Network Working Group
Internet-Draft
Intended status: Informational
Expires: 13 May 2022

B. Dickson
GoDaddy

9 November 2021

Authenticated DNS over TLS to Authoritative Servers
draft-dickson-dprive-adot-auth-06

Abstract

This Internet Draft proposes a mechanism for DNS resolvers to discover support for TLS transport to authoritative DNS servers, to validate this indication of support, and to authenticate the TLS certificates involved.

This requires that the name server _names_ are in a DNSSEC signed zone.

This also requires that the delegation of the zone served is protected by [I-D.dickson-dnsop-ds-hack], since the NS names are the keys used for discovery of TLS transport support.

Additional recommendations relate to use of various techniques for efficiency and scalability, and new EDNS options to minimize round trips and for signaling between clients and resolvers.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Domain Name System (DNS) predates any concerns over privacy, including the possibility of pervasive surveillance. The original transports for DNS were UDP and TCP, unencrypted. Additionally, DNS did not originally have any form of data integrity protection, including against active on-path attackers.

DNSSEC (DNS Security extensions) added data integrity protection, but did not address privacy concerns. The original DNS over TLS [RFC7858] and DNS over HTTPS [RFC8484] specifications were limited to client-to-resolver traffic.

The remaining privacy component is recursive-to-authoritative servers. This Internet Draft is designed to provide a solution to this problem.

2. 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.

3. Background

The result is that the parental side of the zone cut has records needed for DNS resolution which are not signed and not validatable.

4. Purpose

Authoritative DNS over TLS is intended to provide the following for communications from recursive resolvers to authoritative servers:

* Enable discovery of support for ADoT

* Validate the name server name

* Validate the server’s TLS certificate

* Provide channel security using TLS

4.1. New DNS Elements

The following are new protocol components, which are either included in this document, or are in other documents. Some are strictly required, while others are strongly suggested components to allow better scalability and performance. Some of the new elements are aliases to already documented standards, for purposes of these improvements. DNST refers to [I-D.dickson-dprive-dnst]

Element	New/Alias/ OPT	Format/ Base	Required	Description
DNST	New	Flags	Y	DNS Transport - support for DoT
TLSADOT	Alias	TLSA	Y	TLSA without prefixing
ADOTD	New	OPT RR (flag)	N	Signal desire for ADOT (client-resolver)
ADOTA	New	OPT RR (flag)	N	Signal availablity of ADOT (resolver-client)
NSECD	New	OPT RR (flag)	N	Signal desire for NSEC(3) for [RFC8198]
NSV	New	DNSKEY Alg	Y	Protect NS - see [I-D.dickson-dnsop-ds-hack]

Table 1

5. Requirements, and Limitations

This protocol depends on correct configuration and operation of the respective components, and that those are maintained according to Best Current Practices:

* Use of DS records [I-D.dickson-dnsop-ds-hack] for the protection of the delegation to the authoritative name servers

* Use of "glueless" zone(s) for name server names’ zone [I-D.dickson-dnsop-glueless]

* DNSSEC signing of the zone serving the authoritative name servers’ names [@RFC4034;@RFC4035;RFC5155]

- Proper management of key signing material for DNSSEC

- Ongoing management of RRSIGs on a timely basis (avoiding RRSIG expiry)

- Ensuring TLSADOT records are kept synchronized with the TLS certificates used

- Proper management of TLS private keys for TLS certificates used

There are external dependencies that impact the system security of any DNSSEC zone, which are inherently unavoidable in establishing this scheme. Specifially, the original DS record enrollment and any updates to the DS records involved in DNSSEC delegations are presumed secure and outside of the scope of the DNS protocol per se.

Other risks relate to normal information security practices, including access controls, role based access, audits, multi-factor authentication, multi-party controls, etc. These are out of scope for this protocol itself.

6. DNS Records To Publish for ADoT

ADoT is a property of DNS servers. The signaling is done at the server level, using a DNS record with the same owner name as the server itself (i.e. where the A and AAAA records for the server are published).

6.1. Server DNS Transport Support Signaling

In order to support ADoT for a DNS server, it is necessary to publish a record specifyig explicit DoT support. This record also indicates other supported transports for the DNS server, e.g. the standard ports (TCP and UDP port 53).

The record type is "DNST" (DNS Transport), which is a single resource record consisting of flags for different supported transport types.

The zone serving the record MUST be DNSSEC signed. The absence of the DNST RRTYPE is proved by the NSEC(3) record, or else the DNST RRTYPE plus RRSIG is returned in response to a query for this record if it exists.

6.1.1. Examples

Suppose the name server ns1.example.net supports only the normal DNS ports, and the name server ns2.example.net supports both the normal ports and ADoT. The zone example.net would include the records:

```
    ns1.example.net. IN DNST UDP TCP
    ns2.example.net. IN DNST UDP TCP DOT
```

And similarly, if another zone with many name server names wanted to have a policy of all-ADoT support (i.e. every name server supports ADoT), they would each be encoded as:

```
    ns1.example2.net DNST UDP TCP DOT
    ns2.example2.net DNST UDP TCP DOT
    ns3.example2.net DNST UDP TCP DOT
    ns4.example2.net DNST UDP TCP DOT
```

6.2. DANE TLSA Records for ADoT (TLSADOT)

The presence of ADoT requires additionally that a TLSA [RFC6698] record be provided. A new RRTYPE is to be created for this as an alias of TLSA, with mnemonic of "TLSADOT" (TLS ADOT Certificate). This record will be published at the location NS_NAME, where NS_NAME is the name of the name server. Any valid TLSA RDATA is permitted. The use of Certificate Usage types PKIX-TA and PKIX-EE is NOT RECOMMENDED since PKIX requires web PKI interactions. DANE types only require DNSSEC support. The use of Certificate Usage types DANE-TA records may provide more flexibility in provisioning and validation. On the other hand, DANE-EE is more secure, with fewer consequences for private key loss and certificate revocation. Per [RFC7218][RFC7671] the RECOMMENDED Selector and Matching types for this are SPKI and SHA2-256, giving the recommended TLSADOT record type of DANE-TA SPKI SHA2-256.

6.2.1. Example

In the above example, ns2.example.net supports DNS over TLS, and thus would need to have a TLSADOT record. The zone would include:

```
    ns2.example.net. IN TLSADOT DANE-TA SPKI SHA2-256 (hash data)
```

If there were another zone containing many DNS server names, example2.net, it would be relatively simple to replicate otherwise identical records which use the same signing cert (rather than end-entity cert) in the TLSADOT record.

This would look like the following:

```
       ns1.example2.net IN TLSADOT DANE-TA SPKI SHA2-256 (hash data)
       ns2.example2.net IN TLSADOT DANE-TA SPKI SHA2-256 (hash data)
       ns3.example2.net IN TLSADOT DANE-TA SPKI SHA2-256 (hash data)
       ns4.example2.net IN TLSADOT DANE-TA SPKI SHA2-256 (hash data)
       ns1.example2.net IN A IP4_ADDRESS1
       ns2.example2.net IN A IP4_ADDRESS2
       ns3.example2.net IN A IP4_ADDRESS3
       ns4.example2.net IN A IP4_ADDRESS4
       ns1.example2.net IN AAAA IP6_ADDRESS1
       ns2.example2.net IN AAAA IP6_ADDRESS2
       ns3.example2.net IN AAAA IP6_ADDRESS3
       ns4.example2.net IN AAAA IP6_ADDRESS4
```

6.3. Signaling DNS Transport for a Name Server

This transport signaling MUST only be trusted if the name server names for the domain containing the relevant name servers’ names are protected with [I-D.dickson-dnsop-ds-hack] and validated. The name servers must also be in a DNSSEC signed zone (i.e. securely delegated where the delegation has been successfully DNSSEC validated).

The specific DNS transport that a name server supports is indicated via use of an RRSet of RRTYPE "DNST".

6.3.1. Examples

We re-use the same examples from above, indicating whether or not individual authoritative name servers support DoT:

```
       ns1.example.net. IN DNST UDP TCP DOTDNST
       ns2.example.net. IN DNST UDP TCP DOTDNST
```

And similarly, if another zone with many name server names wanted to have a policy of all-ADoT support (i.e. every name server supports ADoT), this could be encoded as:

```
       ns1.example2.net DNST UDP TCP DOT
       ns2.example2.net DNST UDP TCP DOT
       ns3.example2.net DNST UDP TCP DOT
       ns4.example2.net DNST UDP TCP DOT
```

6.4. Signaling DNS Transport for a Domain

A domain inherits the signaled transport for the name servers serving the domain.

This transport signaling MUST only be trusted for use of ADoT if the delegated name server names for the domain are protected with [I-D.dickson-dnsop-ds-hack].

The delegation to NS names "A" and "B", along with the DS record protecting/encoding "A" and "B", results in the DNS transport that is signaled for "A" and "B" being applied to the domain being delegated. This transport will include ADoT IFF the transport for "A" and "B" has included ADoT via DNS records.

6.4.1. Examples

No additional configuration is needed, beyond use of authority servers which signal DoT support. The following examples assumes the previous DNS records are provisioned:

```
example.com NS ns1.example.net. // does not support ADoT
example.com NS ns2.example.net. // supports ADoT

example2.com NS ns1.example2.net. // all support ADoT
example2.com NS ns2.example2.net. // all support ADoT
```

In this example, ns1 does not have ADoT support (since the DNST record excludes the DOT flag), while ns2 does support ADoT (since it includes DOT).

7. Validation Using DS Records, DNST Records, TLSADOT Records, and DNSSEC Validation

These records are used to validate corresponding delegation records, DNST records, and TLSADOT records, as follows:

* Initial domain NS records are validated using [I-D.dickson-dnsop-ds-hack]

* All DS records implementing [I-D.dickson-dnsop-ds-hack] must be DNSSEC validated prior to use

* Once the NS names have been validated, and the delegations to the appropriate name servers are validated, the DNST records for the NS name are obtained to identify the DNS transport methods supported.

* If ADoT is among the supported transports, the TLSADOT record for the name server is obtained, and used for verification of the TLS certificate when making the TLS connection.

7.1. Complete Example

7.1.1. DNS Record Data

Suppose a client requests resolution for the IP address of "sensitive-name.example.com". Suppose the client's resolver has a "cold" cache without any entries beyond the standard Root Zone and relevant TLD name server records.

Suppose the following entries are present at their respective TLD authority servers, delegating to the respective authority servers:

```
    // (Single NS for brevity only, please use 2 NS minimum )
    // Unsigned delegations to various single-operator servers
    example2.com NS ns1.example2.net. // all support ADoT
    example3.com NS ns2.example2.net. // all support ADoT
    example4.com NS ns3.example2.net. // all support ADoT
    example5.com NS ns4.example2.net. // all support ADoT

    // Zone serving NS data for single-operator's servers
    example2.net NS ns1.infra2.example
    example2.net NS ns2.infra2.example
    example2.net DS (DS record data)
    // glueless name servers are used

    // Special zone serving NS data for previous zone
    infra2.example NS ns1-glue.infra2.example
    infra2.example NS ns2-glue.infra2.example
    infra2.example DS (DS record data)
    // Note use of glue for only this zone's delegation
    ns1-glue.infra2.example A (glue A data)
    ns1-glue.infra2.example AAAA (glue AAAA data)
    ns2-glue.infra2.example A (glue A data)
    ns2-glue.infra2.example AAAA (glue AAAA data)
```

Suppose the following additional entries are in the respective authority servers for the ADOT signaling/certs:

```
       example2.net SOA ( SOA record data )
       // glueless name servers are used
       example2.net NS ns1.infra2.example
       example2.net NS ns2.infra2.example
       //
       // DNS Transport for discovery of support
       ns1.example2.net DNST UDP TCP DOT
       ns2.example2.net DNST UDP TCP DOT
       ns3.example2.net DNST UDP TCP DOT
       ns4.example2.net DNST UDP TCP DOT
       //
       // TLSADOT signing cert
       ns1.example2.net IN TLSADOT DANE-TA SPKI SHA2-256 (hash data)
       ns2.example2.net IN TLSADOT DANE-TA SPKI SHA2-256 (hash data)
       ns3.example2.net IN TLSADOT DANE-TA SPKI SHA2-256 (hash data)
       ns4.example2.net IN TLSADOT DANE-TA SPKI SHA2-256 (hash data)
       //
       // Addresses of name servers serving customer zones
       // E.g. example2.com to example5.com served on these
       ns1.example2.net IN A IP4_ADDRESS1
       ns2.example2.net IN A IP4_ADDRESS2
       ns3.example2.net IN A IP4_ADDRESS3
       ns4.example2.net IN A IP4_ADDRESS4
       ns1.example2.net IN AAAA IP6_ADDRESS1
       ns2.example2.net IN AAAA IP6_ADDRESS2
       ns3.example2.net IN AAAA IP6_ADDRESS3
       ns4.example2.net IN AAAA IP6_ADDRESS4
       //
       // plus RRSIGs and NSEC(3) records and their RRSIGs

       infra2.example SOA ( SOA record data )
       infra2.example NS ns1-glue.infra2.example
       infra2.example NS ns2-glue.infra2.example
       ns1-glue.infra2.example A (same as glue A data)
       ns1-glue.infra2.example AAAA (same as glue AAAA data)
       ns2-glue.infra2.example A (same as glue A data)
       ns2-glue.infra2.example AAAA (same as glue AAAA data)
       //
       //  name server info for example2.net zone
       ns1.infra2.example A (glueless A data)
       ns1.infra2.example AAAA (glueless AAAA data)
       ns2.infra2.example A (glueless A data)
       ns2.infra2.example AAAA (glueless AAAA data)
       //
       // plus RRSIGs and NSEC(3) records and their RRSIGs
```

7.1.2. Discussion Point - Wildcard-like Records

Wildcards records have RRTYPE(s), but are only instantiated when an owner name does not exist.

If wildcards were instantiated whenever the 3-tuple (name, class, type) did not exist, use of wildcard records for DNST and TLSADOT would be a logical choice.

The discussion point is as follows:

* Would it make sense to support a wildcard-like behavior for covering many owner names which did not have explicit DNST and/or TLSADOT records of their own?

* If so, when/how would that be signalled?

  - It could be explicit, using a separate RRTYPE to flag the need to use the parent name (zone apex) for the required RRTYPE.

    o This would support use of NSEC(3) records to check for the flag

    o A resolver could use the flag to optimize cache usage for the parent record. Once the parent is in the cache, the flag in the NSEC(3) for the owner name would trigger use of the cached parent record.

  - It could be implicit, meaning the absence of the explicit record type results in the need to search for the record type at another name (e.g. zone apex).

    o The lack of explicit record could be detected from NSEC(3) records

    o The implicit flag would be handled the same as the explicit flag case above.

* The TLSADOT record at the parent zone would only be viable for DANE-TA or PKIX-TA types.

7.1.3. Resolver Iterative Queries For Final TLS Query

(In the following, use of wildcard-type records and semantics is assumed, but not explictly described currently. Literal wildcard record labels ("*") are used as a placeholder, pending the above Discussion Point’s resolution.)

The following are the necessary queries to various servers necessary to do a private TLS-protected lookup.

Several examples are provided in order, from a presumed cold cache state. Root Priming and TLD queries are presumed to already have been complete.

1. Query for sensitive-name.example2.com:

    1. Query for NS for example2.com => get NS ns1.example2.net plus DS => validate the DS and proceed

    2. Query for NS for example2.net => get NS ns1/ ns2.infra2.example plus DS => validate the DS and proceed

    3. Query for NS for infra2.example2.net => get NS ns1-glue/ ns2-glue.infra2.example plus DS plus glue A/AAAA => validate the DS and proceed

    4. Query with NSECD for A for ns1/ns2.infra2.example => get A for ns1/ns2.infra2.example plus RRSIGs plus NSEC(3) plus RRSIG => validate the RRSIGs and proceed

    5. Query with NSECD for A for ns1.example2.net => get A for ns1.example2.net plus RRSIG plus NSEC(3) plus RRSIG => validate the RRSIGs and proceed

    6. Query with NSECD for DNST for ns1.example2.net => get DNST for *.example2.net plus RRSIG plus special wildcard NSEC(3)s plus RRSIGs => validate the RRSIGs and proceed

    7. Query with NSECD for TLSADOT for ns1.example2.net => get TLSADOT for *.example2.net plus RRSIG plus special wildcard NSEC(3)s plus RRSIGs => validate the RRSIGs and proceed

    8. Query over TLS for sensitive-name.example2.com (to ns1.example2.net, match TLS cert chain against DANE-TA cert, only query once TLS established)

2. Query for sensitive-name.example3.com:

    1. Query for NS for example2.com => get NS ns1.example2.net plus DS => validate the DS and proceed

    2. Query with NSECD for A for ns1.example2.net => get A for ns1.example2.net plus RRSIG plus NSEC(3) plus RRSIG => validate the RRSIGs and proceed

   3. NB: already have wildcards for DNST and TLSADOT plus NSEC3 proving no non-wildcards exist for ns1.example2.net for those types, synthesize DNST and TLSADOT records)

   4. Query over TLS for sensitive-name.example2.com (to ns1.example2.net, match TLS cert chain against DANE-TA cert, only query once TLS established)

3. Query for sensitive-name.example4.com:

   1. Query for NS for example2.com => get NS ns1.example2.net plus DS => validate the DS and proceed

   2. Query with NSECD for A for ns1.example2.net => get A for ns1.example2.net plus RRSIG plus NSEC(3) plus RRSIG => validate the RRSIGs and proceed

   3. NB: already have wildcards for DNST and TLSADOT plus NSEC3 proving no non-wildcards exist for ns1.example2.net for those types, synthesize DNST and TLSADOT records)

   4. Query over TLS for sensitive-name.example2.com (to ns1.example2.net, match TLS cert chain against DANE-TA cert, only query once TLS established)

4. Query for sensitive-name.example5.com:

   1. Query for NS for example2.com => get NS ns1.example2.net plus DS => validate the DS and proceed

   2. Query with NSECD for A for ns1.example2.net => get A for ns1.example2.net plus RRSIG plus NSEC(3) plus RRSIG => validate the RRSIGs and proceed

   3. NB: already have wildcards for DNST and TLSADOT plus NSEC3 proving no non-wildcards exist for ns1.example2.net for those types, synthesize DNST and TLSADOT records)

   4. Query over TLS for sensitive-name.example2.com (to ns1.example2.net, match TLS cert chain against DANE-TA cert, only query once TLS established)

5. Query for sensitive-name2.example2.com:

   1. (Already have delegation entry for example2.com in cache.)

   2. (Already have A for ns1.example2.net in cache.)

3. (Already have all TLS info in the cache.)

4. Query over TLS for sensitive-name.example2.com (to ns1.example2.net, match TLS cert chain against DANE-TA cert, only query once TLS established)

Once the initial query or queries for a name server zone has been done, if that zone uses wildcards for DNST and TLSADOT, the only queries needed for a new name server are the A and/or AAAA records. Once the initial query for a name server has been done, all of the address and TLS information is available in the cache, and the DOT query can be made upon receipt of the TLD delegation record. Once the initial query for a second-level domain has been done, the TLD delegation and all of the address and TLS information is available in the cache, and the DOT query can be made immediately.

Once a cache is populated with wildcards from the name server domain, additional delegation queries require no more trips than those needed for normal UDP queries:

1. Query for delegation from TLD, and validate the response

2. Query for the name server’s address(es), and validate the response

3. Send the query to the authoritative server for the domain with the sensitive name (over TLS or over UDP/TCP, depending on transport supported by the authoritative server)

Once a cache is populated with name server addresses and wildcards, additional delegation queries require no more trips than those needed for normal UDP queries:

1. Query for delegation from TLD, and validate the response

2. Send the query to the authoritative server for the domain with the sensitive name (over TLS or over UDP/TCP, depending on transport supported by the authoritative server)

8. Signaling Resolver Support and Client Desire for ADoT

The following presume some new OPT sub-types, to be added to the IANA action section or to be split out as separate drafts. The sub-type mnemonics are "ADOTA" (available) and "ADOTD" (desired), each with an enumerated set of values and mnemonic codes. Respectively those are: "Always", "Upon Request", and "Never"; and "Force", "If Available", and "Never".

8.1. Server Side Support Signaling

A DNS server (e.g. recursive resolver or forwarder) MAY signal to clients that it offers the use of ADoT. The mechanism used is to set the EDNS option "ADOTA". The values for this option are "Always", "Upon Request", and "Never". The value "Always" indicates the server will always attempt to use ADoT without regards to client requests for ADoT. The value "Upon Request" indicates that the server will ONLY use ADoT for upstream queries if the client requests that ADoT be used. These values have no effect on answers served from the resolver’s cache. (The "Never" case is unusual, in that it signals the server understands the option, but does not perform ADoT. Generally this would be used to allow a client to track changes in the status, if the client is interested in uses of ADoT.)

8.2. Client Side Desire Signaling

A DNS client (e.g. stub or forwarder) MAY signal the desire to have the resolver use ADoT. The mechanism used is to set the EDNS option "ADOTD". The values for this option are "Force", "If Available", and "Never". The value "Force" indicates the server should attempt to use ADoT, and return a failure code of XXXX and an EDE value of YYYY if the authoritative server does not offer ADoT, or any other ADoT failure occurs. The value "If Available" indicates that the server should use ADoT for upstream queries if it is availble, but SHOULD NOT allow any downgrades if the authoritative server signals that ADoT is available. These values have no effect on answers served from the resolver’s cache. (The "Never" case is unusual, in that it signals the client understands the option, but does not perform ADoT. Generally this would be used to allow a server to track changes in the client base, so the server operator can make informed decisions about enabling ADoT.)

9. Security Considerations

As outlined above, there could be security issues in various use cases.

10. IANA Considerations

This document may or many not have any IANA actions. (e.g. if the RRTYPEs, EDNS subtypes, DNSKEY algorithms, etc., are defined in other documents, no IANA actions are needed.)

11. Normative References

[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>.

[RFC7218] Gudmundsson, O., "Adding Acronyms to Simplify Conversations about DNS-Based Authentication of Named Entities (DANE)", RFC 7218, DOI 10.17487/RFC7218, April 2014, <https://www.rfc-editor.org/info/rfc7218>.

[RFC7671] Dukhovni, V. and W. Hardaker, "The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance", RFC 7671, DOI 10.17487/RFC7671, October 2015, <https://www.rfc-editor.org/info/rfc7671>.

[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>.

[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>.

[RFC8198] Fujiwara, K., Kato, A., and W. Kumari, "Aggressive Use of DNSSEC-Validated Cache", RFC 8198, DOI 10.17487/RFC8198, July 2017, <https://www.rfc-editor.org/info/rfc8198>.

[RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://www.rfc-editor.org/info/rfc8484>.

12. Informative References

[I-D.dickson-dnsop-ds-hack]
Dickson, B., "DS Algorithms for Securing NS and Glue", Work in Progress, Internet-Draft, draft-dickson-dnsop-ds-hack-02, 19 September 2021, <https://datatracker.ietf.org/doc/html/draft-dickson-dnsop-ds-hack-02>.

[I-D.dickson-dnsop-glueless]
Dickson, B., "Operating a Glueless DNS Authority Server", Work in Progress, Internet-Draft, draft-dickson-dnsop-glueless-02, 22 September 2021, <https://datatracker.ietf.org/doc/html/draft-dickson-dnsop-glueless-02>.

[I-D.dickson-dprive-dnst]
Dickson, B., "Resource Record for Signaling Transport for DNS to Authority Servers", Work in Progress, Internet-Draft, draft-dickson-dprive-dnst-00, 24 October 2021, <https://datatracker.ietf.org/doc/html/draft-dickson-dprive-dnst-00>.

[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>.

Appendix A. Acknowledgments

Thanks to everyone who helped create the tools that let everyone use Markdown to create Internet Drafts, and the RFC Editor for xml2rfc.

Thanks to Dan York for his Tutorial on using Markdown (specificially mmark) for writing IETF drafts.

Thanks to YOUR NAME HERE for contributions, reviews, etc.

Author’s Address

Brian Dickson
GoDaddy

Email: brian.peter.dickson@gmail.com