry changes

Any pre-delegation or periodic checks by registries should honor the Zone validator changes from the previous section.

This specification trusts that appearance of TBD in https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml will eventually lead registries to accept DS/(C)DNSKEY submissions for algorithm TBD.

Registries that limit the total number of DS records for a delegation SHOULD consider having a separate limit for algorithm TBD DS records, as their management is separate from actual DNSSEC key management.

7. Security Considerations

This document defines a way to convey, authoritatively, that resolvers must use DoT to do their queries to the name servers for a certain zone. By doing so, that exchange gains confidentiality, data integrity, peer entity authentication.

8. Implementation Status

[RFC Editor: please remove this section before publication]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this document, and is based on a proposal described in [RFC6982]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 6982, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

8.1. PoC

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

9. Design Considerations

[RFC Editor: please remove this section before publication]

A protocol design is nothing without a clear statement of the constraints it was designed to meet, and perhaps a list of other constraints it meets by accident.

We humbly acknowledge Petr Spacek's excellent summary of the 'nice properties' this protocol has as a source of inspiration for this section.

Manu's DSPKI proposal had the following excellent properties:

- no extra roundtrips (assuming DSPKI came 'for free' with delegations like DS records do today)
- downgrade resistance
- simple protocol, no indirections

It also had this one very important undesirable property:

- a new RRtype with 'special' behaviour would be pretty much impossible to deploy

In various private and public discussions, it was quickly realised that fitting this into the actual DS record would solve that problem. The first obvious answer to that is 'just assign some numbers and do in DS what DSPKI defined in its own type'. Petr Spacek and others pointed out that this would be incompatible with 'DNSKEY-style' registries, i.e. those that demand DNSKEY, not DS, in their communications (those communications being either EPP, some registry-specific protocol, or CDNSKEY). In other words, a protocol that would not allow the DS to be hashed 'the usual way' from a DNSKEY would not go far, as many registries are slow to update their software even just for a couple of new numbers in an IANA registry.

With that, the puzzle was clear. We need some format to signal and pin DoT with a DNSKEY, in such a way that a DS can be hashed from it without software changes in parties such as registries, and such that that DS is enough for a resolver to validate a TLS connection.

Eventually we realised that a resolver could take the TLS SubjectPublicKeyInfo, construct a 'pseudo' DNSKEY from it, and hash that into a DS. This resolves the one bad property of DSPKI (deployability without changing every auth, resolver, and registry stack in the world).

The design constraints we felt we must meet with this protocol were:

- *deployability without demanding massive software changes or even 'flag days'

- *downgrade resistance

And we feel we have met those. The other positive properties of DSPKI (simplicity, no extra roundtrips) have been kept intact more by accident than by strong intention.

We can understand that several people are saying that this is hacky (we do not even disagree), and that TLSA should have been used. However, we feel that any TLSA-based protocol we can imagine would be a lot more complex, and therefore prone to breakage which might be hard to debug. It would also be very easy to accidentally introduce chicken-and-egg problems with a more indirect approach. Note that we are responding to imagined TLSA-based protocols here. If a draft appears for a TLSA-based approach to DoT signaling/pinning, we would love to read it. Depending on what that draft looks like, it might even make sense to have that protocol and the protocol described in this document.

The biggest downside to this DS-based protocol is that a change in TLS keys on an auth may require DS updates for thousands or even hundreds of thousands of domains. This issue is partially mitigated by allowing backup keys to be part of those DS sets. Furthermore we hope that efforts from Cloudflare and others for shortening the path between auth operator and domain registrar one day work out. Those efforts are focused on NSset updates and DS updates for DNSSEC validation, but they would also aid key rollovers for this protocol greatly.

10. IANA Considerations

This document updates the IANA registry "DNS Security Algorithm Numbers" at https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml

The following entries have been added to the registry:

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

11. Acknowledgements

The authors would like to thank the following individuals for their useful input: Job Snijders, Maik Zumstrull, Petr Spacek, Pieter Lexis, Ralph Dolmans, Remi Gacogne, Seth Arnold, and Vladimir Cunat.

12. Normative References

[I-D.bretelle-dprive-dot-for-insecure-delegations] Bretelle, E., "DNS-over-TLS for insecure delegations", Work in Progress, Internet-Draft, draft-bretelle-dprive-dot-for-insecure-delegations-01, 11 March 2019, <https://tools.ietf.org/html/draft-bretelle-dprive-dot-for-insecure-delegations-01>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <https://www.rfc-editor.org/info/rfc4034>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC6982] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", RFC 6982, DOI 10.17487/RFC6982, July 2013, <https://www.rfc-editor.org/info/rfc6982>.

[RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <https://www.rfc-editor.org/info/rfc7858>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7344] Kumari, W., Gudmundsson, O., and G. Barwood, "Automating DNSSEC Delegation Trust Maintenance", RFC 7344, DOI 10.17487/RFC7344, September 2014, <https://www.rfc-editor.org/info/rfc7344>.

[RFC8078] Gudmundsson, O. and P. Wouters, "Managing DS Records from the Parent via CDS/CDNSKEY", RFC 8078, DOI 10.17487/RFC8078, March 2017, <https://www.rfc-editor.org/info/rfc8078>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

13. Informative References

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <https://www.rfc-editor.org/info/rfc6698>.

[RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <https://www.rfc-editor.org/info/rfc8499>.

Appendix A. Document history

A.1. Changes between -00 and -01

1. Lots of clarifying text that does not change any semantics, including:

   *a section on how resolvers would actually use this protocol.

   *we made it clearer that multiple DS records for a delegation are allowed, and why you would want this.

2. DNSKEY flags are now set to 257, because it looks like this will make it a lot easier for many registries to accept the records.

3. Added a 'Design Considerations' section to give some background to why this protocol is what it is.

We have tried to do a review of this protocol against the requirement of the DPRIVE phase 2 document. You can find this review (which might be updated outside of revisions of this draft or the phase 2 draft) in our GitHub repo.

Authors' Addresses

Peter van Dijk
PowerDNS
Den Haag
Netherlands

Email: peter.van.dijk@powerdns.com

Robin Geuze
TransIP
Delft
Netherlands

Email: robing@transip.nl

Emmanuel Bretelle
Facebook

Email: chantra@fb.com