

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: 23 March 2022

B. Dickson  
GoDaddy  
19 September 2021

DS Algorithms for Securing NS and Glue  
draft-dickson-dnsop-ds-hack-02

Abstract

This Internet Draft proposes a mechanism to encode relevant data for NS records on the parental side of a zone cut by encoding them in DS records based on a new DNSKEY algorithm.

Since DS records are signed by the parent, this creates a method for validation of the otherwise unsigned delegation records.

Notably, support for updating DS records in a parent zone is already present (by necessity) in the Registry-Registrar-Registrant (RRR) provisioning system, EPP. Thus, no changes to the EPP protocol are needed, and no changes to registry database or publication systems upstream of the DNS zones published by top level domains (TLDs).

This NS validation mechanism is beneficial if the name server `_names_` need to be validated prior to use.

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 23 March 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Conventions and Definitions . . . . .	3
3. Background . . . . .	3
3.1. Attack Example . . . . .	3
4. New DNSKEY Algorithm . . . . .	4
4.1. Algorithm {TBD1} . . . . .	4
4.1.1. Example . . . . .	4
5. Validation Using These DS Records . . . . .	5
6. Protection of glue records . . . . .	5
7. Security Considerations . . . . .	5
8. IANA Considerations . . . . .	5
9. Normative References . . . . .	5
10. Informative References . . . . .	6
Appendix A. Acknowledgments . . . . .	6
Author's Address . . . . .	6

## 1. Introduction

Currently, any query for delegation NS records over an unprotected transport path returns results which do not have protection from tampering by an active on-path attacker, or against successful cache poisoning attacks. This is because the parent NS records are being authoritative, and thus do not have RRSIGs. The child NS records with the same owner name are authoritative, but the parent NS records are what get used for delegations.

There is new privacy work that relies on the name server names in the delegation RDATA. Unsigned records are vulnerable to modification by on-path attackers and to cache poisoning by off-path attackers. That privacy work uses the name for TLS validation, and the only source of the name server name is the NS record in the delegation.

This document is about protecting the RDATA of NS record, not the privacy issues per se.

Note that the use of an encrypted transport (such as DoT [RFC7858]) to the parent would be an alternative approach, but in the absence of encrypted transport, the current approach is recommended.

If an attacker alters the NS records returned, or poisons the resolver's cache for the unsigned delegation NS, the recursive resolver could be directed to a server operated by an attacker.

## 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 methods developed for adding security to the Domain Name System, collectively referred to as DNSSEC, had as a primary requirement that they be backward compatible. The original specifications for DNS used the same Resource Record Type (RRTYPE) on both the parent and child side of a zone cut (the NS record). The main goal of DNSSEC was to ensure data integrity by using cryptographic signatures. However, owing to this overlap in the NS record type where the records above and below the zone cut have the same owner name created an inherent conflict, as only the child zone is authoritative for these records.

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

This has no security (data validation) impact on DNS zones which are fully DNSSEC signed (anchored at the IANA DNS Trust Anchor), but does impact unsigned zones regardless of where the transition from secure to insecure occurs.

### 3.1. Attack Example

Suppose a resolver queries for the NS records for "example.com", at the name servers for the "com" TLD. Suppose this domain has been published in the com zone as "example.com NS ns1.example.net".

The response is not protected against MITM attacks. An on-path attacker can substitute its own name, "ns1.attacker.example". The resolver would then send its queries to the attacker.

Note that this vulnerability to MITM is present even if the domain "example.net" (the domain serving records for "ns1.example.net") is DNSSEC signed, and the resolver intends to use TLS to make queries for names within the child zone, "example.com".

Substituting the name server name is sufficient to prevent the resolver from validating the TLS connection. It can validate the received TLS certificate, but would do expect the certificate to be for "ns1.attacker.example".

#### 4. New DNSKEY Algorithm

This new DNSKEY algorithm conforms to the structure requirements from [RFC4034], but is not itself used as actual DNSKEY algorithm. It is assigned a value from the DNSKEY algorithm table. No DNSKEY records are published in the child zone using this algorithm.

This DNSKEY is used only as the input to the corresponding DS hashes published in the parent zone.

Note that this method is orthogonal to the specific choice of DS hashes. Examples here refer to the what is published currently in the IANA tables for recommended DNSSEC parameters, including recommended choices. Any valid supported hash for DS records MAY be used.

##### 4.1. Algorithm {TBD1}

This algorithm is used to validate the NS records of the delegation for the owner name.

The original NS records are canonicalized according to the DNSSEC signing process [RFC4034] section 6, including removing any label compression, and normalizing the character cases to lower case. The RDATA field of the record is hashed using the selected digest algorithm(s), e.g. SHA2-256 for DS digest algorithm 2.

Note that only the RDATA from the wire format of the original NS record is used in constructing the DS record.

##### 4.1.1. Example

Consider the delegation in the COM zone:

```
example.com NS ns1.Example.Net  
example.com NS ns2.Example.Net
```

The input to the digest for each NS record is the uncompressed wire format of their respective RVALUES.

The Key Tag is calculated per [RFC4034] using this value as the RDATA.

The resulting combination of NS and DS records are:

```
example.com NS ns1.Example.Net
example.com NS ns2.Example.Net
; example.com DS KeyTag=FOO Algorithm={TBD1}
;   DigestType=2 Digest=sha2-256(wireformat("ns1.example.net"))
example.com DS KeyTag=FOO Algorithm={TBD1} DigestType=2 Digest=...
; example.com DS KeyTag=FOO Algorithm={TBD1}
;   DigestType=2 Digest=sha2-256(wireformat("ns2.example.net"))
example.com DS KeyTag=FOO Algorithm={TBD1} DigestType=2 Digest=...
```

#### 5. Validation Using These DS Records

These new DS records are used to validate corresponding delegation records and glue. Each NS record must have a matching DS record. The expected DS record RDATA is constructed, and a matching DS record with identical RDATA MUST be present. Any NS record without matching valid DS record MUST be ignored.

\* NS records are validated using {TBD1}. The RDATA consists of only the RDATA from the NS record.

#### 6. Protection of glue records

For the issue of glue records (parent side A/AAAA records which are not signed), please see the proposal [I-D.dickson-dnsop-glueless].

#### 7. Security Considerations

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

The target domain containing each name server name is presumed (and required) to be DNSSEC signed.

#### 8. IANA Considerations

This document has no IANA actions. (FIXME - update this doc to specify the required IANA actions - add TBD1 to the DNSKEY algorithm table)

#### 9. Normative References

- [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>>.
- [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>>.

## 10. Informative References

- [I-D.dickson-dnsop-glueless]  
Dickson, B., "Operating a Glueless DNS Authority Server", Work in Progress, Internet-Draft, draft-dickson-dnsop-glueless-00, 17 September 2021, <<https://datatracker.ietf.org/doc/html/draft-dickson-dnsop-glueless-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>>.
- [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>>.

## 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 (specifically 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](mailto:brian.peter.dickson@gmail.com)

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: 26 March 2022

B. Dickson  
GoDaddy  
22 September 2021

Operating a Glueless DNS Authority Server  
draft-dickson-dnsop-glueless-02

Abstract

This Internet Draft proposes a method for protecting authority servers against MITM and poisoning attacks, using a domain naming strategy to not require glue A/AAAA records and use of DNSSEC.

This technique assumes the use of validating resolvers.

MITM and poisoning attacks should only be effective/possible against unsigned domains.

However, until all domains are signed, this guidance is relevant, in that it can limit the attack surface of unsigned domains.

This guidance should be combined with [I-D.dickson-dnsop-ds-hack]

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 26 March 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction	2
2. Conventions and Definitions	2
3. Background	2
4. Proposed Solutions	3
5. Terminology:	3
6. Recommendations	4
7. Security Considerations	7
8. IANA Considerations	8
9. Normative References	8
10. Informative References	8
Appendix A. Acknowledgments	8
Author's Address	8

## 1. Introduction

DNS Security Extensions (DNSSEC) are additions to the DNS protocol which provide data integrity and authenticity protections, but do not provide privacy.

## 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

Use of DNSSEC requires upgrades to software for authoritative servers, resolvers, and optionally clients, in order to benefit from these protections. It also requires that DNS operators actually sign their zones and secure the corresponding delegations at the parent.

When a given domain is unsigned or not securely delegated, those protections to the zone contents are not available.



Any such insecure domain is trivially able to be altered by an on-path attacker.

An off-path attacker is limited to use of cache poisoning attacks.

However, some class of cache poisoning attacks target unsigned delegation data. These records consist of the necessary NS records, and when necessary, "glue" records for IP addresses corresponding to these NS records.

The impact to successful cache poisoning of delegation records is that the attacker may substitute their own name servers for the legitimate name server. In other words, the attacker is able to promote itself to being effectively on-path, and trivially modify unsigned domain results.

#### 4. Proposed Solutions

This work does not propose any protocol changes. It provides guidance on strategies and techniques for name server naming.

There are two kinds of delegation records that require protection against off-path attackers, for unsigned domains.

For protecting NS records used in delegations, there is a new proposal for use of a new DS record. See [I-D.dickson-dnsop-ds-hack] for details.

The present draft addresses the "glue" records, by recommending methods to make them mostly unnecessary. If there is no delegation glue data, an attacker cannot poison that data. The resolver cache would contain only authoritative address records associated with NS names. Authoritative data cannot be pre-empted by such poisoning attacks, since those are only able to replace less trusted glue records.

Additional recommendations are made to reduce the chances for errors caused by DNS operators when changing delegation records, by avoiding re-use of name server names which require glue address records.

#### 5. Terminology:

The following terms are used to disambiguate domains and server names:

- \* Registered domain - end-user (registrant) domain

- In the parent zone, the registered domain is the left-hand side of the NS record
- \* Registered domain name server - the name of the name server serving the registered domain
- In the parent zone, the registered domain name server is the right-hand side of the NS record

## 6. Recommendations

The following practice is RECOMMENDED for unsigned domains:

- \* Do not use in-bailiwick registered domain name servers for unsigned domains.
- \* Instead, use out-of-zone names for the registered domain name servers of unsigned domains.

Example:

Do NOT do the following (delegations requiring glue):

\$ORIGIN example.

```
// Records in example TLD, with relative names
unsigned-domain NS ns1.unsigned-domain
unsigned-domain NS ns2.unsigned-domain
// glue
// "strictly necessary glue"
// always required for successful resolution
ns1.unsigned-domain A (IP address)
ns1.unsigned-domain AAAA (IP address)
ns2.unsigned-domain A (IP address)
ns2.unsigned-domain AAAA (IP address)
```

Instead, do the following (glueless delegations):

\$ORIGIN example.

```
// Records in example TLD, with relative names
// This is the minimum "glueless" set-up
// NS target name is not a "registered" host
// NS target is not used for glue for any domains
unsigned-domain NS ns1.nameserver-signed-domain
unsigned-domain NS ns2.nameserver-signed-domain
//
// Delegation to signed domain containing name server names
// (This domain serves the address records of name servers
// such as the glueless example above)
nameserver-signed-domain NS ns1.nameserver-signed-domain
nameserver-signed-domain NS ns2.nameserver-signed-domain
nameserver-signed-domain DS (DS record data)
// However, this domain needs to be resolvable, and needs glue
// glue records for this delegation
ns1.nameserver-signed-domain A (IP address)
ns1.nameserver-signed-domain AAAA (IP address)
ns2.nameserver-signed-domain A (IP address)
ns2.nameserver-signed-domain AAAA (IP address)
```

The following practice is RECOMMENDED:

- \* For any name server domain (domain containing addresses and related data for name servers used by registered domains), use distinct dedicated name servers for the domain itself
  - I.e. avoid sharing name servers between the name server domain and any registered domains
- \* Consider making the name server domain itself fully glueless, with an out-of-zone name server (using a tertiary domain)

- \* For this tertiary domain, also consider using separating the in-bailiwick name servers, from the names used for serving the name server domain
  - Limiting the in-bailiwick NS names ensures that changes and updates to the tertiary domain don't affect any other domains
  - Depending on parent zone policy (e.g. TLD database policy), renaming or renumbering name servers may affect delegations using them (NS entries)
  - A single domain with non-reused NS names guarantees side effects of this sort are not possible
- \* Overhead of tertiary domain and not re-using (or sharing) name server names in the tertiary domain:
  - Additional lookups are required on the initial reference to get the addresses of name servers for the main glueless domain
  - Subsequent (new) queries for the IP addresses of glueless name servers only require single queries

Example:

Entries in the example TLD  
\$ORIGIN example.

```
//  
// Same unsigned domain uses the same name servers  
// However, the name server is in its own glueless domain  
unsigned-registrant-domain NS ns1.signed-nameserver-domain  
unsigned-registrant-domain NS ns2.signed-nameserver-domain  
//  
signed-nameserver-domain NS ns1.tertiary-domain  
signed-nameserver-domain NS ns2.tertiary-domain  
signed-nameserver-domain DS (DS record data)  
//  
tertiary-domain NS special-ns1.tertiary-domain  
tertiary-domain NS special-ns2.tertiary-domain  
tertiary-domain DS (DS record data)  
// glue for special-ns1 and -2  
// special-ns1 and -2 are used only for/by tertiary-domain  
special-ns1.tertiary-domain A (IP address)  
special-ns1.tertiary-domain AAAA (IP address)  
special-ns2.tertiary-domain A (IP address)  
special-ns2.tertiary-domain AAAA (IP address)
```

Zone file for signed-nameserver-domain.example:

```
$ORIGIN signed-nameserver-domain.example.  
@ SOA (soa record data)  
// glueless NS are used  
@ NS ns1.tertiary-domain  
@ NS ns2.tertiary-domain  
// actual glueless address records for "real" name server names  
ns1 A (IP address)  
ns1 AAAA (IP address)  
ns2 A (IP address)  
ns2 AAAA (IP address)  
// etc etc etc
```

```
Zone file for tertiary-domain.example:  
$ORIGIN tertiary-domain.example.  
@ SOA (soa record data)  
//  
// This is the only non-glueless NS in use  
// (NB: matches glue address records in the parent)  
@ NS special-ns1  
@ NS special-ns2  
special-ns1 A (IP address)  
special-ns1 AAAA (IP address)  
special-ns2 A (IP address)  
special-ns2 AAAA (IP address)  
//  
// actual address records for "real" name server name  
// (only used by signed-nameserver-domain)  
// (These match glue records in the parent zone)  
ns1 A (IP address)  
ns1 AAAA (IP address)  
ns2 A (IP address)  
ns2 AAAA (IP address)
```

## 7. Security Considerations

This guidance is useful in preventing off-path attackers from poisoning DNS cache entries necessary for delegations.

However, an on-path attacker is still able to manipulate DNS responses sent over UDP or unencrypted TCP.

This guidance is not a substitute for use of DNSSEC for DNS domains. The only mechanism that can protect against on-path attackers is cryptographic protection. DNSSEC signing of domains is both necessary and sufficient to provide data integrity protection.

Use of an encrypted transport is may be effective at preventing MITM attacks (i.e. DNS over TLS from resolver to authoritative server, aka ADoT), but does not provide provable data integrity.

Encrypted transport may be used in combination with DNSSEC signed zones and glueless name server domains.

Encrypted transport does not incrementally improve the data integrity or protection against MITM. DNSSEC is sufficient alone for this purpose. However, encrypted transport does add privacy protection against passive observers.

## 8. IANA Considerations

This document has no IANA actions.

## 9. Normative 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>>.

## 10. 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-00, 11 August 2021,  
<<https://datatracker.ietf.org/doc/html/draft-dickson-dnsop-ds-hack-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 (specifically 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](mailto:brian.peter.dickson@gmail.com)

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

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 13 May 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.



This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Conventions and Definitions . . . . .	3
3. Background . . . . .	3
4. Purpose . . . . .	3
4.1. New DNS Elements . . . . .	4
5. Requirements, and Limitations . . . . .	4
6. DNS Records To Publish for ADoT . . . . .	5
6.1. Server DNS Transport Support Signaling . . . . .	5
6.1.1. Examples . . . . .	5
6.2. DANE TLSA Records for ADoT (TLSADOT) . . . . .	6
6.2.1. Example . . . . .	6
6.3. Signaling DNS Transport for a Name Server . . . . .	7
6.3.1. Examples . . . . .	7
6.4. Signaling DNS Transport for a Domain . . . . .	7
6.4.1. Examples . . . . .	8
7. Validation Using DS Records, DNST Records, TLSADOT Records, and DNSSEC Validation . . . . .	8
7.1. Complete Example . . . . .	9
7.1.1. DNS Record Data . . . . .	9
7.1.2. Discussion Point - Wildcard-like Records . . . . .	11
7.1.3. Resolver Iterative Queries For Final TLS Query . . . . .	11
8. Signaling Resolver Support and Client Desire for ADoT . . . . .	14
8.1. Server Side Support Signaling . . . . .	15
8.2. Client Side Desire Signaling . . . . .	15
9. Security Considerations . . . . .	15
10. IANA Considerations . . . . .	15
11. Normative References . . . . .	15
12. Informative References . . . . .	16
Appendix A. Acknowledgments . . . . .	17
Author's Address . . . . .	17

## 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 availability 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. Specifically, 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 specifying 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 RRSset 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

Wildcard 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 explicitly 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 NSEC3 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 NSEC3 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 NSEC3 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 NSEC3 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 NSEC3 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 NSEC3 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 NSEC3 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 available, 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 may 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 (specifically 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](mailto:brian.peter.dickson@gmail.com)



Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 27 April 2022

B. Dickson  
GoDaddy  
24 October 2021

Resource Record for Signaling Transport for DNS to Authority Servers  
draft-dickson-dprive-dnst-00

Abstract

This Internet Draft proposes an RRTYPE to signal explicit support for transport types for DNS service. This new RRTYPE is "DNST". The available transports to signal are TCP and UDP on port 53 (DNS), and DoT (DNS over TLS) transport using TCP port 853.

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 27 April 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Conventions and Definitions . . . . .	2
3. Background . . . . .	2
4. Remove Before Publication . . . . .	2
5. DNS Transport RRTYPE . . . . .	3
6. Restrictions . . . . .	3
7. Wire Format . . . . .	3
8. Presentation Format . . . . .	3
9. Additional Processing . . . . .	4
10. Security Considerations . . . . .	4
11. IANA Considerations . . . . .	4
12. Normative References . . . . .	4
13. Informative References . . . . .	4
Appendix A. Acknowledgments . . . . .	4
Author's Address . . . . .	5

## 1. Introduction

## 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

DNS over TLS is defined in [RFC7858]. However, there is no explicit signaling for when DoT should be used. Without explicit signaling, there is no protection against downgrade attacks by an on-path attacker.

## 4. Remove Before Publication

Notes on design decisions, including the decision NOT to use an SVCB-compatible format:

- \* NS records MUST point to non-CNAME records. Thus, there is no need for the SVCB "Alias-form" behavior. DNST does not support aliasing,
- \* DNST allows for explicit rejection of default transport (UDP/53 and TCP/53)
- \* DNST allows explicit signaling of DoT

- \* There is no need for alternate port numbers for UDP or TCP port 53, or for DoT port 853.
- \* There is no need for DoH, since the expected clients are limited to DNS resolvers.

## 5. DNS Transport RRTYPE

The solution to this problem is to introduce a method for explicit signaling for when DoT is available. When combined with TLSA [RFC6698] records for the corresponding DNS server name, any client wishing to use DoT is able to know that it is available, and can detect and avoid any attempts at transport downgrade.

This document defines the RRTYPE value {TBD} with mnemonic name DNST ("DNS Transport"). This consists of a set of flags indicating supported transport for the DNS server at the owner name. The flag bits represent transports:

- \* UDP on port 53
- \* TCP on port 53
- \* DoT (DNS over TLS) on port 853

## 6. Restrictions

The DNST record may occur anywhere, including at the apex of a DNS zone, and may co-exist with any other type that also permits other types.

## 7. Wire Format

The RDATA wire format is an 8-bit octet of flag bits.

| UDP | TCP | DOT | 5 unused bits |

## 8. Presentation Format

OWNER CLASS TTL DNST [UDP] [TCP] [DOT]

At least one of the transport types must be present.

## 9. Additional Processing

The authoritative server MAY/SHOULD return both the DNST record(s) and any/all A and AAAA records with the same owner name. This reduces the number of queries the resolver would otherwise have to make (i.e. two additional queries for A and AAAA record types).

## 10. Security Considerations

The DNST record MUST be in a DNSSEC-signed zone. This ensures protection against downgrade attacks on the transport signaling.

## 11. IANA Considerations

IANA is directed to add a new record to the DNS RRTYPES table to add the entry "DNST" with value "TBD", referencing this document.

## 12. 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>>.
- [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>>.

## 13. Informative References

- [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 (specifically 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](mailto:brian.peter.dickson@gmail.com)

DNSOP Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 8 September 2022

P. van Dijk  
PowerDNS  
L. Peltan  
CZ.NIC  
O. Sury  
Internet Systems Consortium  
W. Toorop  
NLnet Labs  
K. Monshouwer

P. Thomassen  
deSEC, SSE - Secure Systems Engineering  
7 March 2022

DNS Catalog Zones  
draft-ietf-dnsop-dns-catalog-zones-05

Abstract

This document describes a method for automatic DNS zone provisioning among DNS primary and secondary nameservers by storing and transferring the catalog of zones to be provisioned as one or more regular DNS zones.

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 8 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. Description . . . . .	4
4. Catalog Zone Structure . . . . .	4
4.1. SOA and NS Records . . . . .	4
4.2. Member Zones . . . . .	4
4.3. Global Properties . . . . .	5
4.3.1. Schema Version (version property) . . . . .	6
4.4. Member Zone Properties . . . . .	6
4.4.1. Change of Ownership (coo property) . . . . .	6
4.4.2. Groups (group property) . . . . .	7
4.5. Custom Properties (*.ext properties) . . . . .	8
5. Nameserver Behavior . . . . .	9
5.1. General Requirements . . . . .	9
5.2. Member zone name clash . . . . .	9
5.3. Member zone removal . . . . .	10
5.4. Member node name change . . . . .	10
5.5. Migrating member zones between catalogs . . . . .	10
5.6. Zone associated state reset . . . . .	11
6. Implementation Notes . . . . .	11
7. Security Considerations . . . . .	11
8. Acknowledgements . . . . .	12
9. Normative References . . . . .	12
10. Informative References . . . . .	13
Appendix A. Implementation Status . . . . .	14
Appendix B. Change History (to be removed before final publication) . . . . .	14
Authors' Addresses . . . . .	17

## 1. Introduction

The content of a DNS zone is synchronized amongst its primary and secondary nameservers using AXFR and IXFR. However, the list of zones served by the primary (called a catalog in [RFC1035]) is not automatically synchronized with the secondaries. To add or remove a zone, the administrator of a DNS nameserver farm not only has to add or remove the zone from the primary, they must also add/remove the

zone from all secondaries, either manually or via an external application. This can be both inconvenient and error-prone; it is also dependent on the nameserver implementation.

This document describes a method in which the catalog is represented as a regular DNS zone (called a "catalog zone" here), and transferred using DNS zone transfers. As zones are added to or removed from the catalog zone, these changes are distributed to the secondary nameservers in the normal way. The secondary nameservers then add/remove/modify the zones they serve in accordance with the changes to the catalog zone. Other use-cases of nameserver remote configuration by catalog zones are possible, where the catalog consumer might not be a secondary.

## 2. Terminology

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.

**Catalog zone** A DNS zone containing a DNS catalog, that is, a list of DNS zones and associated properties.

**Member zone** A DNS zone whose configuration is published inside a catalog zone.

**Member node** The DNS name in the Catalog zone representing a Member zone.

**\$CATZ** Used in examples as a placeholder to represent the domain name of the catalog zone itself. \$OLDCATZ and \$NEWCATZ are used to discuss migration a member zone from one catalog zone \$OLDCATZ to another catalog zone \$NEWCATZ.

**Catalog producer** An entity that generates and is responsible for the contents of the catalog zone.

**Catalog consumer** An entity that extracts information from the catalog zone (such as a DNS server that configures itself according to the catalog zone's contents).



### 3. Description

A catalog zone is a DNS zone whose contents are specially crafted. Its records primarily constitute a list of PTR records referencing other DNS zones (so-called "member zones"). The catalog zone may contain other records indicating additional metadata (so-called "properties") associated with these member zones.

Catalog consumers SHOULD ignore any RR in the catalog zone which is meaningless or useless to the implementation.

Authoritative servers may be preconfigured with multiple catalog zones, each associated with a different set of configurations.

Although the contents of a catalog zone are interpreted and acted upon by nameservers, a catalog zone is a regular DNS zone and so must adhere to the standards for such zones.

A catalog zone is primarily intended for the management of a farm of authoritative nameservers. The content of catalog zones may not be accessible from any recursive nameserver.

### 4. Catalog Zone Structure

#### 4.1. SOA and NS Records

As with any other DNS zone, a catalog zone MUST have a syntactically correct SOA record and at least one NS record at its apex.

The SOA record's SERIAL, REFRESH, RETRY and EXPIRE fields [RFC1035] are used during zone transfer. A catalog zone's SOA SERIAL field MUST increase when an update is made to the catalog zone's contents as per serial number arithmetic defined in [RFC1982]. Otherwise, catalog consumers might not notice updates to the catalog zone's contents.

There is no requirement to be able to query the catalog zone via recursive nameservers. Catalog consumers MUST ignore and MUST NOT assume or require NS records at the apex. However, at least one is still required so that catalog zones are syntactically correct DNS zones. A single NS RR with a NSDNAME field containing the absolute name "invalid." is RECOMMENDED [RFC2606].

#### 4.2. Member Zones

The list of member zones is specified as a collection of member nodes, represented by domain names under the owner name "zones" where "zones" is a direct child domain of the catalog zone.

The names of member zones are represented on the RDATA side (instead of as a part of owner names) of a PTR record, so that all valid domain names may be represented regardless of their length [RFC1035]. This PTR record MUST be the only record in the PTR RRset with the same name. More than one record in the RRset denotes a broken catalog zone which MUST NOT be processed (see Section 5.1).

For example, if a catalog zone lists three zones "example.com.", "example.net." and "example.org.", the member node RRs would appear as follows:

```
<unique-1>.zones.$CATZ 0 IN PTR example.com.  
<unique-2>.zones.$CATZ 0 IN PTR example.net.  
<unique-3>.zones.$CATZ 0 IN PTR example.org.
```

where <unique-N> is a label that tags each record in the collection. <unique-N> has an unique value in the collection.

Member node labels carry no informational meaning beyond labeling member zones. A changed label may indicate that the state for a zone needs to be reset (see Section 5.6).

Having the zones uniquely tagged with the <unique-N> label ensures that additional RRs can be added below the member node (see Section 4.4).

The CLASS field of every RR in a catalog zone MUST be IN (1).

The TTL field's value is not defined by this memo. Catalog zones are for authoritative nameserver management only and are not intended for general querying via recursive resolvers.

#### 4.3. Global Properties

Apart from catalog zone metadata stored at the apex (NS, SOA and the like), catalog zone information is stored in the form of "properties". Catalog consumers SHOULD ignore properties they do not understand.

This specification defines a number of so-called properties, as well as a mechanism to allow implementers to store additional information in the catalog zone with Custom properties, see Section 4.5. The meaning of such custom properties is determined by the implementation in question.

Some properties are defined at the global level; others are scoped to apply only to a specific member zone. This document defines a single mandatory global property in Section 4.3.1. Member-specific properties are described in Section 4.4.

More properties may be defined in future documents.

#### 4.3.1. Schema Version (version property)

The catalog zone schema version is specified by an integer value embedded in a TXT RR named version.\$CATZ. All catalog zones MUST have a TXT RRset named version.\$CATZ with exactly one RR. Catalog consumers MUST NOT apply catalog zone processing to zones without the expected value in the version.\$CATZ TXT RR, but they may be transferred as ordinary zones. For this memo, the value of the version.CATZ TXT RR MUST be set to "2", i.e.:

```
version.$CATZ 0 IN TXT "2"
```

NB: Version 1 was used in a draft version of this memo and reflected the implementation first found in BIND 9.11.

#### 4.4. Member Zone Properties

Each member zone MAY have one or more additional properties, described in this chapter. These properties are completely optional and catalog consumers SHOULD ignore those it does not understand. Member zone properties are represented by RRsets below the corresponding member node.

##### 4.4.1. Change of Ownership (coo property)

The coo property facilitates controlled migration of a member zone from one catalog to another.

A Change Of Ownership is signaled by the coo property in the catalog zone currently "owning" the zone. The name of the new catalog is in the value of a PTR record in the old catalog. For example if member "example.com." will migrate from catalog zone \$OLDCATZ to catalog zone \$NEWCATZ, this appears in the \$OLDCATZ catalog zone as follows:

```
<unique-N>.zones.$OLDCATZ 0 IN PTR example.com.  
coo.<unique-N>.zones.$OLDCATZ 0 IN PTR $NEWCATZ
```

The PTR RRset MUST consist of a single PTR record. More than one record in the RRset denotes a broken catalog zone which MUST NOT be processed (see Section 5.1).

When a consumer of catalog zone \$OLDCATZ receives an update which adds or changes a coo property for a member zone in \$OLDCATZ signalling a new owner \$NEWCATZ, it does not migrate the member zone immediately.

This is because the catalog consumer may not have the <unique-N> identifier associated with the member zone in \$NEWCATZ and because name servers do not index Resource Records by RDATA, it may not know whether or not the member zone is configured in \$NEWCATZ at all. It may have to wait for an update of \$NEWCATZ adding or changing that member zone. When a consumer of catalog zone \$NEWCATZ receives an update of \$NEWCATZ which adds or changes a member zone, and that consumer had the member zone associated with \$OLDCATZ, and there is a coo property of the member zone in \$OLDCATZ pointing to \$NEWCATZ, only then it will reconfigure the member zone with the for \$NEWCATZ preconfigured settings.

Unless the member node label (i.e. <unique-N>) for the member is the same in \$NEWCATZ, all associated state for a just migrated zone **MUST** be reset (see Section 5.6). Note that the owner of \$OLDCATZ allows for the zone associated state to be taken over by the owner of \$NEWCATZ by default. To prevent the takeover, the owner of \$OLDCATZ has to enforce a zone state reset by changing the member node label (see Section 5.6) before or simultaneous with adding the coo property. (see also Section 7)

The old owner may remove the member zone containing the coo property from \$OLDCATZ once it has been established that all its consumers have processed the Change of Ownership.

#### 4.4.2. Groups (group property)

With a group property, consumer(s) can be signalled to treat some member zones within the catalog zone differently.

The consumer **MAY** apply different configuration options when processing member zones, based on the value of the group property. The exact handling of configuration referred to by the group property value is left to the consumer's implementation and configuration. The property is defined by a TXT record in the sub-node labelled group.

The producer **MAY** assign a group property to all, some, or none of the member zones within a catalog zone. The producer **MUST NOT** assign more than one group property to one member zone.

The consumer **MUST** ignore either all or none of the group properties in a catalog zone.

The value of the TXT record MUST be at most 255 octets long and MUST NOT contain whitespace characters. The consumer MUST interpret the value case-sensitively.

#### 4.4.2.1. Example

```
<unique-1>.zones.$CATZ      0 IN PTR    example.com.  
group.<unique-1>.zones.$CATZ 0 IN TXT    sign-with-nsec3  
<unique-2>.zones.$CATZ      0 IN PTR    example.net.  
group.<unique-2>.zones.$CATZ 0 IN TXT    nodnssec
```

In this case, the consumer might be implemented and configured in the way that the member zones with "nodnssec" group assigned will not be signed with DNSSEC, and the zones with "sign-with-nsec3" group assigned will be signed with DNSSEC with NSEC3 chain.

By generating the catalog zone (snippet) above, the producer signals how the consumer shall treat DNSSEC for the zones example.net. and example.com., respectively.

#### 4.5. Custom Properties (\*.ext properties)

Implementations and operators of catalog zones may choose to provide their own properties. Custom properties can occur both globally, or for a specific member zone. To prevent a name clash with future properties, such properties should be represented below the label ext.

ext is not a placeholder, so a custom property would have domains names as follows:

```
<your-property>.ext.$CATZ      # for a global custom property  
<your-property>.ext.<unique-N>.zones.$CATZ # for a member zone custom property
```

<your-property> may consist of one or more labels.

Implementations MAY use such properties on the member zone level to store additional information about member zones, for example to flag them for specific treatment (such as ...).

Further, implementations MAY use custom properties on the global level to store additional information about the catalog zone itself. While there may be many use cases for this, a plausible one is to store default values for custom properties on the global level, then overriding them using a property of the same name on the member level (= under the ext label of the member node) if so desired. A property description should clearly say what semantics apply, and whether a property is global, member, or both.

The meaning of the custom properties described in this section is determined by the implementation alone, without expectation of interoperability. A catalog consumer SHOULD ignore custom properties it does not understand.

## 5. Nameserver Behavior

### 5.1. General Requirements

As it is a regular DNS zone, a catalog zone can be transferred using DNS zone transfers among nameservers.

Although they are regular DNS zones, catalog zones contain only information for the management of a set of authoritative nameservers. For this reason, operators may want to limit the systems able to query these zones. It may be inconvenient to serve some contents of catalog zones via DNS queries anyway due to the nature of their representation. A separate method of querying entries inside the catalog zone may be made available by nameserver implementations (see Section 6).

Catalog updates should be automatic, i.e., when a nameserver that supports catalog zones completes a zone transfer for a catalog zone, it SHOULD apply changes to the catalog within the running nameserver automatically without any manual intervention.

As with regular zones, primary and secondary nameservers for a catalog zone may be operated by different administrators. The secondary nameservers may be configured as catalog consumer to synchronize catalog zones from the primary, but the primary's administrators may not have any administrative access to the secondaries.

Nameservers MAY allow loading and transfer of broken zones with incorrect catalog zone syntax (as they are treated as regular zones), but catalog consumers MUST NOT process such broken zones as catalog zones. For the purpose of catalog processing, the broken catalogs MUST be ignored.

### 5.2. Member zone name clash

If there is a clash between an existing zone's name (either from an existing member zone or otherwise configured zone) and an incoming member zone's name (via transfer or update), the new instance of the zone MUST be ignored and an error SHOULD be logged.

A clash between an existing member zone's name and an incoming member zone's name (via transfer or update), may be an attempt to migrate a zone to a different catalog, but should not be treated as one except as described in {#cooproperty}.

### 5.3. Member zone removal

When a member zone is removed from a specific catalog zone, an authoritative server MUST NOT remove the zone and associated state data if the zone was not configured from that specific catalog zone. Only when the zone was configured from a specific catalog zone, and the zone is removed as a member from that specific catalog zone, the zone and associated state (such as zone data and DNSSEC keys) MUST be removed.

### 5.4. Member node name change

When via a single update or transfer, the member node's label value (<unique-N>) changes, catalog consumers MUST process this as a member zone removal including all the zone's associated state (as described in Section 5.3), immediately followed by processing the member as a newly to be configured zone in the same catalog.

### 5.5. Migrating member zones between catalogs

If all consumers of the catalog zones involved support the cooproperty, it is RECOMMENDED to perform migration of a member zone by following the procedure described in Section 4.4.1. Otherwise a migration of member zone from a catalog zone \$OLDCATZ to a catalog zone \$NEWCATZ has to be done by: first removing the member zone from \$OLDCATZ; second adding the member zone to \$NEWCATZ.

If in the process of a migration some consumers of the involved catalog zones did not catch the removal of the member zone from \$OLDCATZ yet (because of a lost packet or down time or otherwise), but did already see the update of \$NEWCATZ, they may consider the update adding the member zone in \$NEWCATZ to be a name clash (see Section 5.2) and as a consequence the member is not migrated to \$NEWCATZ. This possibility needs to be anticipated with a member zone migration. Recovery from such a situation is out of the scope of this document. It may for example entail a manually forced retransfer of \$NEWCATZ to consumers after they have been detected to have received and processed the removal of the member zone from \$OLDCATZ.

### 5.6. Zone associated state reset

It may be desirable to reset state (such as zone data and DNSSEC keys) associated with a member zone.

A zone state reset may be performed by a change of the member node's name (see Section 5.4).

## 6. Implementation Notes

Catalog zones on secondary nameservers would have to be setup manually, perhaps as static configuration, similar to how ordinary DNS zones are configured. The secondary additionally needs to be configured as a catalog consumer for the catalog zone to enable processing of the member zones in the catalog, such as automatic synchronization of the member zones for secondary service.

An administrator may want to look at data inside a catalog zone. Typical queries might include dumping the list of member zones, dumping a member zone's effective configuration, querying a specific property value of a member zone, etc. Because of the structure of catalog zones, it may not be possible to perform these queries intuitively, or in some cases, at all, using DNS QUERY. For example, it is not possible to enumerate the contents of a multi-valued property (such as the list of member zones) with a single QUERY. Implementations are therefore advised to provide a tool that uses either the output of AXFR or an out-of-band method to perform queries on catalog zones.

## 7. Security Considerations

As catalog zones are transmitted using DNS zone transfers, it is RECOMMENDED that catalog zone transfer are protected from unexpected modifications by way of authentication, for example by using TSIG [RFC8945], or Strict or Mutual TLS authentication with DNS Zone transfer over TLS [RFC9103].

Use of DNS UPDATE [RFC2136] to modify the content of catalog zones SHOULD similarly be authenticated.

Zone transfers of member zones SHOULD similarly be authenticated. TSIG shared secrets used for member zones SHOULD NOT be mentioned in the catalog zone data. However, key identifiers may be shared within catalog zones.



Catalog zones reveal the zones served by the consumers of the catalog zone. It is RECOMMENDED to limit the systems able to query these zones. It is RECOMMENDED to transfer catalog zones confidentially [RFC9103].

Administrative control over what zones are served from the configured name servers shifts completely from the server operator (consumer) to the "owner" (producer) of the catalog zone content.

With migration of member zones between catalogs using the `coo` property, it is possible for the owner of the target catalog (i.e. `$NEWCATZ`) to take over all associated state with the zone from the original owner (i.e. `$OLDCATZ`) by maintaining the same member node label (i.e. `<unique-N>`). To prevent the takeover of the zone associated state, the original owner has to enforce a zone state reset by changing the member node label (see Section 5.6) before or simultaneously with adding the `coo` property.

## 8. Acknowledgements

Our deepest thanks and appreciation go to Stephen Morris, Ray Bellis and Witold Krecicki who initiated this draft and did the bulk of the work.

Catalog zones originated as the chosen method among various proposals that were evaluated at ISC for easy zone management. The chosen method of storing the catalog as a regular DNS zone was proposed by Stephen Morris.

The initial authors discovered that Paul Vixie's earlier [Metazones] proposal implemented a similar approach and reviewed it. Catalog zones borrows some syntax ideas from Metazones, as both share this scheme of representing the catalog as a regular DNS zone.

Thanks to Leo Vandewoestijne. Leo's presentation in the DNS devroom at the FOSDEM'20 [FOSDEM20] was one of the motivations to take up and continue the effort of standardizing catalog zones.

Thanks to Brian Conry, Klaus Darilion, Brian Dickson, Tony Finch, Evan Hunt, Shane Kerr, Patrik Lundin, Victoria Risk, Petr Spacek and Carsten Strotmann for reviewing draft proposals and offering comments and suggestions.

## 9. Normative References

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.

- [RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <<https://www.rfc-editor.org/info/rfc1982>>.
- [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>>.
- [RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <<https://www.rfc-editor.org/info/rfc2136>>.
- [RFC2606] Eastlake 3rd, D. and A. Panitz, "Reserved Top Level DNS Names", BCP 32, RFC 2606, DOI 10.17487/RFC2606, June 1999, <<https://www.rfc-editor.org/info/rfc2606>>.
- [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>>.
- [RFC8945] Dupont, F., Morris, S., Vixie, P., Eastlake 3rd, D., Gudmundsson, O., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", STD 93, RFC 8945, DOI 10.17487/RFC8945, November 2020, <<https://www.rfc-editor.org/info/rfc8945>>.
- [RFC9103] Toorop, W., Dickinson, S., Sahib, S., Aras, P., and A. Mankin, "DNS Zone Transfer over TLS", RFC 9103, DOI 10.17487/RFC9103, August 2021, <<https://www.rfc-editor.org/info/rfc9103>>.

## 10. Informative References

- [FOSDEM20] Vandewoestijne, L., "Extending Catalog zones - another approach in automating maintenance", 2020, <[https://archive.fosdem.org/2020/schedule/event/dns\\_catz/](https://archive.fosdem.org/2020/schedule/event/dns_catz/)>.
- [Metazones] Vixie, P., "Federated Domain Name Service Using DNS Metazones", 2005, <<http://family.redbarn.org/~vixie/mz.pdf>>.

## Appendix A. Implementation Status

\*Note to the RFC Editor\*: please remove this entire section before publication.

In the following implementation status descriptions, "DNS Catalog Zones" refers to DNS Catalog Zones as described in this document.

- \* Knot DNS 3.1 (released August 2, 2021) supports full producing and consuming of catalog zones, including the group property.
- \* PowerDNS has a proof of concept external program called PowerCATZ (<https://github.com/PowerDNS/powercatz/>), that can process DNS Catalog Zones.
- \* Proof of concept python scripts (<https://github.com/IETF-Hackathon/NSDCatZ>) that can be used for both generating and consuming DNS Catalog Zones with NSD have been developed during the hackathon at the IETF-109.

Interoperability between the above implementations has been tested during the hackathon at the IETF-109.

## Appendix B. Change History (to be removed before final publication)

- \* draft-muks-dnsop-dns-catalog-zones-00
  - | Initial public draft.
- \* draft-muks-dnsop-dns-catalog-zones-01
  - | Added Witold, Ray as authors. Fixed typos, consistency issues. Fixed references. Updated Area. Removed newly introduced custom RR TYPES. Changed schema version to 1. Changed TSIG requirement from MUST to SHOULD. Removed restrictive language about use of DNS QUERY. When zones are introduced into a catalog zone, a primary SHOULD first make the new zones available for transfers first (instead of MUST). Updated examples, esp. use IPv6 in examples per Fred Baker. Add catalog zone example.
- \* draft-muks-dnsop-dns-catalog-zones-02
  - | Addressed some review comments by Patrik Lundin.
- \* draft-muks-dnsop-dns-catalog-zones-03
  - | Revision bump.

\* draft-muks-dnsop-dns-catalog-zones-04

Reordering of sections into more logical order. Separation of multi-valued properties into their own category.

\* draft-toorop-dnsop-dns-catalog-zones-00

New authors to pickup the editor pen on this draft

Remove data type definitions for zone properties Removing configuration of member zones through zone properties altogether

Remove Open issues and discussion Appendix, which was about zone options (including primary/secondary relationships) only.

\* draft-toorop-dnsop-dns-catalog-zones-01

Added a new section "The Serial Property", introducing a new mechanism which can help with disseminating zones from the primary to the secondary nameservers in a timely fashion more reliably.

Three different ways to provide a "serial" property with a member zone are offered to or the workgroup for discussion.

Added a new section "Implementation Status", listing production ready, upcoming and Proof of Concept implementations, and reporting on interoperability of the different implementations.

\* draft-toorop-dnsop-dns-catalog-zones-02

Adding the coo property for zone migration in a controlled fashion

Adding the group property for reconfigure settings of member zones in an atomic update

Adding the epoch property to reset zone associated state in a controlled fashion

\* draft-toorop-dnsop-dns-catalog-zones-03

Big cleanup!

Introducing the terms catalog consumer and catalog producer

Reorganized topics to create a more coherent whole

Properties all have consistent format now

Try to assume the least possible from implementations w.r.t.:

- 1) Predictability of the <unique-N> IDs of member zones
- 2) Whether or not fallback catalog zones can be found for a member
- 3) Whether or not a catalog consumer can maintain state

\* draft-toorop-dnsop-dns-catalog-zones-04

Move Implementation status to appendix

Miscellaneous textual improvements

coo property points to \$NEWCATZ (and not zones.\$NEWCATZ)

Remove suggestion to increase serial and remove member zone from \$OLDCATZ after migration

More consistent usage of the terms catalog consumer and catalog producer throughout the document

Better (safer) description of resetting refresh timers of member zones with the serial property

Removing a member MUST remove zone associated state

Make authentication requirements a bit less prescriptive in security considerations

Updated implementation status for KnotDNS

Describe member node name changes and update "Zone associated state reset" to use that as the mechanism for it.

Add Peter Thomassen as co-author

Complete removal of the epoch property. We consider consumer optimizations with predictable member node labels (for example based on a hash) out of the scope of this document.

Miscellaneous editorial improvements

\* draft-toorop-dnsop-dns-catalog-zones-05

Add Kees Monshouwer as co-author

Removed the "serial" property

| Allow custom properties on the global level

#### Authors' Addresses

Peter van Dijk  
PowerDNS  
Den Haag  
Netherlands  
Email: peter.van.dijk@powerdns.com

Libor Peltan  
CZ.NIC  
Czechia  
Email: libor.peltan@nic.cz

Ondrej Sury  
Internet Systems Consortium  
Czechia  
Email: ondrej@isc.org

Willem Toorop  
NLnet Labs  
Science Park 400  
1098 XH Amsterdam  
Netherlands  
Email: willem@nlnetlabs.nl

Kees Monshouwer  
Netherlands  
Email: mind@monshouwer.eu

Peter Thomassen  
deSEC, SSE - Secure Systems Engineering  
Berlin  
Germany  
Email: peter@desec.io

Network Working Group  
Internet-Draft  
Intended status: Best Current Practice  
Expires: 18 October 2022

W. Hardaker  
USC/ISI  
V. Dukhovni  
Bloomberg, L.P.  
16 April 2022

Guidance for NSEC3 parameter settings  
draft-ietf-dnsop-nsec3-guidance-08

Abstract

NSEC3 is a DNSSEC mechanism providing proof of non-existence by asserting that there are no names that exist between two domain names within a zone. Unlike its counterpart NSEC, NSEC3 avoids directly disclosing the bounding domain name pairs. This document provides guidance on setting NSEC3 parameters based on recent operational deployment experience.

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 18 October 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	3
1.1. Requirements Notation . . . . .	3
2. NSEC3 Parameter Value Considerations . . . . .	3
2.1. Algorithms . . . . .	3
2.2. Flags . . . . .	4
2.3. Iterations . . . . .	4
2.4. Salt . . . . .	5
3. Recommendations for Deploying and Validating NSEC3 Records .	5
3.1. Best-practice for Zone Publishers . . . . .	6
3.2. Recommendation for Validating Resolvers . . . . .	6
3.3. Recommendation for Primary / Secondary Relationships . .	7
4. Security Considerations . . . . .	8
5. Operational Considerations . . . . .	8
6. IANA Considerations . . . . .	8
7. References . . . . .	8
7.1. Normative References . . . . .	8
7.2. Informative References . . . . .	9
Appendix A. Deployment measurements at time of publication . . .	9
Appendix B. Computational burdens of processing NSEC3 iterations . . . . .	9
Appendix C. Acknowledgments . . . . .	10
Appendix D. Github Version of This Document . . . . .	10
Appendix E. Implementation Notes . . . . .	11
E.1. OpenDNSSEC . . . . .	11
E.2. PowerDNS . . . . .	11
E.3. Knot DNS and Knot Resolver . . . . .	11
E.4. Google Public DNS Resolver . . . . .	11
E.5. Google Cloud DNS . . . . .	11
Authors' Addresses . . . . .	11



## 1. Introduction

As with NSEC [RFC4035], NSEC3 [RFC5155] provides proof of non-existence that consists of signed DNS records establishing the non-existence of a given name or associated Resource Record Type (RRTYPE) in a DNSSEC [RFC4035] signed zone. In the case of NSEC3, however, the names of valid nodes in the zone are obfuscated through (possibly multiple iterations of) hashing (currently only SHA-1 is in use within the Internet).

NSEC3 also provides "opt-out support", allowing for blocks of unsigned delegations to be covered by a single NSEC3 record. Use of the opt-out feature allows large registries to only sign as many NSEC3 records as there are signed DS or other RRsets in the zone; with opt-out, unsigned delegations don't require additional NSEC3 records. This sacrifices the tamper-resistance proof of non-existence offered by NSEC3 in order to reduce memory and CPU overheads.

NSEC3 records have a number of tunable parameters that are specified via an NSEC3PARAM record at the zone apex. These parameters are the hash algorithm, processing flags, the number of hash iterations and the salt. Each of these has security and operational considerations that impact both zone owners and validating resolvers. This document provides some best-practice recommendations for setting the NSEC3 parameters.

### 1.1. Requirements Notation

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.

## 2. NSEC3 Parameter Value Considerations

The following sections describe recommendations for setting parameters for NSEC3 and NSEC3PARAM.

### 2.1. Algorithms

The algorithm field is not discussed by this document.

## 2.2. Flags

The NSEC3PARAM flags field currently contains no flags, but individual NSEC3 records contain the "Opt-Out" flag [RFC5155], which specifies whether or not that NSEC3 record provides proof of non-existence or not. In general, NSEC3 with the Opt-Out flag enabled should only be used in large, highly dynamic zones with a small percentage of signed delegations. Operationally, this allows for fewer signature creations when new delegations are inserted into a zone. This is typically only necessary for extremely large registration points providing zone updates faster than real-time signing allows or when using memory-constrained hardware. Smaller zones, or large but relatively static zones, are encouraged to use a flags value of 0 (zero) and take advantage of DNSSEC's proof-of-non-existence support.

## 2.3. Iterations

NSEC3 records are created by first hashing the input domain and then repeating that hashing algorithm a number of times based on the iteration parameter in the NSEC3PARAM and NSEC3 records. The first hash is typically sufficient to discourage zone enumeration performed by "zone walking" an NSEC or NSEC3 chain. Only determined parties with significant resources are likely to try and uncover hashed values, regardless of the number of additional iterations performed. If an adversary really wants to expend significant CPU resources to mount an offline dictionary attack on a zone's NSEC3 chain, they'll likely be able to find most of the "guessable" names despite any level of additional hashing iterations.

Most names published in the DNS are rarely secret or unpredictable. They are published to be memorable, used and consumed by humans. They are often recorded in many other network logs such as email logs, certificate transparency logs, web page links, intrusion detection systems, malware scanners, email archives, etc. Many times a simple dictionary of commonly used domain names prefixes (www, ftp, mail, imap, login, database, etc) can be used to quickly reveal a large number of labels within a zone. Because of this, there are increasing performance costs yet diminishing returns associated with applying additional hash iterations beyond the first.

Although Section 10.3 of [RFC5155] specifies upper bounds for the number of hash iterations to use, there is no published guidance for zone owners about good values to select. Recent academic studies have shown that NSEC3 hashing provides only moderate protection [GPUNSEC3][ZONEENUM].

## 2.4. Salt

NSEC3 records provide an additional salt value, which can be combined with an FQDN to influence the resulting hash, but properties of this extra salt are complicated.

In cryptography, salts generally add a layer of protection against offline, stored dictionary attacks by combining the value to be hashed with a unique "salt" value. This prevents adversaries from building up and remembering a single dictionary of values that can translate a hash output back to the value that it derived from.

In the case of DNS, the situation is different because the hashed names placed in NSEC3 records are always implicitly "salted" by hashing the fully-qualified domain name from each zone. Thus, no single pre-computed table works to speed up dictionary attacks against multiple target zones. An attacker is always required to compute a complete dictionary per zone, which is expensive in both storage and CPU time.

To understand the role of the additional NSEC3 salt field, we have to consider how a typical zone walking attack works. Typically, the attack has two phases - online and offline. In the online phase, an attacker "walks the zone" by enumerating (almost) all hashes listed in NSEC3 records and storing them for the offline phase. Then, in the offline cracking phase, the attacker attempts to crack the underlying hash. In this phase, the additional salt value raises the cost of the attack only if the salt value changes during the online phase of the attack. In other words, an additional, constant salt value does not change the cost of the attack.

Changing a zone's salt value requires the construction of a complete new NSEC3 chain. This is true both when resigning the entire zone at once, and when incrementally signing it in the background where the new salt is only activated once every name in the chain has been completed. As a result, re-salting is a very complex operation, with significant CPU time, memory, and bandwidth consumption. This makes very frequent re-salting impractical, and renders the additional salt field functionally useless.

## 3. Recommendations for Deploying and Validating NSEC3 Records

The following subsections describe recommendations for the different operating realms within the DNS.

### 3.1. Best-practice for Zone Publishers

First, if the operational or security features of NSEC3 are not needed, then NSEC SHOULD be used in preference to NSEC3. NSEC3 requires greater computational power (see Appendix B) for both authoritative servers and validating clients. Specifically, there is a nontrivial complexity in finding matching NSEC3 records to randomly generated prefixes within a DNS zone. NSEC mitigates this concern. If NSEC3 must be used, then an iterations count of 0 MUST be used to alleviate computational burdens. Note that extra iteration counts other than 0 increase the impact of CPU-exhausting DoS attacks, and also increase the risk of interoperability problems.

Note that deploying NSEC with minimally covering NSEC records [RFC4470] also incurs a cost, and zone owners should measure the computational difference in deploying both RFC4470 or NSEC3.

In short, for all zones, the recommended NSEC3 parameters are as shown below:

```
; SHA-1, no extra iterations, empty salt:
;
bcp.example. IN NSEC3PARAM 1 0 0 -
```

For small zones, the use of opt-out based NSEC3 records is NOT RECOMMENDED.

For very large and sparsely signed zones, where the majority of the records are insecure delegations, opt-out MAY be used.

Operators are encouraged to forgo using a salt entirely by using a zero-length salt value instead (represented as a "-" in the presentation format).

If salts are used, note that since the NSEC3PARAM RR is not used by validating resolvers (see [RFC5155] section 4), the iterations and salt parameters can be changed without the need to wait for RRsets to expire from caches. A complete new NSEC3 chain needs to be constructed and the zone resigned.

### 3.2. Recommendation for Validating Resolvers

Because there has been a large growth of open (public) DNSSEC validating resolvers that are subject to compute resource constraints when handling requests from anonymous clients, this document recommends that validating resolvers change their behavior with respect to large iteration values. Specifically, validating resolver operators and validating resolver software implementers are

encouraged to continue evaluating NSEC3 iteration count deployments and lower their default acceptable limits over time. Similarly, because treating a high iterations count as insecure leaves zones subject to attack, validating resolver operators and validating resolver software implementers are further encouraged to lower their default and acceptable limit for returning SERVFAIL when processing NSEC3 parameters containing large iteration count values. See Appendix A for measurements taken near the time of publication and potential starting points.

Validating resolvers MAY return an insecure response to their clients when processing NSEC3 records with iterations larger than 0. Note also that a validating resolver returning an insecure response MUST still validate the signature over the NSEC3 record to ensure the iteration count was not altered since record publication (see [RFC5155] section 10.3).

Validating resolvers MAY also return a SERVFAIL response when processing NSEC3 records with iterations larger than 0. Validating resolvers MAY choose to ignore authoritative server responses with iteration counts greater than 0, which will likely result in returning a SERVFAIL to the client when no acceptable responses are received from authoritative servers.

Validating resolvers returning an insecure or SERVFAIL answer to their client after receiving and validating an unsupported NSEC3 parameter from the authoritative server(s) SHOULD return an Extended DNS Error (EDE) [RFC8914] EDNS0 option of value (RFC EDITOR: TBD). Validating resolvers that choose to ignore a response with an unsupported iteration count (and do not validate the signature) MUST NOT return this EDE option.

Note that this specification updates [RFC5155] by significantly decreasing the requirements originally specified in Section 10.3 of [RFC5155]. See the Security Considerations for arguments on how to handle responses with non-zero iteration count.

### 3.3. Recommendation for Primary / Secondary Relationships

Primary and secondary authoritative servers for a zone that are not being run by the same operational staff and/or using the same software and configuration must take into account the potential differences in NSEC3 iteration support.

Operators of secondary services should advertise the parameter limits that their servers support. Correspondingly, operators of primary servers need to ensure that their secondaries support the NSEC3 parameters they expect to use in their zones. To ensure reliability,

after primaries change their iteration counts, they should query their secondaries with known non-existent labels to verify the secondary servers are responding as expected.

#### 4. Security Considerations

This entire document discusses security considerations with various parameters selections of NSEC3 and NSEC3PARAM fields.

The point where a validating resolver returns insecure vs the point where it returns SERVFAIL must be considered carefully. Specifically, when a validating resolver treats a zone as insecure above a particular value (say 100) and returns SERVFAIL above a higher point (say 500), it leaves the zone subject to attacker-in-the-middle attacks as if it was unsigned between these values. Thus, validating resolver operators and software implementers SHOULD set the point above which a zone is treated as insecure for certain values of NSEC3 iterations counts to the same as the point where a validating resolver begins returning SERVFAIL.

#### 5. Operational Considerations

This entire document discusses operational considerations with various parameters selections of NSEC3 and NSEC3PARAM fields.

#### 6. IANA Considerations

This document requests a new allocation in the "Extended DNS Error Codes" of the "Domain Name System (DNS) Parameters" registration table with the following characteristics:

- \* INFO-CODE: (RFC EDITOR: TBD)
- \* Purpose: Unsupported NSEC3 iterations value
- \* Reference: (RFC EDITOR: this document)

#### 7. References

##### 7.1. Normative References

- [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>>.

- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <<https://www.rfc-editor.org/info/rfc4035>>.
- [RFC4470] Weiler, S. and J. Ihren, "Minimally Covering NSEC Records and DNSSEC On-line Signing", RFC 4470, DOI 10.17487/RFC4470, April 2006, <<https://www.rfc-editor.org/info/rfc4470>>.
- [RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, DOI 10.17487/RFC5155, March 2008, <<https://www.rfc-editor.org/info/rfc5155>>.

## 7.2. Informative References

- [GPUNSEC3] Wander, M., Schwittmann, L., Boelmann, C., and T. Weis, "GPU-Based NSEC3 Hash Breaking", DOI 10.1109/NCA.2014.27, 2014, <<https://doi.org/10.1109/NCA.2014.27>>.
- [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>>.
- [ZONEENUM] Wang, Z., Xiao, L., and R. Wang, "An efficient DNSSEC zone enumeration algorithm", n.d..

## Appendix A. Deployment measurements at time of publication

At the time of publication, setting an upper limit of 100 iterations for treating a zone as insecure is interoperable without significant problems, but at the same time still enables CPU-exhausting DoS attacks.

At the time of publication, returning SERVFAIL beyond 500 iterations appears to be interoperable without significant problems.

## Appendix B. Computational burdens of processing NSEC3 iterations

The queries per second (QPS) of authoritative servers will decrease due to computational overhead when processing DNS requests for zones containing higher NSEC3 iteration counts. The table (Appendix C) below shows the drop in QPS for various iteration counts.

Iterations	QPS [% of 0 iterations QPS]
0	100 %
10	89 %
20	82 %
50	64 %
100	47 %
150	38 %

#### Appendix C. Acknowledgments

The authors would like to thank the dns-operations discussion participants, which took place on mattermost hosted by DNS-OARC.

Additionally, the following people contributed text or review comments to the draft:

- \* Vladimir &#268;unat
- \* Tony Finch
- \* Paul Hoffman
- \* Warren Kumari
- \* Alexander Mayrhofer
- \* Matthijs Mekking
- \* Florian Obser
- \* Petr &#352;pa&#269;ek
- \* Paul Vixie
- \* Tim Wicinski

#### Appendix D. Github Version of This Document

While this document is under development, it can be viewed, tracked, issued, pushed with PRs, ... here:

<https://github.com/hardaker/draft-hardaker-dnsop-nsec3-guidance>



## Appendix E. Implementation Notes

The following implementations have implemented the guidance in this document. They have graciously provided notes about the details of their implementation below.

### E.1. OpenDNSSEC

The OpenDNSSEC configuration checking utility will alert the user about nsec3 iteration values larger than 100.

### E.2. PowerDNS

PowerDNS 4.5.2 changed the default value of nsec3-max-iterations to 150.

### E.3. Knot DNS and Knot Resolver

Knot DNS 3.0.6 warns when signing with more than 20 NSEC3 iterations. Knot Resolver 5.3.1 treats NSEC3 iterations above 150 as insecure.

### E.4. Google Public DNS Resolver

Google Public DNS treats NSEC3 iterations above 100 as insecure since September 2021.

### E.5. Google Cloud DNS

Google Cloud DNS uses 1 iteration and 64-bits of fixed random salt for all zones using NSEC3. These parameters cannot be adjusted by users.

## Authors' Addresses

Wes Hardaker  
USC/ISI  
Email: ietf@hardakers.net

Viktor Dukhovni  
Bloomberg, L.P.  
Email: ietf-dane@dukhovni.org

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: 8 September 2022

S. Sahib  
Brave Software  
S. Huque  
Salesforce  
P. Wouters  
Aiven  
7 March 2022

Survey of Domain Verification Techniques using DNS  
draft-sahib-domain-verification-techniques-03

Abstract

Many services on the Internet need to verify ownership or control of a domain in the Domain Name System (DNS) [RFC1034] [RFC1035]. This verification is often done by requesting a specific DNS record to be visible in the domain. This document surveys various techniques in wide use today, the pros and cons of each, and proposes some practises to avoid known problems.

Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at <https://github.com/ShivanKaul/draft-sahib-domain-verification-techniques>.

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 8 September 2022.

## Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Conventions and Definitions . . . . .	3
3. Verification Techniques . . . . .	3
3.1. TXT based . . . . .	3
3.1.1. Examples . . . . .	4
3.2. CNAME based . . . . .	5
3.2.1. Examples . . . . .	5
3.3. Common Patterns . . . . .	6
3.3.1. Name . . . . .	6
3.3.2. RDATA . . . . .	6
4. Recommendations . . . . .	6
4.1. Targeted Domain Verification . . . . .	6
4.2. Targeted Service Verification . . . . .	7
4.3. TXT vs CNAME . . . . .	7
4.4. Time-bound checking . . . . .	8
5. Email sending authorization . . . . .	9
6. Security Considerations . . . . .	9
7. Operational Considerations . . . . .	9
8. IANA Considerations . . . . .	10
9. References . . . . .	10
9.1. Normative References . . . . .	10
9.2. Informative References . . . . .	10
Acknowledgments . . . . .	11
Authors' Addresses . . . . .	11

## 1. Introduction

Many providers of internet services need domain owners to prove that they control a particular domain before they can operate a services or grant some privilege to the associated domain. For instance, certificate authorities like Let's Encrypt [LETSencrypt] ask requesters of TLS certificates to prove that they operate the domain they are requesting the certificate for. Providers generally allow for several different ways of proving domain control. This document describes and recommends common practises with using DNS based techniques for domain verification. Other techniques such as email or HTTP(S) based verification are out-of-scope.

In practice, DNS-based verification takes the form of the provider generating a random value visible only to the requester, and then asking the requester to create a DNS record containing this random value and placing it at a location within the domain that the provider can query for. Generally only one temporary DNS record is sufficient for proving domain ownership, although sometimes the DNS record must be kept in the zone to prove continued ownership of the domain.

## 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.

Provider: an internet-based provider of a service, for e.g., Let's Encrypt provides a certificate authority service or GitHub provides code-hosting services. These services often require a user to verify that they control a domain.

## 3. Verification Techniques

### 3.1. TXT based

TXT record-based DNS domain verification is usually the default option for DNS verification. The service provider asks the user to add a DNS TXT record (perhaps through their domain host or DNS provider) at the domain with a certain value. Then, the service provider does a DNS TXT query for the domain being verified and checks that the value exists. For example, this is what a DNS TXT verification record could look like:

```
example.com.    IN      TXT      "foo-verification=bar-237943648324687364"
```

Here, the value "bar-237943648324687364" for the attribute "foo-verification" serves as the randomly-generated TXT value being added to prove ownership of the domain to Foo provider. Although the original DNS protocol specifications did not associate any semantics with the DNS TXT record, [RFC1464] describes how to use them to store attributes in the form of ASCII text key-value pairs for a particular domain. In practice, there is wide variation in the content of DNS TXT records used for domain verification, and they often do not follow the key-value pair model. Even so, the rdata portion of the DNS TXT record has to contain the value being used to verify the domain. The value is usually a randomly-generated token in order to guarantee that the entity who requested that the domain be verified (i.e. the person managing the account at Foo provider) is the one who has (direct or delegated) access to DNS records for the domain. The generated token typically expires in a few days. The TXT record is placed at the domain being verified ("example.com" in the example above). After a TXT record has been added, the service provider will usually take some time to verify that the DNS TXT record with the expected token exists for the domain.

The same domain name can have multiple distinct TXT records (a TXT Record Set), where each TXT record may be associated with a distinct service. Having many of these may cause operational issues, and it is RECOMMENDED that providers use a prefix (eg "\_foo.example.com") instead of using the top of the domain ("APEX") directly, such as:

```
_foo.example.com. IN TXT "bar-237943648324687364"
```

### 3.1.1. Examples

#### 3.1.1.1. Let's Encrypt

Let's Encrypt [LETSencrypt] has a challenge type DNS-01 that lets a user prove domain ownership in accordance with the ACME protocol [RFC8555]. In this challenge, Let's Encrypt asks you to create a TXT record with a randomly-generated token at \_acme-challenge.<YOUR\_DOMAIN>. For example, if you wanted to prove domain ownership of example.com, Let's Encrypt could ask you to create the DNS record:

```
_acme-challenge.example.com. IN TXT "cE3A8qQpEzAIYq-T9DWNdLJ1_YRXamdxcjGTbz  
rOH5L"
```

[RFC8555] (section 8.4) places requirements on the random value.

#### 3.1.1.2. Google Workspace

[GOOGLE-WORKSPACE-TXT] asks the user to sign in with their administrative account and obtain their verification token as part of the setup process for Google Workspace. The verification token is a 68-character string that begins with "google-site-verification=", followed by 43 characters. Google recommends a TTL of 3600 seconds. The owner name of the TXT record is the domain or subdomain name being verified.

#### 3.1.1.3. GitHub

GitHub asks you to create a DNS TXT record under `_github-challenge-ORGANIZATION-<YOUR_DOMAIN>`, where ORGANIZATION stands for the GitHub organization name [GITHUB-TXT]. The code is a numeric code that expires in 7 days.

### 3.2. CNAME based

Less commonly than TXT record verification, service providers also provide the ability to verify domain ownership via CNAME records. One reason for using CNAME is for the case where the user cannot create TXT records. One common reason is that the domain name may already have a CNAME record that aliases it to a 3rd-party target domain. CNAMEs have a technical restriction that no other record types can be placed along side them at the same domain name ([RFC1034], Section 3.6.2).. The CNAME based domain verification method typically uses a randomized label prepended to the domain name being verified.

#### 3.2.1. Examples

##### 3.2.1.1. Google

[GOOGLE-WORKSPACE-CNAME] lets you specify a CNAME record for verifying domain ownership. The user gets a unique 12-character string that is added as "Host", with TTL 3600 (or default) and Destination an 86-character string beginning with "gv-" and ending with ".domainverify.googlehosted.com.".

To verify a subdomain, the unique 12-character string is appended with the subdomain name for "Host" field for e.g. `JLKDER712AFP.subdomain` where subdomain is the subdomain being verified.

#### 3.2.1.2. AWS Certificate Manager (ACM)

To get issued a certificate by AWS Certificate Manager (ACM), you can create a CNAME record to verify domain ownership [ACM-CNAME]. The record name for the CNAME looks like:

```
`\_<random-token1>.example.com.    IN    CNAME \_RANDOM-TOKEN.acm-validations.aws.
```

Note that if there are more than 5 CNAMEs being chained, then this method does not work.

### 3.3. Common Patterns

#### 3.3.1. Name

ACME and GitHub have a suffix of `_PROVIDER_NAME-challenge` in the Name field of the TXT record challenge. For ACME, the full Host is `_acme-challenge.<YOUR_DOMAIN>`, while for GitHub it is `_github-challenge-ORGANIZATION-<YOUR_DOMAIN>`. Both these patterns are useful for doing targeted domain verification, as discussed in section (#targeted-domain-verification) because if the provider knows what it is looking for (domain in the case of ACME, organization name + domain in case of GitHub) it can specifically do a DNS query for that TXT record, as opposed to having to do a TXT query for the apex.

ACME does the same name construction for CNAME records.

#### 3.3.2. RDATA

One pattern that quite a few providers follow (Dropbox, Atlassian) is constructing the rdata of the TXT DNS record in the form of `PROVIDER-SERVICE-domain-verification=` followed by the random value being checked for. This is in accordance with [RFC1464] which mandates that attributes must be stored as key-value pairs.

### 4. Recommendations

#### 4.1. Targeted Domain Verification

The TXT record being used for domain verification is most commonly placed at the domain name being verified. For example, if `example.com` is being verified, then the DNS TXT record will have `example.com` in the Name section. Unfortunately, this practise does not scale very well.

Many services are now attempting to verify domain names, causing many of these TXT records to be placed at that same location at the top of the domain (the APEX).

When a DNS administrator sees 15 DNS TXT records for their domain based on only random letters, they can no longer determine for which service or vendor the DNS TXT records were added. This causes administrators to leave all DNS TXT records in there, as they want to avoid breaking a service. Over time, the domain ends up with a lot of no longer needed, unknown and untracable DNS TXT records.

An operational issue arises from the DNS protocol only being able to query for "all TXT records" at a single location. If multiple services all require TXT records, this can cause the DNS answer for TXT records to become very large. It has been observed that some well known domains had so many services deployed that their DNS TXT answer did not fit in a single UDP DNS packet. This results in fragmentation which is known to be vulnerable to various attacks draft-ietf-dnsop-avoid-fragmentation-06. It can also lead to UDP packet truncation, causing a retry over TCP. Not all networks properly transport DNS over TCP and some DNS software mistakenly believe TCP support is optional draft-ietf-dnsop-dns-tcp-requirements-15.

#### 4.2. Targeted Service Verification

One malicious service that promises to deliver something after domain verification could surreptitiously ask another service provider to start processing or sending mail for the target domain and then present the victim domain administrator with this DNS TXT record pretending to be for their service. Once the administrator has added the DNS TXT record, instead of getting their service, their domain is now certifying another service of which they are not aware they are now a consumer.

If services use a clear description and name attribution in the required DNS TXT record, this can be avoided. For example by requiring a DNS TXT record at `_vendorname.example.com` instead of at `example.com`, a malicious service could no longer replay this without the DNS administrator noticing this. The LetsEncrypt ACME challenge uses this method.

#### 4.3. TXT vs CNAME

The inherent problem of a CNAME is that it cannot co-exist with any other data. What happens when both a CNAME and other data such as a TXT record or NS record exist depends on the DNS implementation. But most likely, either the CNAME or the other records will be silently ignored. The user interface for adding a record might not check for this. It might also break in unexpected ways. If a CNAME is added for continuous authorization, and for another service a TXT record is added, the TXT record might work but the CNAME record might break.



Operational experience has also shown a vendor that provides two difference services, one requiring a CNAME and one requiring a TXT record for authorization that needed to be deployed at the same location. If both services would have used a TXT record, this would not have caused any problems.

Another issues with CNAME records is that they MUST NOT point to another CNAME. But where this might be true in an initial deployment, if the target that the CNAME points to is changed from a non-CNAME record to a CNAME record, some DNS software might no longer resolve this as expected.

Early web based DNS administration tools did not always have the TXT record available in a pulldown menu for DNS record types, while CNAME would be available. However as many anti-spam meassures now require TXT records, this support is now generally available. It is recommended that the CNAME method is only used for delegating authorization to an actual subdomain, for example:

```
recrutement.example.com.    IN    CNAME    example.recrutement-vendor.com.
```

#### 4.4. Time-bound checking

After domain verification is done, there is no need for the TXT or CNAME record to continue to exist as the presence of the domain-verifying DNS record for a service only implies that a user with access to the service also has DNS control of the domain at the time the code was generated. It should be safe to remove the verifying DNS record once the verification is done and the service provider doing the verification should specify how long the verification will take (i.e. after how much time can the verifying DNS record be deleted). However, despite this, some services ask the record to exist in perpetuity [ATLASSIAN-VERIFY].

If a provider will use the DNS TXT record only for a one-time verification, it is RECOMMENDED that they clearly indicate this in the RDATA of the TXT record, so a DNS administrator at the target domain can easilly spot an obsolete record in the future. For example:

```
_provider-token.example.com.  IN  TXT  "type=activation_only  
expiry=2023-10-12 token=TOKENDATA"
```

If a provider requires the continued precense of the TXT record as proof that the domain owner is still authorizing the service, this should also be clear from the TXT record RDATA. For example:

\_provider-service.example.com. IN TXT "type=continued\_service  
expiry=never token=TOKENDATA"

## 5. Email sending authorization

Some vendors use a hosted service that wants to generate emails that appear to be from the customer. When a customer has deployed anti-spam measures such as DKIM [RFC6376], DMARC [RFC7489] or SPF [RFC7208], the vendor's mail service needs to be added to the list of allowed mail servers. However, some customers might not want to give permission for a vendor to send emails from their entire domain. It is recommended that a vendor uses a subdomain. If the vendor's domain is example-vendor.com, and the customer domain is example-customer.com, the vendor could use the subdomain example-customer.example-vendor.com to send emails. Alternatively, the customer could delegate a subdomain example-vendor.example-customer.com to the vendor for email sending, as those email addresses would have a stronger origin appearance of being emails sent by the customer to their clients.

Besides requiring proof of ownership of the domain, the customer needs to authorize the hosted service to send email on their behalf.

## 6. Security Considerations

Both the provider and the service being authenticated and authorized should be obvious from the TXT content to prevent malicious services from misleading the domain owner into certifying a different provider or service.

It is RECOMMENDED that DNSSEC [RFC4033] is employed by the domain owner. A service provider MUST enable DNSSEC validation when verifying domain name challenges to protect against domain name spoofing.

## 7. Operational Considerations

It is often consumers of the provider services that are not DNS experts that need to relay information from a provider's website to their local DNS administrators. The exact DNS record type, content and location is often not clear when the DNS administrator receives the information. It is RECOMMENDED that providers offer extremely detailed help pages, that are accessible without needing a login on the provider website, as the DNS administrator often has no login account on the provider service website. It is recommended that any instructions given by the provider contain the entire DNS record using a Fully Qualified Domain Name (FQDN).

## 8. IANA Considerations

This document has no IANA actions.

## 9. References

### 9.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/rfc/rfc1034>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/rfc/rfc1035>>.
- [RFC1464] Rosenbaum, R., "Using the Domain Name System To Store Arbitrary String Attributes", RFC 1464, DOI 10.17487/RFC1464, May 1993, <<https://www.rfc-editor.org/rfc/rfc1464>>.
- [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/rfc/rfc2119>>.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <<https://www.rfc-editor.org/rfc/rfc4033>>.
- [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/rfc/rfc8174>>.

### 9.2. Informative References

- [ACM-CNAME] AWS, "Option 1: DNS Validation", n.d., <<https://docs.aws.amazon.com/acm/latest/userguide/dns-validation.html>>.
- [ATLASSIAN-VERIFY] Atlassian, "Verify over DNS", n.d., <<https://support.atlassian.com/user-management/docs/verify-a-domain-to-manage-accounts/#Verifyadomainforyourorganization-VerifyoverDNS>>.

## [GITHUB-TXT]

GitHub, "Verifying your organization's domain", n.d.,  
<<https://docs.github.com/en/github/setting-up-and-managing-organizations-and-teams/verifying-your-organizations-domain>>.

## [GOOGLE-WORKSPACE-CNAME]

Google, "CNAME record values", n.d.,  
<<https://support.google.com/a/answer/112038>>.

## [GOOGLE-WORKSPACE-TXT]

Google, "TXT record values", n.d.,  
<<https://support.google.com/a/answer/2716802>>.

## [LETSencrypt]

Let's Encrypt, "Challenge Types: DNS-01 challenge", 2020,  
<<https://letsencrypt.org/docs/challenge-types/#dns-01-challenge>>.

[RFC6376] Crocker, D., Ed., Hansen, T., Ed., and M. Kucherawy, Ed.,  
"DomainKeys Identified Mail (DKIM) Signatures", STD 76,  
RFC 6376, DOI 10.17487/RFC6376, September 2011,  
<<https://www.rfc-editor.org/rfc/rfc6376>>.

[RFC7208] Kitterman, S., "Sender Policy Framework (SPF) for  
Authorizing Use of Domains in Email, Version 1", RFC 7208,  
DOI 10.17487/RFC7208, April 2014,  
<<https://www.rfc-editor.org/rfc/rfc7208>>.

[RFC7489] Kucherawy, M., Ed. and E. Zwicky, Ed., "Domain-based  
Message Authentication, Reporting, and Conformance  
(DMARC)", RFC 7489, DOI 10.17487/RFC7489, March 2015,  
<<https://www.rfc-editor.org/rfc/rfc7489>>.

[RFC8555] Barnes, R., Hoffman-Andrews, J., McCarney, D., and J.  
Kasten, "Automatic Certificate Management Environment  
(ACME)", RFC 8555, DOI 10.17487/RFC8555, March 2019,  
<<https://www.rfc-editor.org/rfc/rfc8555>>.

## Acknowledgments

TODO

## Authors' Addresses

Shivan Sahib  
Brave Software  
Email: [shivankaulsahib@gmail.com](mailto:shivankaulsahib@gmail.com)

Shumon Huque  
Salesforce  
Email: shuque@gmail.com

Paul Wouters  
Aiven  
Email: paul.wouters@aiven.io

DNSOP Working Group	P. Thomassen
Internet-Draft	deSEC, Secure Systems Engineering
Intended status: Standards Track	N. Wisiol
Expires: 2 June 2022	deSEC, Technische Universität Berlin
	29 November 2021

Automatic DNSSEC Bootstrapping using Authenticated Signals from the  
Zone's Operator  
draft-thomassen-dnsop-dnssec-bootstrapping-03

## Abstract

This document introduces an in-band method for DNS operators to publish arbitrary information about the zones they are authoritative for, in an authenticated fashion and on a per-zone basis. The mechanism allows managed DNS operators to securely announce DNSSEC key parameters for zones under their management, including for zones that are not currently securely delegated.

Whenever DS records are absent for a zone's delegation, this signal enables the parent's registry or registrar to cryptographically validate the CDS/CDNSKEY records found at the child's apex. The parent can then provision DS records for the delegation without resorting to out-of-band validation or weaker types of cross-checks such as "Accept after Delay" ([RFC8078]).

This document updates [RFC8078] and replaces its Section 3 with Section 3.2 of this document.

[ Ed note: Text inside square brackets ([ ]) is additional background information, answers to frequently asked questions, general musings, etc. They will be removed before publication. This document is being collaborated on at <https://github.com/desec-io/draft-thomassen-dnsop-dnssec-bootstrapping/> (<https://github.com/desec-io/draft-thomassen-dnsop-dnssec-bootstrapping/>). The most recent version of the document, open issues, etc. should all be available there. The authors gratefully accept pull requests. ]

## 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 2 June 2022.

#### Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

#### Table of Contents

1. Introduction . . . . .	3
1.1. Terminology . . . . .	4
1.2. Requirements Notation . . . . .	5
2. Signaling . . . . .	5
2.1. Chain of Trust . . . . .	5
2.2. Signaling Names . . . . .	5
3. Bootstrapping a DNSSEC Delegation . . . . .	5
3.1. Signaling Consent to Act as the Child's Signer . . . . .	6
3.1.1. Example . . . . .	6
3.2. Validating CDS/CDNSKEY Records for DNSSEC Bootstrapping . . . . .	6
3.2.1. Example . . . . .	7
3.3. Triggers . . . . .	8
3.4. Limitations . . . . .	9
4. Operational Recommendations . . . . .	9
4.1. Child DNS Operator . . . . .	9
4.2. Parental Agent . . . . .	10
5. Implementation Status . . . . .	10
5.1. Child DNS Operator-side . . . . .	10
5.2. Parental Agent-side . . . . .	10
6. Security Considerations . . . . .	11
7. IANA Considerations . . . . .	11
8. Acknowledgements . . . . .	12
9. Normative References . . . . .	12
Appendix A. Change History (to be removed before publication) .	13

Authors' Addresses	14
--------------------	----

## 1. Introduction

Securing a DNS delegation for the first time requires that the Child's DNSSEC parameters be conveyed to the Parent through some trusted channel. How exactly this is done depends on the relationship the Child has with the Parent.

In general, the communication has to occur between the DNSSEC key holder and the Parent registry. It is often the case that the key is held by the Child DNS Operator. Furthermore, depending on the circumstances, the communication may also involve the Registrar, possibly via the Registrant (for details, see [RFC7344], Appendix A).

As observed in [RFC7344], this is often a manual process -- and not an easy one. Any manual process is susceptible to mistakes and/or errors. In addition, due to the annoyance factor of the process, involved parties may avoid the process of getting a DS record set published in the first place.

To alleviate these problems, automated provisioning of DS records has been specified in ([RFC8078]). It is based on the Parental Agent (registry or registrar) fetching DNSSEC key parameters in the form of CDS and CDNSKEY records ([RFC7344]) from the Child zone's apex, and validating them somehow. This validation can be done using DNSSEC itself if the objective is to update an existing DS record (such as during key rollover). However, when bootstrapping a DNSSEC delegation, the Child zone has no existing DNSSEC validation path, and other means to ensure the CDS/CDNSKEY records' legitimacy must be found.

For lack of a comprehensive DNS-innate solution, either out-of-band methods have been used so far to complete the chain of trust, or cryptographic validation has been entirely dispensed with, in exchange for weaker types of cross-checks such as "Accept after Delay" ([RFC8078] Section 3.3). An in-band validation method for enabling DNSSEC has been missing.

This document aims to close this gap by introducing an in-band method for DNS Operators to publish arbitrary information about the zones they are authoritative for, in an authenticated manner and on a per-zone basis. The mechanism allows managed DNS Operators to securely announce DNSSEC key parameters for zones under their management. The Parent can then use this signal to cryptographically validate the CDS/CDNSKEY records found at an insecure Child zone's apex, and upon success secure the delegation.



While applicable to the vast majority of domains, the protocol does not support certain edge cases, such as excessively long Child zone names, or DNSSEC bootstrapping for in-bailiwick domains (see Section 3.4).

Readers are expected to be familiar with DNSSEC, including [RFC4033], [RFC4034], [RFC4035], [RFC6781], [RFC7344], and [RFC8078].

### 1.1. Terminology

This section defines the terminology used in this document.

**Child** The entity on record that has the delegation of the domain from the Parent.

**Parent** The domain in which the Child is registered.

**Child DNS Operator** The entity that maintains and publishes the zone information for the Child DNS.

**Parental Agent** The entity that has the authority to insert DS records into the Parent zone on behalf of the Child. (It could be the registry, registrar, a reseller, or some other authorized entity.)

**Signaling Domain** A hostname from the Child's NS record set, prefixed with the label `_dsauth`. There are as many Signaling Domains as there are distinct NS targets.

**Signaling Zone** The zone which is authoritative for a given Signaling Domain.

**Signaling Name** The labels that are prefixed to a Signaling Domain in order to identify a Child zone's name (see Section 2.2).

**Signaling Record** A DNS record located at a Signaling Name under a Signaling Domain. Signaling Records are used by the Child DNS Operator to publish information about the Child.

**CDS/CDNSKEY** This notation refers to CDS and/or CDNSKEY, i.e., one or both.

**Base32hex Encoding** "Base 32 Encoding with Extended Hex Alphabet" as per [RFC4648].

## 1.2. Requirements Notation

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.

## 2. Signaling

When setting up initial trust, the Child zone's CDS/CDNSKEY RRsets need to be authenticated. This is achieved using an authentication signal from the Child DNS Operator that the Parent can discover and validate, thus transferring trust from the Child DNS Operator to the Child zone.

### 2.1. Chain of Trust

If a Child DNS Operator implements the protocol, each Signaling Zone MUST be securely delegated, i.e. have a valid DNSSEC chain of trust.

For example, when performing DNSSEC bootstrapping for a Child zone with NS records ns1.example.net and ns2.example.net, the Child DNS Operator needs to ensure that a valid DNSSEC chain of trust exists for the zone(s) that are authoritative for the Signaling Domains \_dsauth.ns1.example.net and \_dsauth.ns2.example.net.

### 2.2. Signaling Names

To publish a piece of information about the Child zone in an authenticated fashion, the Child DNS Operator MUST publish one or more Signaling Records at the Child's Signaling Name under each Signaling Domain.

Signaling Records MUST be accompanied by RRSIG records created with the corresponding Signaling Zone's key(s). The type and contents of these Signaling Records are detailed in Section 3.1.

The Signaling Name is identical to the Child name, with the final root label removed.

## 3. Bootstrapping a DNSSEC Delegation

Child DNS Operators and Parental Agents who wish to use CDS/CDNSKEY records for DNSSEC bootstrapping SHOULD support the protocol described in this section.

### 3.1. Signaling Consent to Act as the Child's Signer

To confirm its willingness to act as the Child's delegated signer and authenticate the Child's CDS/CDNSKEY RRsets, the Child DNS Operator MUST co-publish them at the corresponding Signaling Name under each Signaling Domain as defined in Section 2.2.

Existing use of CDS/CDNSKEY records is specified at the Child apex only ([RFC7344], Section 4.1). This protocol extends the use of these record types to non-apex owner names for the purpose of DNSSEC bootstrapping. To exclude the possibility of semantic collision, there MUST NOT be a zone cut at a Signaling Name.

Unlike the CDS/CDNSKEY records at the Child's apex, Signaling Records MUST be signed with the corresponding Signaling Zone's key(s). Their contents MUST be identical to the corresponding records published at the Child's apex.

#### 3.1.1. Example

For the purposes of bootstrapping the Child zone example.co.uk with NS records ns1.example.net and ns2.example.net, the required Signaling Domains are \_dsauth.ns1.example.net and \_dsauth.ns2.example.net.

In the zones containing these domains, the Child DNS Operator authenticates the CDS/CDNSKEY records found at the Child's apex by co-publishing them at the names:

```
example.co.uk._dsauth.ns1.example.net
example.co.uk._dsauth.ns2.example.net
```

The records are accompanied by RRSIG records created using the key(s) of the respective Signaling Zone.

### 3.2. Validating CDS/CDNSKEY Records for DNSSEC Bootstrapping

This section replaces Section 3 of [RFC8078].

To validate a Child's CDS/CDNSKEY RRset for DNSSEC bootstrapping, the Parental Agent, knowing both the Child zone name and its NS hostnames, MUST execute the following steps:

1. verify that the Child is not currently securely delegated;
2. query the CDS/CDNSKEY records at the Child zone apex directly from each of the authoritative servers listed in the delegation's NS record set;

3. query the CDS/CDNSKEY records located at the Signaling Name under each Signaling Domain using a trusted DNS resolver and enforce DNSSEC validation;
4. check (separately by record type) that all record sets retrieved in Steps 2 and 3 have equal contents;

If the above steps succeed without error, the CDS/CDNSKEY records are successfully validated, and the Parental Agent can proceed with the publication of the DS record set under the precautions described in [RFC8078], Section 5.

If, however, an error condition occurs, in particular:

- \* in Step 1: the Child is already securely delegated;
- \* in Step 2: any failure during the retrieval of the CDS/CDNSKEY records located at the Child apex from any of the authoritative nameservers, with an empty record set qualifying as a failure;
- \* in Step 3: any failure to retrieve the CDS/CDNSKEY RRsets located at the Signaling Name under any Signaling Domain, including failure of DNSSEC validation, unauthenticated data (AD bit not set), or an empty record set;
- \* in Step 4: inconsistent responses;

the Parental Agent MUST abort the procedure.

#### 3.2.1. Example

To verify the CDS/CDNSKEY records for the Child `example.co.uk`, the Parental Agent (assuming that the Child delegation's NS records are `ns1.example.net` and `ns2.example.net`)

1. checks that the Child zone is not yet securely delegated;
2. queries CDS/CDNSKEY records for `example.co.uk` directly from `ns1.example.net` and `ns2.example.net`;
3. queries the CDS/CDNSKEY records located at (see Section 2.2)

`example.co.uk._dsauth.ns1.example.net`  
`example.co.uk._dsauth.ns2.example.net`

4. checks that the CDS/CDNSKEY record sets retrieved in Steps 2 and 3 agree across responses.

If all these steps succeed, the Parental Agent can proceed to publish a DS record set as indicated by the validated CDS/CDNSKEY records.

### 3.3. Triggers

[ Clarity of this section needs to be improved. ]

Parental Agents SHOULD trigger the procedure described in Section 3.2 once one of the following conditions is fulfilled:

- \* The Parental Agent receives a new or updated NS record set for a Child;
- \* The Parental Agent encounters Signaling Records for its Children during a scan (e.g. daily) of known Signaling Domains (derived from the NS records found in the Parent zone).

To perform such a scan, the Parental Agent iterates over some or all of its delegations and strips the first label off each one to construct the set of immediate ancestors of its children. (For delegations one level below the Parent's apex, such as second-level domain registrations, this will simply be the name of the Parent zone.) The Parental Agent then uses these names to compute the second label of the Signaling Names as described in Section 2.2. The scan is completed by either

- performing a targeted NSEC walk starting one level below the Signaling Domain, at the label that encodes the Child's ancestor; or by
- performing a zone transfer of the zone containing the (relevant part of the) Signaling Domain, if the Signaling Zone operator allows it, and iterating over its contents.

The Child's name is constructed by prepending the first label of the encountered Signaling Names to the ancestor name from which the Signaling Name's second label was computed;

- \* The Parental Agent encounters Signaling Records during a proactive, opportunistic scan (e.g. daily queries for the Signaling Records of some or all of its delegations);
- \* Any other condition as deemed appropriate by local policy.

One of the inputs of the bootstrapping algorithm in Section 3.2 is the NS record set of the Child's delegation. It is therefore necessary to establish knowledge of the delegation's NS record set before firing the trigger.

In some cases, the trigger context contains reliable information about the Child's delegation, such as when bootstrapping is triggered by the registrant changing their NS record set, or during a daily scan of existing delegations. In such cases, the delegation's NS RRset can be used as is.

In cases where the trigger context does not provide sufficient knowledge of the NS record set, the Parental Agent MUST fetch the delegation's NS record set and ensure that the proper NS record set is fed to the bootstrapping algorithm (Section 3.2).

In particular, when discovering Signaling Names by means of an NSEC walk or zone transfer, the Parental Agent MUST NOT assume that the nameserver(s) under whose Signaling Domain(s) a Signaling Name is discovered is in fact authoritative for the corresponding Child. Before firing the trigger for a particular candidate Child, the Parental Agent MUST ascertain that the Child's delegation actually contains the nameserver hostname under whose Signaling Domain the scan occurred.

### 3.4. Limitations

As a consequence of Step 3 in Section 3.2, DS bootstrapping does not work for in-bailiwick delegations, as no pre-existing chain of trust to the Child domain is available during bootstrapping.

The protocol is further restricted by the fact that the fully qualified Signaling Names fit within the general limits that apply to DNS names (such as their length and their label count).

## 4. Operational Recommendations

### 4.1. Child DNS Operator

To keep the size of the Signaling Zones minimal and bulk processing efficient (such as via NSEC walks or zone transfers), Child DNS Operators SHOULD remove Signaling Records which are found to have been acted upon.

Signaling Domains SHOULD be delegated as zones of their own, so that the Signaling Zone's apex coincides with the Signaling Domain (such as `_dsauth.nsl.example.net`). While it is permissible for the Signaling Domain to be contained in a Signaling Zone of fewer labels (such as `example.net`), a zone cut ensures that bootstrapping activities do not require modifications of the zone containing the nameserver hostname.

In addition, Signaling Zones SHOULD use NSEC to allow efficient discovery of pending bootstrapping operations by means of zone walking (see Section 3.3). This is especially useful for bulk processing after a Child DNS Operator has enabled the protocol.

#### 4.2. Parental Agent

It is RECOMMENDED to perform queries within Signaling Domains (Section 3.2) with an (initially) cold resolver cache as to retrieve the most current information regardless of TTL. (When a batch job is used to attempt bootstrapping for a large number of delegations, the cache does not need to get cleared in between.)

[It is expected that Signaling Records have few consumers only, so that caching would not normally have a performance benefit. On the other hand, perhaps it is better to RECOMMEND low TTLs instead?]

### 5. Implementation Status

\*Note to the RFC Editor\*: please remove this entire section before publication.

#### 5.1. Child DNS Operator-side

- \* Knot DNS supports manual creation of non-apex CDS/CDNSKEY records.
- \* PowerDNS supports manual creation of non-apex CDS/CDNSKEY records.
- \* Proof-of-concept Signaling Domains with several thousand Signaling Names exist at [\\_dsauth.nsl.desec.io](https://dsauth.nsl.desec.io) and [\\_dsauth.ns2.desec.org](https://dsauth.ns2.desec.org). Signaling Names can be discovered via NSEC walking. Some other operators are planning an experimental implementation.
- \* A tool to automatically generate signaling records for bootstrapping purposes is under development by the authors.

#### 5.2. Parental Agent-side

- \* A tool to retrieve and process Signaling Records for bootstrapping purposes, either directly or via zone walking, is available at <https://github.com/desec-io/dsbootstrap> (<https://github.com/desec-io/dsbootstrap>). The tool outputs the validated DS records which then can be added to the Parent zone.
- \* Some registries/registrars are planning experimental implementations of the protocol.

## 6. Security Considerations

The protocol adds authentication to the CDS/CDNSKEY-based bootstrapping concept of [RFC8078], while removing nothing. Its security level is therefore strictly higher than that of existing approaches described in that document (e.g. "Accept after Delay"). Apart from this general improvement, the same Security Considerations apply as in [RFC8078].

The level of rigor in Section 3.2 is needed to prevent publication of a half-backed DS RRset (authorized only under a subset of NS hostnames). This ensures that an operator in a multi-homed setup cannot enable DNSSEC unless all other operators agree. [ TODO In principle, this applies to any CDS update. Should we phrase it as a general update to [RFC8078]? ]

[ Thoughts on the Chain of Trust:

Actors in the chain(s) of trust of the Signaling Zone(s) (the DNS Operator themselves, plus entities further up in the chain) can undermine the protocol. However,

- \* that's possible with CDS/CDNSKEY, too (new method is not weaker);
- \* if the Child DNS Operator doesn't trust the zones in which its NS hostnames live (including their nameservers' A records) because their path from the root is untrusted, you probably don't want to trust that operator as a whole;
- \* when bootstrapping is done upon receipt of a new NS record set, the window of opportunity is very small;
- \* mitigation exists by diversifying e.g. the nameserver hostname's TLDs, which is advisable anyways;
- \* correct bootstrapping is easily monitored by the Child DNS Operator.

]

## 7. IANA Considerations

\*TODO:\* reserve \_dsauth?

This document has no IANA actions.



## 8. Acknowledgements

Thanks to Brian Dickson, Ondrej Caletka, John R. Levine, Christian Elmerot, and Oli Schacher for reviewing draft proposals and offering comments and suggestions.

Thanks also to Steve Crocker, Hugo Salgado, and Ulrich Wisser for early-stage brainstorming.

## 9. Normative References

- [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>>.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <<https://www.rfc-editor.org/info/rfc4033>>.
- [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>>.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <<https://www.rfc-editor.org/info/rfc4035>>.
- [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>>.
- [RFC6781] Kolkman, O., Mekking, W., and R. Gieben, "DNSSEC Operational Practices, Version 2", RFC 6781, DOI 10.17487/RFC6781, December 2012, <<https://www.rfc-editor.org/info/rfc6781>>.
- [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>>.
- [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>>.

#### Appendix A. Change History (to be removed before publication)

\* draft-thomassen-dnsop-dnssec-bootstrapping-03

Clarified importance of record cleanup by moving paragraph up.

Pointed out limitations.

Replace [RFC8078] Section 3 with our Section 3.2.

Changed `_boot` label to `_dsauth`.

Removed hashing of Child name components in Signaling Names.

Editorial changes.

\* draft-thomassen-dnsop-dnssec-bootstrapping-02

Reframed as an authentication mechanism for RFC 8078.

Removed multi-signer use case (focus on RFC 8078 authentication).

Triggers need to fetch NS records (if not implicit from context).

Improved title.

Recognized that hash collisions are dealt with by Child apex check.

\* draft-thomassen-dnsop-dnssec-bootstrapping-01

Add section on Triggers.

Clarified title.

Improved abstract.

Require CDS/CDNSKEY records at the Child.

| Reworked Signaling Name scheme.

| Recommend using cold cache for consumption.

| Updated terminology (replace "Bootstrapping" by "Signaling").

| Added NSEC recommendation for Bootstrapping Zones.

| Added multi-signer use case.

| Editorial changes.

\* draft-thomassen-dnsop-dnssec-bootstrapping-00

| Initial public draft.

#### Authors' Addresses

Peter Thomassen  
deSEC, Secure Systems Engineering  
Berlin  
Germany

Email: peter@desec.io

Nils Wisiol  
deSEC, Technische Universität Berlin  
Berlin  
Germany

Email: nils@desec.io

DNSOP WG  
Internet-Draft  
Updates: 8914 (if approved)  
Intended status: Standards Track  
Expires: 29 October 2022

D. Wing  
Citrix  
T. Reddy  
Akamai  
N. Cook  
Open-Xchange  
M. Boucadair  
Orange  
27 April 2022

Structured Data for Filtered DNS  
draft-wing-dnsop-structured-dns-error-page-03

Abstract

DNS filtering is widely deployed for network security, but filtered DNS responses lack information for the end user to understand the reason for the filtering. Existing mechanisms to provide detail to end users cause harm especially if the blocked DNS response is to an HTTPS website.

This document updates RFC8914's EXTRA-TEXT field to provide information on DNS filtering. This information can be parsed by the client and displayed, logged, or used for other purposes.

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 29 October 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	5
3. I-JSON in EXTRA-TEXT field . . . . .	6
4. Protocol Operation . . . . .	6
4.1. Client Generating Request . . . . .	6
4.2. Server Generating Response . . . . .	7
4.3. Client Processing Response . . . . .	7
5. Examples . . . . .	8
6. Security Considerations . . . . .	9
7. IANA Considerations . . . . .	9
8. Changes . . . . .	10
9. References . . . . .	11
9.1. Normative References . . . . .	11
9.2. Informative References . . . . .	12
Authors' Addresses . . . . .	13

## 1. Introduction

DNS filters are deployed for a variety of reasons including endpoint security, parental filtering, and filtering required by law enforcement. Network-based security solutions such as firewalls and Intrusion Prevention Systems (IPS) rely upon network traffic inspection to implement perimeter-based security policies and operate by filtering DNS responses. In a home, DNS filtering is used for the same reasons as above and additionally for parental control. Internet Service Providers typically block access to some DNS domains due to a requirement imposed by an external entity (e.g., law enforcement agency) also performed using DNS-based content filtering.

Users of DNS services which perform filtering may wish to receive more information about such filtering to resolve problems with the filter -- for example to contact the administrator to allowlist a domain that was erroneously filtered or to understand the reason a particular domain was filtered. With that information, the user can choose another network, open a trouble ticket with the DNS administrator to resolve erroneous filtering, log the information, or other uses.

DNS responses can be filtered by sending a bogus (also called, "forged") A or AAAA response, NXDOMAIN error or empty answer, or an extended DNS error (EDE) code defined in [RFC8914]. Each of these methods have advantages and disadvantages that are discussed below:

1. The DNS response is forged to provide a list of IP addresses that points to an HTTP(S) server alerting the end user about the reason for blocking access to the requested domain (e.g., malware). When an HTTP(S) enabled domain name is blocked, the network security device (e.g., CPE, firewall) presents a block page instead of the HTTP response from the content provider hosting that domain. If an HTTP enabled domain name is blocked, the network security device intercepts the HTTP request and returns a block page over HTTP. If an HTTPS enabled domain is blocked, the block page is also served over HTTPS. In order to return a block page over HTTPS, man in the middle (MITM) is enabled on endpoints by generating a local root certificate and an accompanying (local) public/private key pair. The local root certificate is installed on the endpoint while the network security device(s) stores a copy of the private key. During the TLS handshake, the network security device modifies the certificate provided by the server and (re)signs it using the private key from the local root certificate.
  - \* However, configuring the local root certificate on endpoints is not a viable option in several deployments like home networks, schools, Small Office/Home Office (SOHO), and Small/Medium Enterprise (SME). In these cases, the typical behavior is that the filtered DNS response points to a server that will display the block page. If the client is using HTTPS (via web browser or another application) this results in a certificate validation error which gives no information to the end-user about the reason for the DNS filtering. Browsers will display errors such as "The security certificate presented by this website was not issued by a trusted certificate authority" (Internet Explorer/Edge), "The site's security certificate is not trusted" (Chrome), "This Connection is Untrusted" (Firefox), "Safari can't verify the identity of the website..." (Safari on MacOS). Applications might display even more cryptic error messages.
  - \* Enterprise networks do not assume that all the connected devices are managed by the IT team or Mobile Device Management (MDM) devices, especially in the quite common Bring Your Own Device (BYOD) scenario. In addition, the local root certificate cannot be installed on IoT devices without a device management tool.

- \* An end user does not know why the connection was prevented and, consequently, may repeatedly try to reach the domain but with no success. Frustrated, the end user may switch to an alternate network that offers no DNS filtering against malware and phishing, potentially compromising both security and privacy. Furthermore, certificate errors train users to click through certificate errors, which is a bad security practice. To eliminate the need for an end user to click through certificate errors, an end user may manually install a local root certificate on a host device. Doing so, however, is also a bad security practice as it creates a security vulnerability that may be exploited by a MITM attack. When a manually installed local root certificate expires, the user has to (again) manually install the new local root certificate.
- 2. The DNS response is forged to provide a NXDOMAIN response to cause the DNS lookup to terminate in failure. In this case, an end user does not know why the domain cannot be reached and may repeatedly try to reach the domain but with no success. Frustrated, the end user may use insecure connections to reach the domain, potentially compromising both security and privacy.
- 3. The extended error codes Blocked, Censored, and Filtered defined in Section 4 of [RFC8914] can be returned by a DNS server to provide additional information about the cause of a DNS error. If the extended error code "Forged Answer" defined in Section 4.5 of [RFC8914] is returned by the DNS server, the client can identify the DNS response is forged together with the reason for HTTPS certificate error.
- 4. These extended error codes do not suffer from the limitations discussed in bullets (1) and (2), but the user still does not know the exact reason nor he/she is aware of the exact entity blocking the access to the domain. For example, a DNS server may block access to a domain based on the content category such as "Adult Content" to enforce parental control, "Violence & Terrorism" due to an external requirement imposed by an external entity (e.g., Law Enforcement Agency), etc. These content categories cannot be standardized because the classification of domains into content categories is vendor specific, typically ranges from 40 to 100 types of categories depending on the vendor and the categories keep evolving. Furthermore, the threat data used to categorize domains may sometimes misclassify domains (e.g., domains wrongly classified as Domain Generation Algorithm (DGA) by deep learning techniques, domain wrongly classified as phishing due to crowd sourcing, new domains not categorized by the threat data). A user needs to know the contact details of the IT/InfoSec team to raise a complaint.

5. When a resolver or forwarder forwards the received EDE option, the EXTRA-TEXT field only conveys the source of the error (Section 3 of [RFC8914]) and does not provide additional textual information about the cause of the error.

For both DNS filtering mechanisms described above, the DNS server can return extended error codes Blocked, Censored, Filtered, or Forged Answer defined in Section 4 of [RFC8914]. However, these codes only explain that filtering occurred but lack detail for the user to diagnose erroneous filtering.

No matter which type of response is generated (forged IP address(es), NXDOMAIN or empty answer, even with an extended error code), the user who triggered the DNS query has little chance to understand which entity filtered the query, how to report a mistake in the filter, or why the entity filtered it at all. This document describes a mechanism to provide such detail.

One of the other benefits of this approach is to eliminate the need to "spoof" block pages for HTTPS resources. This is achieved since clients implementing this approach would be able to display a meaningful error message, and would not need to connect to such a block page. This approach thus avoids the need to install a local root certificate authority on those IT-managed devices.

This document describes a format for computer-parsable data in the EXTRA-TEXT field of Extended DNS Errors [RFC8914].

This document does not recommend DNS filtering but provides a mechanism for better transparency to explain to the users why some DNS queries are filtered.

## 2. Terminology

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.

This document uses terms defined in DNS Terminology [RFC8499].

"Requestor" refers to the side that sends a request. "Responder" refers to an authoritative, recursive resolver or other DNS component that responds to questions. Other terminology is used here as defined in the RFCs cited by this document.



"Encrypted DNS" refers to any encrypted scheme to convey DNS messages, for example, DNS-over-HTTPS [RFC8484], DNS-over-TLS [RFC7858], or DNS-over-QUIC [I-D.ietf-dprive-dnssoquic].

### 3. I-JSON in EXTRA-TEXT field

Servers compliant with this specification send I-JSON data in the EXTRA-TEXT field [RFC8914] using the Internet JSON (I-JSON) message format [RFC7493].

Note that [RFC7493] was based on [RFC7159], but [RFC7159] was replaced by [RFC8259].

This document defines the following JSON names:

- c: (contact) The contact details of the IT/InfoSec team to report mis-classified DNS filtering. This field is structured as an array of contact URIs (e.g., tel, sips, https). At least one contact URI MUST be included. This field is mandatory.
- j: (justification) the textual justification for this particular DNS filtering. This field is mandatory.
- o: (organization) human-friendly name of the organization that filtered this particular DNS query. This field is optional.

New JSON names MUST be defined in the IANA registry (Section 7), consist only of lower-case ASCII characters, digits, and hyphens (that is, Unicode characters U+0061 through 007A, U+0030 through U+0039, and U+002D). These names MUST be 63 characters or shorter and it is RECOMMENDED they be as short as possible.

To reduce packet overhead the generated JSON SHOULD be as short as possible: short domain names, concise text in the values for the "j" and "o" names, and minified JSON (that is, without spaces or line breaks between JSON elements).

The JSON data can be parsed to display to the user, logged, or otherwise used to assist the end-user or IT staff with troubleshooting and diagnosing the cause of the DNS filtering.

## 4. Protocol Operation

### 4.1. Client Generating Request

When generating a DNS query, the client MUST include the OPT pseudo-RR [RFC6891] to elicit the Extended DNS Error option [RFC8914] in the DNS response.

#### 4.2. Server Generating Response

When the DNS server filters its DNS response to an A or AAAA record query, the DNS response MAY contain an empty answer, NXDOMAIN, or a forged A or AAAA response, as desired by the DNS server. In addition, if the query contained the OPT pseudo-RR the DNS server MAY return more detail in the EXTRA-TEXT field as described in Section 4.3.

Servers may decide to return small TTL values in filtered DNS responses (e.g., 2 seconds) to handle domain category and reputation updates.

#### 4.3. Client Processing Response

On receipt of a DNS response with an Extended DNS Error option, the following actions are performed if the EXTRA-TEXT field contains valid JSON:

- \* The response MUST be received over an encrypted DNS channel. If not, the requestor MUST discard data in the EXTRA-TEXT field.
- \* The response MUST be received from a DNS server which advertised EDE support via RESINFO [I-D.reddy-add-resolver-info].
- \* Servers which don't support this specification might use plain text in the EXTRA-TEXT field so that requestors SHOULD properly handle both plaintext and JSON text in the EXTRA-TEXT field.
- \* The DNS response MUST also contain an extended error code of "Censored", "Blocked", "Filtered" or "Forged" [RFC8914], otherwise the EXTRA-TEXT field is discarded.
- \* If either of the mandatory JSON names "c" and "j" are missing or have empty values in the EXTRA-TEXT field, the entire JSON is discarded.
- \* If a DNS client has enabled opportunistic privacy profile (Section 5 of [RFC8310]) for DoT, the DNS client will either fallback to an encrypted connection without authenticating the DNS server provided by the local network or fallback to clear text DNS, and cannot exchange encrypted DNS messages. Both of these fallback mechanisms adversely impacts security and privacy. If the DNS client has enabled opportunistic privacy profile for DoT, the DNS client MUST ignore the EXTRA-TEXT field of the EDE responses, but SHOULD process other parts of the response.

- \* If a DNS client has enabled strict privacy profile (Section 5 of [RFC8310]) for DoT, the DNS client requires an encrypted connection and successful authentication of the DNS server; this mitigates both passive eavesdropping and client redirection (at the expense of providing no DNS service if an encrypted, authenticated connection is not available). If the DNS client has enabled strict privacy profile for DoT, the client MAY process the EXTRA-TEXT field of the DNS response. Note that the strict and opportunistic privacy profiles as defined in [RFC8310] only apply to DoT; there has been no such distinction made for DoH.
- \* If the DNS client determines that the encrypted DNS server does not offer DNS filtering service, it MUST discard the EXTRA-TEXT field of the EDE response. For example, the DNS client can learn whether the encrypted DNS resolver performs DNS-based content filtering or not by retrieving resolver information using the method defined in [I-D.reddy-add-resolver-info].
- \* When a forwarder receives an EDE option, whether or not (and how) to pass along JSON information in the EXTRA-TEXT on to their client is implementation dependent [RFC5625]. Implementations MAY choose to not forward the JSON information, or they MAY choose to create a new EDE option that conveys the information in the "c" and "j" fields encoded in the JSON object.

## 5. Examples

An example showing the nameserver at 'ns.example.net' that filtered a DNS "A" record query for 'example.org' is shown in Figure 1.

```
{
  "c": ["tel:+358-555-1234567", "sips:bob@bobphone.example.com",
        "https://ticket.example.com?d=example.org&t=1650560748"],
  "j": "malware present for 23 days",
  "o": "example.net Filtering Service"
}
```

Figure 1: JSON returned in EXTRA-TEXT field of Extended DNS Error response

In Figure 2 the same content is shown with minified JSON (no whitespace, no blank lines) with '\\' line wrapping per [RFC8792].

```
===== NOTE: '\ ' line wrapping per RFC 8792 =====  
  
{"c":["tel:+358-555-1234567","sips:bob@bobphone.example.com", \  
"https://ticket.example.com?d=example.org&t=1650560748"], \  
"j":"malware present for 23 days","o":"example.net Filtering \  
Service"}
```

Figure 2: Minified response

## 6. Security Considerations

Security considerations in Section 6 of [RFC8914] apply to this document.

To minimize impact of active on-path attacks on the DNS channel, the client validates the response as described in Section 4.3.

A client might choose to display the information in the EXTRA-TEXT field if and only if the encrypted resolver has sufficient reputation, according to some local policy (e.g. user configuration, administrative configuration, or a built-in list of respectable resolvers). This limits the ability of a malicious encrypted resolver to cause harm. If the client decides not to display the all of the information in the EXTRA-TEXT field, it can be logged for diagnostics purpose and the client can only display the resolver hostname that blocked the domain and error description for the EDE code to the end-user.

When displaying the free-form text of "c" and "j", the browser SHOULD NOT make any of those elements into actionable (clickable) links.

An attacker might inject (or modify) the EDE EXTRA-TEXT field with an DNS proxy or DNS forwarder that is unaware of EDE. Such a DNS proxy or DNS forwarder will forward that attacker-controlled EDE option. To prevent such an attack, clients supporting this document MUST discard the EDE option if their DNS server does not signal EDE support via RESINFO [I-D.reddy-add-resolver-info]. As recommended in [I-D.reddy-add-resolver-info], RESINFO should be retrieved over an encrypted DNS channel or integrity protected with DNSSEC.

## 7. IANA Considerations

This document requests IANA to register the "application/json+structured-dns-error" media type in the "Media Types" registry [IANA-MediaTypes]. This registration follows the procedures specified in [RFC6838]:

Type name: application

Subtype name: json+structured-dns-error

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: as defined in Section NN of [RFCXXXX].

Security considerations: See Section NNN of [RFCXXXX].

Interoperability considerations: N/A

Published specification: [RFCXXXX]

Applications that use this media type: Section NNNN of [RFCXXXX].

Fragment identifier considerations: N/A

Additional information: N/A

Person & email address to contact for further information: IETF,  
iesg@ietf.org

Intended usage: COMMON

Restrictions on usage: none

Author: See Authors' Addresses section.

Change controller: IESG

Provisional registration? No

## 8. Changes

This section is to be removed before publishing as an RFC.

### 8.1. Changes from 02 to 03

- \* Require using RESINFO [I-D.reddy-add-resolver-info] in client processing and added discussion of attack mitigation of using RESINFO.

- \* Removed validation of URI domain suffix, which we can't do for some URLs (e.g., tel:), is difficult/impossible for others when 3rd party is handling level one support (e.g., sips:). Instead rely on RESINFO telling us if EDE is supported by the DNS server and, if so, expect it to properly support EDE rather than blindly forward an unknown DNS option.
- \* Removed 'partial URI' text

## 8.2. Changes from 01 to 02

- \* repurpose Extended DNS Error's EXTRA-TEXT field to carry JSON, which also means this document updates RFC8914
- \* clarified DNS forwarders might forward EXTRA-TEXT without change or might rewrite "j" and "d"

## 8.3. Changes from 00 to 01

- \* removed support for multiple responsible parties
- \* one-character JSON names to minimize JSON length
- \* partial URI sent in "c" and "r" names, combined with "d" name sent in JSON to minimize attack surface and minimize JSON length
- \* moved EDNS(0) forgery-mitigation text, some Security Considerations text, and some other text from [I-D.reddy-dnsop-error-page] to this document

## 9. References

### 9.1. Normative References

- [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>>.
- [RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <<https://www.rfc-editor.org/info/rfc6838>>.
- [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <<https://www.rfc-editor.org/info/rfc6891>>.

- [RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <<https://www.rfc-editor.org/info/rfc7159>>.
- [RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <<https://www.rfc-editor.org/info/rfc7493>>.
- [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>>.
- [RFC8310] Dickinson, S., Gillmor, D., and T. Reddy, "Usage Profiles for DNS over TLS and DNS over DTLS", RFC 8310, DOI 10.17487/RFC8310, March 2018, <<https://www.rfc-editor.org/info/rfc8310>>.

## 9.2. Informative References

- [I-D.ietf-dprive-dnssoquic]  
Huitema, C., Dickinson, S., and A. Mankin, "DNS over Dedicated QUIC Connections", Work in Progress, Internet-Draft, draft-ietf-dprive-dnssoquic-12, 20 April 2022, <<https://datatracker.ietf.org/doc/html/draft-ietf-dprive-dnssoquic-12>>.
- [I-D.reddy-add-resolver-info]  
Reddy, T. and M. Boucadair, "DNS Resolver Information", Work in Progress, Internet-Draft, draft-reddy-add-resolver-info-05, 13 April 2022, <<https://datatracker.ietf.org/doc/html/draft-reddy-add-resolver-info-05>>.
- [I-D.reddy-dnsop-error-page]  
Reddy, T., Cook, N., Wing, D., and M. Boucadair, "DNS Access Denied Error Page", Work in Progress, Internet-Draft, draft-reddy-dnsop-error-page-08, 4 June 2021, <<https://datatracker.ietf.org/doc/html/draft-reddy-dnsop-error-page-08>>.
- [IANA-MediaTypes]  
IANA, "Media Types", <<https://www.iana.org/assignments/media-types>>.
- [RFC5625] Bellis, R., "DNS Proxy Implementation Guidelines", BCP 152, RFC 5625, DOI 10.17487/RFC5625, August 2009, <<https://www.rfc-editor.org/info/rfc5625>>.

- [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>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [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>>.
- [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>>.
- [RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <<https://www.rfc-editor.org/info/rfc8792>>.
- [RFC8914] Kumari, W., Hunt, E., Arends, R., Hardaker, W., and D. Lawrence, "Extended DNS Errors", RFC 8914, DOI 10.17487/RFC8914, October 2020, <<https://www.rfc-editor.org/info/rfc8914>>.

## Authors' Addresses

Dan Wing  
Citrix Systems, Inc.  
United States of America  
Email: [dwing-ietf@fuggles.com](mailto:dwing-ietf@fuggles.com)

Tirumaleswar Reddy  
Akamai  
Bangalore  
Karnataka  
India  
Email: [kondtir@gmail.com](mailto:kondtir@gmail.com)

Neil Cook  
Open-Xchange  
United Kingdom  
Email: [neil.cook@noware.co.uk](mailto:neil.cook@noware.co.uk)



Mohamed Boucadair  
Orange  
35000 Rennes  
France  
Email: mohamed.boucadair@orange.com

Domain Name System Operations (dnsop)	U. Wisser
Internet-Draft	The Swedish Internet Foundation
Intended status: Standards Track	S. Huque
Expires: 7 September 2022	Salesforce
	6 March 2022

DNSSEC automation  
draft-wisser-dnssec-automation-03

## Abstract

This document describes an algorithm and a protocol to automate DNSSEC Multi-Signer [RFC8901] "Multi-Signer DNSSEC Models" setup, operations and decommissioning. Using Model 2 of the Multi-Signer specification, where each operator has their own distinct KSK and ZSK sets (or CSK sets), [RFC8078] "Managing DS Records from the Parent via CDS/CDNSKEY" and [RFC7477] "Child-to-Parent Synchronization in DNS" to accomplish this.

## 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 7 September 2022.

## Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	3
1.1. Out-Of-Scope . . . . .	3
1.2. Notation . . . . .	3
1.3. Requirements Language . . . . .	3
2. Use Cases . . . . .	3
2.1. Maintaining a Multi-Signer group . . . . .	4
2.2. Secure Nameserver Operator Transition . . . . .	4
3. Automation Models . . . . .	4
3.1. Centralized . . . . .	4
3.2. Decentralized . . . . .	4
3.3. Capabilities . . . . .	5
4. Algorithms . . . . .	5
4.1. Prerequisites . . . . .	5
4.2. Definitions . . . . .	5
4.2.1. DS Waiting Time . . . . .	5
4.2.2. DNSKEY Waiting Time . . . . .	6
4.2.3. NS Waiting Time . . . . .	6
4.3. Setting up a new Multi-Signer group . . . . .	6
4.4. A Signer joins the Multi-Signer group . . . . .	6
4.5. A signer leaves the Multi-Signer group . . . . .	7
4.6. A Signer performs a ZSK rollover . . . . .	7
4.7. A Signer performs a CSK or KSK rollover . . . . .	8
4.8. Algorithm rollover for the whole Multi-Signer group. . .	8
5. Signers with different algorithms in one Multi-Signer group . . . . .	9
6. Acknowledgements . . . . .	10
7. IANA Considerations . . . . .	10
8. Implementation Status . . . . .	10
9. Security Considerations . . . . .	10
10. Normative References . . . . .	10
11. Informative References . . . . .	11
Appendix A. Change History . . . . .	11
A.1. Change from 01 to 02 . . . . .	11
A.2. Change from 02 to 03 . . . . .	11
Authors' Addresses . . . . .	12

## 1. Introduction

[RFC8901] describes the necessary steps and API for a Multi-Signer DNSSEC configuration. In this document we will combine [RFC8901] with [RFC8078] and [RFC7477] to define an automatable algorithm for setting up, operating and decommissioning of a Multi-Signer DNSSEC configuration.

One of the special cases of Multi-Signer DNSSEC is the secure change of DNS operator. Using Multi-Signer Model 2 the secure change of DNS operator can be accomplished.

### 1.1. Out-Of-Scope

In order for any Multi-Signer group to give consistent answers across all nameservers, the data contents of the zone also have to be synchronized (in addition to infrastructure records like NS, DNSKEY, CDS etc). This content synchronization is out-of-scope for this document (although there are a number of methods that can be used, such as making the the same updates to each operator using their respective APIs, using zone transfer in conjunction with "inline signing" at each operator, etc.)

### 1.2. Notation

Short definitions of expressions used in this document

**\*Signer\***

An entity signing a zone

**\*Multi-Signer Group\***

A group of signers that sign the same zone

**\*Controller\***

An entity controlling the multi-signer group. Used in the decentralized model.

### 1.3. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

## 2. Use Cases

### 2.1. Maintaining a Multi-Signer group

As described in [RFC8901] a Multi-Signer DNSSEC configuration has some challenges that can be overcome with the right infrastructure and following a number of steps for setup and operation.

In this document we describe, except for the initial trust, how the steps in the Multi-Signer DNSSEC setup can be automated.

### 2.2. Secure Nameserver Operator Transition

Changing the nameserver operator of a DNSSEC signed zone can be challenging. Currently the most common method is temporarily "going insecure". This is poor for security, and for users relying on the security of the zone. Furthermore, when DNSSEC is being used for application security functions like DANE [RFC6698], it is critical that the DNSSEC chain of trust remain unbroken during the transfer.

Multi-Signer DNSSEC Model 2 provides a mechanism for transitioning from one nameserver operator to another without "going insecure". A new operator joins the current operator in a temporary Multi-Signer group. Once that is accomplished and stable the old operator leaves the Multi-Signer group completing the transition.

## 3. Automation Models

Automation of the necessary steps can be categorized into two main models, centralized and decentralized. Both have pros and cons, and a zone operator should carefully choose the model that works best.

### 3.1. Centralized

In a centralized model the zone operator will run controller that executes all steps necessary and controls all signers.

A centralized controller needs to have authorized access to all signers. This can be achieved in a variety of different ways. For example will many service providers offer access through a REST API. Another possibility is access through Dynamic Update [RFC2136] with TSIG authentication.

### 3.2. Decentralized

In the decentralized models all signers will communicate with each other and execute the necessary steps on their instance only. For this signers need a specialized protocol to communicate configuration details that are not part of the zone data.

### 3.3. Capabilities

In order for any of the models to work the signer must support the following capabilities.

1. Add DNSKEY records (without the private key)
2. Remove (previously added) DNSKEY record(s)
3. Add CDS and CDNSKEY records for keys not in the DNSKEY set
4. Remove (previously added) CDS and CDNSKEY records
5. Add CSYNC record
6. Remove CSYNC record

## 4. Algorithms

### 4.1. Prerequisites

Each Signer to be added, including the initial Signer, must meet the following prerequisites before joining the Multi-Signer Group

1. A working setup of the zone, including DNSSEC signing.
2. Uses the same algorithm for DNSSEC signing as the Multi-Signer group uses or will use.
3. Signer or controller must be able to differentiate between its own keys and keys from others signers
4. Signer controller must be able to differentiate between NS records that are updated by itself and NS records that receive updates from other signers.
5. The domain must be covered by a CDS/CDNSKEY scanner and a CSYNC scanner. Otherwise updates to the parent zone have to be made manually.

### 4.2. Definitions

#### 4.2.1. DS Waiting Time

Once the parent has picked up and published the new DS record set, the any further changes MUST to be delayed until the new DS set has propagated.

The minimum DS Waiting Time is the TTL of the DS RRset.

#### 4.2.2. DNSKEY Waiting Time

Once the DNSKEY sets of all signers are updated, any further changes MUST to be delayed until the new DNSKEY set has propagated.

The minimum DNSKEY Waiting Time is the maximum of all DNSKEYS TTL values from all signers plus the time it takes to publish the zone on all secondaries.

#### 4.2.3. NS Waiting Time

Once the parent has picked up and published the new NS record set, any further changes MUST be delayed until the new NS set has propagated.

The minimum NS Waiting Time is the maximum of the TTL value of the NS set in the parent zone and all NS sets from all signers.

#### 4.3. Setting up a new Multi-Signer group

The zone is already authoritatively served by one DNS operator and is DNSSEC signed. For full automation both the KSK and ZSK or CSK must be online.

This would be a special case, a Multi-Signer group with only one signer.

#### 4.4. A Signer joins the Multi-Signer group

1. Confirm that the incoming Signer meets the prerequisites.
2. Establish a trust mechanism between the Multi-Signer group and the Signer.
3. Add ZSK for each signer to all other Signers.
4. Calculate CDS/CDNSKEY Records for all KSKs/CSKs represented in the Multi-Signer group.
5. Configure all Signers with the compiled CDS/CDNSKEY RRSET.
6. Wait for Parent to publish the combined DS RRset.
7. Remove CDS/CDNSKEY Records from all Signers. (optional)
8. Wait maximum of DS-Wait-Time and DNSKEY-Wait-Time

9. Compile NS RRSET including all NS records from all Signers.
  10. Configure all Signers with the compiled NS RRSET.
  11. Compare NS RRSET of the Signers to the Parent, if there is a difference publish CSYNC record with NS and A and AAAA bit set on all signers.
  12. Wait for Parent to publish NS.
  13. Remove CSYNC record from all signers. (optional)
- 4.5. A signer leaves the Multi-Signer group
1. Remove exiting Signer's NS records from remaining Signers
  2. Compare NS RRSET of the Signers to the Parent, if there is a difference publish CSYNC record with NS and A and AAAA bit set on remaining signers.
  3. Wait for Parent to publish NS RRSET.
  4. Remove CSYNC record from all signers. (optional)
  5. Wait NS-Wait-Time
  6. Stop the exiting Signer from answering queries.
  7. Calculate CDS/CDNSKEY Records for KSKs/CSKs published by the remaining Signers.
  8. Configure remaining Signers with the compiled CDS/CDNSKEY RRSET.
  9. Remove ZSK of the exiting Signer from remaining Signers.
  10. Wait for Parent to publish the updated DS RRset.
  11. Remove CDS/CDNSKEY set from all signers. (Optional)
- 4.6. A Signer performs a ZSK rollover
1. The signer introduces the new ZSK in its own DNSKEY RRset.
  2. Update all signers with the new ZSK.
  3. Wait DNSKEY-Wait-Time
  4. Signer can start using the new ZSK.



5. When the old ZSK is not used in any signatures by the signer, the signer can remove the old ZSK from its DNSKEY RRset.
  6. Remove ZSK from DNSKEY RRset of all signers.
- 4.7. A Signer performs a CSK or KSK rollover
1. Signer publishes new CSK / KSK in its own DNSKEY RRset.
  2. In case of CSK, add CSK to DNSKEY set of all other Signers.
  3. Signer signs DNSKEY RRset with old and new CSK / KSK.
  4. Calculate new CDS/CDNSKEY RRset and publish on all signers.
  5. Wait for parent to pickup and publish new DS RR set.
  6. Wait DS-Wait-Time + DNSKEY-Wait-Time
  7. Signer removes old CSK/KSK from its DNSKEY RR set. And removes all signatures done with this key.
  8. In case of CSK, remove old CSK from DNSKEY set of all other signers.
  9. Calculate new CDS/CDNSKEY RRset and publish on all signers.
  10. Wait for parent to pickup and publish new DS RR set.
  11. Remove CDS/CDNSKEY RR sets from all signers.
- 4.8. Algorithm rollover for the whole Multi-Signer group.
1. All signers publish KSK and ZSK or CSK using the new algorithm.
  2. All signers sign all zone data with the new keys.
  3. Wait until all signers have signed all data with the new key(s).
  4. Add new ZSK of each signer to all other Signers.
  5. Calculate new CDS/CDNSKEY RRset and publish on all signers.
  6. Wait for parent to pickup and publish new DS RR set.
  7. Wait DS-Wait-Time + DNSKEY-Wait-Time

8. Removes all keys and signatures which are using the old algorithm.
  9. Calculate new CDS/CDNSKEY RRset and publish on all signers.
  10. Wait for parent to pickup and publish new DS RR set.
  11. Remove CDS/CDNSKEY RR sets from all signers.
5. Signers with different algorithms in one Multi-Signer group

Section 2.2 of [RFC4035] states that a signed zone MUST include a DNSKEY for each algorithm present in the zone's DS RRset and expected trust anchors for the zone.

A setup where different signers use different key algorithms therefore violates [RFC4035].

According to Section 5.11 of [RFC6840] validators 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.

So a Multi-Signer setup where different signers use different key algorithms should still validate.

This could be an acceptable risk in a situation where going insecure is not desirable or impossible and name servers have to be changed between operators which only support distinct set of key algorithms.

We have to consider the following scenarios

\*Validator supports both algorithms\*

Validation should be stable through all stages of the multi-signer algorithms.

\*Validator supports none of the algorithms\*

The validator will treat the zone as unsigned. Resolution should work through all stages of the multi-signer algorithms.

\*Validator supports only one of the algorithms\*

The validator will not be able to validate the DNSKEY RR set or any data from one of the signers. So in some cases the validator will consider the zone bogus and reply with a SERVFAIL response code.

The later scenario can be mitigated, but not fully eliminated, by selecting two well supported algorithms.

## 6. Acknowledgements

The authors would like to thank the following for their review of this work and their valuable comments: Steve Crocker, Eric Osterweil, Roger Murray, Jonas Andersson, Peter Thomassen.

## 7. IANA Considerations

## 8. Implementation Status

One implementation of a centralized controller which supports updates through Dynamic DNS or REST API's of several vendors has been implemented by the Swedish Internet Foundation.

The code can be found as part of the Multi-Signer project on Github <https://github.com/DNSSEC-Provisioning/multi-signer-controller>

## 9. Security Considerations

Every step of the multi-signer algorithms has to be carefully executed at the right time and date. Any failure could resolve in the loss of resolution for the domain.

Independently of the chosen model, it is crucial that only authorized entities will be able to change the zone data. Some providers or software installation allow to make more specific configuration on the allowed changes. All extra steps to allows as little access to change zone data as possible should be taken.

If used correctly the multi-signer algorithm will strengthen the DNS security by avoiding "going insecure" at any stage of the domain life cycle.

## 10. Normative References

- [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>>.
- [RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <<https://www.rfc-editor.org/info/rfc2136>>.

- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <<https://www.rfc-editor.org/info/rfc4035>>.
- [RFC6840] Weiler, S., Ed. and D. Blacka, Ed., "Clarifications and Implementation Notes for DNS Security (DNSSEC)", RFC 6840, DOI 10.17487/RFC6840, February 2013, <<https://www.rfc-editor.org/info/rfc6840>>.
- [RFC7477] Hardaker, W., "Child-to-Parent Synchronization in DNS", RFC 7477, DOI 10.17487/RFC7477, March 2015, <<https://www.rfc-editor.org/info/rfc7477>>.
- [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>>.

## 11. Informative 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>>.
- [RFC8901] Huque, S., Aras, P., Dickinson, J., Vcelak, J., and D. Blacka, "Multi-Signer DNSSEC Models", RFC 8901, DOI 10.17487/RFC8901, September 2020, <<https://www.rfc-editor.org/info/rfc8901>>.

## Appendix A. Change History

### A.1. Change from 01 to 02

1. Trying to fix wording to be more precise
2. Added algorithm for ZSK rollover
3. Added algorithm for KSK rollover
4. Added algorithm for algorithm rollover

### A.2. Change from 02 to 03

1. Fix sequence of steps in the joining procedure
2. Explicit handling of CSK cases in CSK/ KSK rollover

Authors' Addresses

Ulrich Wisser  
The Swedish Internet Foundation  
Box 92073  
SE-12007 Stockholm  
Sweden  
Email: [ulrich@wisser.se](mailto:ulrich@wisser.se)  
URI: <https://www.internetstiftelsen.se>

Shumon Huque  
Salesforce  
415 Mission Street, 3rd Floor  
San Francisco, CA 94105  
United States of America  
Email: [shuque@gmail.com](mailto:shuque@gmail.com)