dnsext
Internet-Draft
Expires: May 12, 2011

B. Dickson
Brian Dickson

November 8, 2010

DNSSEC Delegation Signature with Canonical Signer Name
draft-dickson-dnsext-ds2-01

Abstract

The Domain Name System Security (DNSSEC) Extensions introduced the DS resource record (RR) for authentication of zone delegations. This document introduces an alternative resource record, DS2, which similarly provides authentication of zone delegations. However, DS2 provides a canonical signer name, for zones whose content may be duplicated with multiple owner names. The zone is signed by the canonical signer, and the DS2 record allows for validation using this signer name.

Author's Note

Intended Status: Proposed Standard.

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 http://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 May 12, 2011.

Copyright Notice

Table of Contents

1. Introduction 4
1.1. Rationale 4
1.2. Requirements 4
1.3. Terminology 4
2. Backward Compatibility 5
3. The DS2 Resource Record 5
3.1. RDATA Fields 6
3.1.1. The Canonical Signer Name Field 6
3.1.2. The Key Tag Field 6
3.1.3. The Algorithm Field 6
3.1.4. The Digest Type Field 6
3.1.5. The Digest Field 7
3.2. DS2 RDATA Wire Format 7
3.3. Presentation Format 7
3.4. DS2 RR Example 8
4. Authoritative Server Considerations 8
4.1. Delegated (Child) Zones Beneath DS2-signed Zone Cuts . . . 8
4.1.1. Zone Signing 9
4.1.2. Dynamic Update 9
4.2. Parent Zone at Zone Cut 10
4.2.1. Zone Serving 10
4.2.2. Referrals to Signed Zones 10
4.2.3. Responding to Queries for DS2 RRs 10
5. Validator Considerations 10
5.1. DS responses without NSEC/NSEC3 11
5.2. NSEC/NSEC3 RRs with DS2 type bit set 11
5.3. DS2 Signed Delegated Zones 11
5.3.1. RRSIG Validation Using Canonical Signer Name 11
6. Security Considerations 12
7. IANA Considerations 12
8. Acknowledgements 12
9. References 12
9.1. Normative References 12

9.2. Informative References 13
Appendix A. Constructs 13
A.1. Elements 13
A.1.1. Zone Master File 13
A.1.2. Zone Delegation (Zone Cut) 13

A.1.3. $ORIGIN 14
A.1.4. $INCLUDE 14
A.2. Basic Constructs 14
A.2.1. Zone Cut as a Replacement for a CNAME 14
A.2.2. Zone Cut as a Replacement for a DNAME 16
A.2.3. Identical Zones with Non-Identical Child Zones 18
Appendix B. Construction Examples, Unsigned 20
B.1. Objective, In Lay Terms 20
B.2. Building a Set of 'same' Zones 20
B.3. Building a Two-Deep Example 22
B.3.1. First Level Zone File 22
B.3.2. Server Conf for First Level Zones 22
B.3.3. Server Conf for Second Level Zones 23
Appendix C. Construction Examples, Signed 23
C.1. Signed Single Level Zones 23
C.1.1. Create the Single Level Zone 23
C.1.2. Sign the zone 24
C.1.3. De-canonicalize the zone 24
C.1.4. Add DS2 Records to Parent Zones 24
C.2. Signed Two Level Zones 24
C.2.1. Create the Child Zone 24
C.2.2. Sign the child zone 25
C.2.3. De-canonicalize the child zone 25
C.2.4. Create the Canonical First-Level Zone And Add DS2 Records 25
C.2.5. Sign the first-level zone 25
C.2.6. De-canonicalize the first-level zone 25
C.2.7. Add DS2 Records to Parent Zones 25
Appendix D. Use Cases 25
D.1. All the same 25
D.2. Two (ore more) levels of sameness 26
D.3. Groupings of sameness with additional singletons, beneath identical zones 26
D.4. All different below one or more levels of all the same . . 27
Appendix E. Signed Zone Examples 28
Appendix F. Example Responses 28

Author's Address 28

1. Introduction

1.1. Rationale

The DNS Security Extensions included the DS RR to provide authenticated zone delegations. Though the DS RR meets the requirements for authentication of delegations, it introduces a side-effect in that otherwise identical zones with different owner names produce non-identical cryptographic signatures. Such zones must be signed repeatedly, using each unique owner name. This property introduces undesired scaling effects.

The existence of Internationalized Domain Names (IDNs) creates the need for "equivalent" zones, whose content is identical, but whose owner name differs. Among the sources of this need, are the existence of many-to-one mappings of symbol to canonical meaning, analogous to the two-to-one mapping of upper/lower to lowercase in ASCII.

When the many-to-one mapping occurs once per label, the inflated number of zones is not unreasonable. However, there is no such limitation on per-label instances, nor on the number of labels having this characteristic, in IDNs. The desire or need for multiple names for identical zones is not limited to IDNs; in particular, identical names under multiple TLDs is another use case.

The cost to cryptographically secure multiple identical zones is high, relative to the perceived security benefit, in certain cases: identical zones known by very many names, large zones, and zones

where zone data will be updated rapidly, and any combination thereof. In these cases, the costs of maintaining the multitude of RRSIGs may be extremely high and use of the "re-usable" RRSIGs for duplicated zone content may be more appropriate.

This document presents the DS2 Resource Record which can be used as an alternative, or even adjunct, to the DS RR to mitigate these issues.

1.2. Requirements

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].

1.3. Terminology

The reader is assumed to be familiar with the basic DNS and DNSSEC concepts described in [RFC1034], [RFC1035], [RFC4033], [RFC4034],

[RFC4035], and subsequent RFCs that update them: [RFC2136], [RFC2181], and [RFC2308].

The following terminology is used throughout this document:

Delegation: an NS RRSet with a name different from the current zone apex, signifying a delegation to a child zone.

Secure delegation: a name containing a delegation (NS RRSet) and a signed DS RRSet, signifying a delegation to a signed child zone.

Insecure delegation: a name containing a delegation (NS RRSet), but lacking a DS RRSet, signifying a delegation to an unsigned child zone.

2. Backward Compatibility

This specification describes a protocol change that is partially backwards compatible with RFC 4033 [RFC4033] [RFC4033] , RFC 4034 [RFC4034] [RFC4034] , and RFC 4035 [RFC4035] [RFC4035] . In particular, security-aware resolvers that are unaware of this

specification (DS2-unaware resolvers) may not validate the responses introduced by this document. Zones delegations signed with DS2 only, will be interpreted as "Insecure" by DS2-unaware resolvers.

Security-unaware resolvers are unaffected by this specification.

3. The DS2 Resource Record

The DS2 Resource Record (RR) is very similar to the DS RR RFC 4034 [RFC4034] [RFC4034] . The main distinction is that the DS2 RR includes an explicit Canonical Signer Name, as opposed to the implicit signer name of the DS RR.

The Canonical Signer Name is used in the validation process, for both the corresponding DNSKEY RR, and for validation of RRSIG RRs in the delegated zone.

The delegated zone is the product of signing a version of the zone, where the owner and signer of the zone, and parent of all RRs in the zone, is the Canonical Signer Name.

The remainder of the DS2 RR is the same as that of the DS RR, where the DS RR was produced using the Canonical Signer Name as the owner name (and signer name) of the DNSKEY RR .

Note that the DS RR with an owner of the Canonical Signer Name MAY exist, but does not need to exist.

Note also that DS and DS2 records may coexist. This is likely to occur in a zone which was itself delegated via a DS2, i.e. for which multiple, differently-named copies of the zone exist.

The type number for the DS2 record is {TBD}.

The DS2 resource record is class independent.

The DS2 RR has no special TTL requirements.

3.1. RDATA Fields

3.1.1. The Canonical Signer Name Field

The Canonical Signer Name field identifies the name to be used as the Signer when performing validation on the delegated zone.

The Canonical Signer Name is also used in validating the DNSKEY which corresponds to the DS2 record in the zone apex of the delegated zone.

This is detailed below.

3.1.2. The Key Tag Field

The Key Tag field lists the key tag of the DNSKEY RR referred to by the DS record, in network byte order.

The Key Tag used by the DS RR is identical to the Key Tag used by RRSIG RRs. Appendix B describes how to compute a Key Tag.

3.1.3. The Algorithm Field

The Algorithm field lists the algorithm number of the DNSKEY RR referred to by the DS record.

The algorithm number used by the DS RR is identical to the algorithm number used by RRSIG and DNSKEY RRs. Appendix A.1 lists the algorithm number types.

3.1.4. The Digest Type Field

The DS RR refers to a DNSKEY RR by including a digest of that DNSKEY RR. The Digest Type field identifies the algorithm used to construct the digest. Appendix A.2 lists the possible digest algorithm types.

3.1.5. The Digest Field

The DS record refers to a DNSKEY RR by including a digest of that DNSKEY RR.

The digest is calculated by concatenating the canonical form of the fully qualified owner name of the DNSKEY RR with the DNSKEY RDATA, and then applying the digest algorithm.

digest = digest_algorithm(DNSKEY owner name | DNSKEY RDATA);

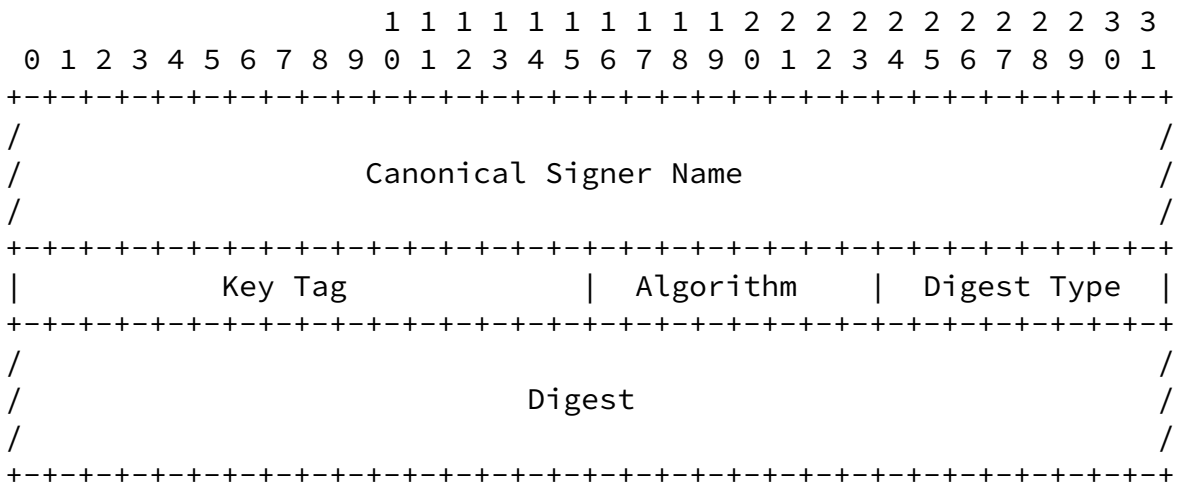
"|" denotes concatenation

DNSKEY RDATA = Flags | Protocol | Algorithm | Public Key.

The size of the digest may vary depending on the digest algorithm and DNSKEY RR size. As of the time of this writing, the only defined digest algorithm is SHA-1, which produces a 20 octet digest.

3.2. DS2 RDATA Wire Format

The RDATA for a DS RR consists of a Canonical Signer Name (domain name), a 2 octet Key Tag field, a 1 octet Algorithm field, a 1 octet Digest Type field, and a Digest field.



3.3. Presentation Format

The presentation format of the RDATA portion is as follows:

The Canonical Signer Name field is represented as a domain name.

The Key Tag field MUST be represented as an unsigned decimal integer.

The Algorithm field MUST be represented either as an unsigned decimal

integer or as an algorithm mnemonic specified in Appendix A.1.

The Digest Type field MUST be represented as an unsigned decimal integer.

The Digest MUST be represented as a sequence of case-insensitive hexadecimal digits. Whitespace is allowed within the hexadecimal text.

3.4. DS2 RR Example

The following example shows a DNSKEY RR and its corresponding DS2 RR.

```
dskey.example.com. 86400 IN DNSKEY 256 3 5 ( AQOeiiR0GOMYkDshWoSKz
                                             9XzfwJr1AYtsmx3TGkJaNXVb
                                             fi/2pHm822aJ5iI9BMzNXxeY
                                             CmZDRD99WYwYqUSdjMmmAphX
                                             dvxegXd/M5+X7OrzKBaMbCVd
                                             FLUUh6DhweJBjEVv5f2wwjM9
                                             XzcnOf+EPbtG9DMBmADjFDc2
                                             w/rljwvFw==
                                             ) ;  key id = 60485

dskey.example.com. 86400 IN DS2 dskey.example.com. 60485 5 1 ( 2BB183
                                             AF5F22588179A53B0A98631FAD
                                             1A292118 )
```

The first four text fields specify the name, TTL, Class, and RR type (DS2). The name "dskey.example.com." is the Canonical Signer Name, which in this case is the same as the owner name. Value 60485 is the key tag for the corresponding "dskey.example.com." DNSKEY RR, and value 5 denotes the algorithm used by this "dskey.example.com." DNSKEY RR. The value 1 is the algorithm used to construct the digest, and the rest of the RDATA text is the digest in hexadecimal.

4. Authoritative Server Considerations

4.1. Delegated (Child) Zones Beneath DS2-signed Zone Cuts

The DS2 signature is designed to permit zones signed by another Signer Name, to be authenticated.

While the motivation for this is enabling duplicated zone content under different owner names, this is not a requirement.

Child zones are always unique from the perspective of resolvers. All delegations are normal delegations. It is only the delegation signatures (DS or DS2) that get special treatment.

The Canonical Signer Name (CSN) is ONLY to be used for signature validation. No relationship between zones sharing CSNs is implied or should be inferred.

In particular, the "strong linking" of child zones (on the authority server(s)) is not required. Even if two child zones started off identical, they would be permitted to subsequently diverge, regardless of SOA Serial Numbers.

4.1.1. Zone Signing

The Child zone below a DS2-signed zone cut, are signed as follows:
The rightmost portion of the owner name matching the zone name, is replaced with the Canonical Signer Name.
The Signer Name used is the Canonical Signer Name.
The RRsets are signed with the above changes, but otherwise as usual.

The normal usage of DS2, is that a "canonical" version of a zone, with owner name of the Canonical Signer Name, is signed normally per RFC 4035. Then, the entire zone, including signatures, is used verbatim, except for changing the rightmost portion of the owner names.

This "copy" of the zone exists for the purpose of having the same zone known by multiple names, e.g. aliasing, such as for IDN usage. The DS2 ensures that the delegation is secure, and that the existing signatures validate properly.

Copying the zone is not the only means by which such a zone may be created or used.

4.1.2. Dynamic Update

In addition to the existing issues for Dynamic Updates for DNSSEC-signed zones, a new problem is introduced by this specification.

When a dynamic update needs to be applied to a DS2-signed delegated zone, might incorrectly be presumed that it needs to be applied to all "copies" of the zone.

The correct behaviour is specified by the Dynamic Update

specification, RFC 2181. Only the Primary Master Server is able to know, and respond appropriately, if zones are in fact "copies".

The specific mechanism by which this occurs is beyond the scope of this document.

However, implementors intending to enable "copies", are encouraged to consider methods which do not duplicate data, but rather maintain a single instance of the zone data. This methodology implies that dynamic updates to the zone, by any of its names, result in the zone being updated for all names by which the zone is known.

4.2. Parent Zone at Zone Cut

4.2.1. Zone Serving

This specification modifies DNSSEC-enabled DNS responses generated by authoritative servers. In particular, when a signed delegation is returned, whether it includes a DS, DS2, or both, it is also necessary to return the corresponding NSEC/NSEC3 record showing the covered RR types.

4.2.2. Referrals to Signed Zones

At a zone cut, it is possible for both DS and DS2 RRs to exist for the same owner name. The new rule applies in all circumstances, whether only DS, only DS2, or both, exist to sign the delegation.

The rule is: always return the corresponding NSEC/NSEC3 record matching the owner (or owner hash).

The reason for this is, that receiving validator will need to know which RR types are expected, in the answer.

A DS2-unaware receiving validator will need the NSEC/NSEC3 in particular, if a DS2 RR exists but no DS RR exists. In that case, the zone will validate as "Insecure", since no DS exists and has been proven by the NSEC/NSEC RR (and its RRSIG).

4.2.3. Responding to Queries for DS2 RRs

This is the same circumstance as section 3.1.4.1 of RFC 4034,

"Responding to Queries for DS RRs". The same rules as DS RRs, apply to DS2 RRs.

5. Validator Considerations

In RFC 5155, the Validator Considerations referencing DS, must be presumed to now read "DS or DS2" (or in the negative sense, "neither DS nor DS2").

Validators MUST use the original Owner Name for any necessary queries for DS, DS2, DNSKEY, and RRSIG resource records. This is because there is no explicit or implicit relationship between the Canonical Signer Name, and a zone whose name corresponds to the Canonical Signer Name (which might not even exist).

5.1. DS responses without NSEC/NSEC3

If a response is received which includes a DS RR, but does not include the corresponding NSEC/NSEC3 RR, the validator MUST attempt to retrieve the NSEC/NSEC RR, and subsequently attempt to retrieve the corresponding DS2 record if the DS2 type bit is set in the NSEC/NSEC3 type bit map.

Intermediate caches or middleboxes may not be DS2-aware, and may not include DS2 records when QTYPE is not DS2. If the middlebox is DS2-unaware, the middleboxes SHOULD still allow responses for exact matches to unknown types, such as DS2.

5.2. NSEC/NSEC3 RRs with DS2 type bit set

If a DS2-aware validator receives a response with an NSEC/NSEC3 RR with the DS2 type bit set, but had not also received the DS2 RR, the validator MUST attempt to retrieve the DS2 RRset.

An intermediate cache or middlebox may have filtered out, or not cached, the DS2 RR. Explicit querying for the DS2 RR may return the missing DS2 RR.

5.3. DS2 Signed Delegated Zones

Zones whose delegation is signed by a DS2 RR, and which DS2 RR

successfully validates, MUST be validated using the Canonical Signer Name (or Names, if multiple DS/DS2 RRs are present).

More specifically, RRSIG RRs in the delegated zone, MUST be validated with Canonical Signer Names (CSN) whose DS2 RR is itself validated. For purposes of this process, every DS RR can be treated as a DS2 RR with the Canonical Signer Name of the DS2 RR's owner name, and MUST be included in the set of CSNs.

5.3.1. RRSIG Validation Using Canonical Signer Name

The procedure for validating an RRSIG is modified as follows:
The RRSIG Signer Name MUST match the Canonical Signer Name. No comparison is made between the Owner Name and either the Canonical Signer Name or the Signer Name of the RRSIG.

The number of labels in the RRset Owner name, minus the numer of labels in the Zone Name, plus the number of labels in the Canonical Signer Name, MUST be greater than or equal to the value in the RRSIG RR's Labels field.
The RRSIG RR's Algorithm, and Key Tag fields MUST match the algorithm, and key tag for some DNSKEY RR in the zone's apex DNSKEY RRset.

6. Security Considerations

Further work on DS/DS2 security issues may be necessary. In particular, it is unclear whether the implicit signalling of use of NSEC/NSEC3 may be vulnerable to downgrade attacks.

7. IANA Considerations

This document updates the IANA registry "DOMAIN NAME SYSTEM PARAMETERS" (http://www.iana.org/assignments/dns-parameters) in sub-registry "TYPES", by defining one new type. Section 3 defines the DS2 RR type {TBD}.

8. Acknowledgements

The problems related to aliasing, as discussed on the DNSEXT WG mailing list, provided helpful direction in limiting what changes could or should be made to DNS, and DNSSEC in particular.

9. References

9.1. Normative References

[RFC1033] Lottor, M., "Domain administrators operations guide", RFC 1033, November 1987.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC2136] Vixie, P., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, July 1997.

[RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, March 1998.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

[RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, March 2008.

9.2. Informative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Appendix A. Constructs

A.1. Elements

The following DNS elements are used in the constructs discussed here.

A.1.1. Zone Master File

The term "zone master file" has the meaning established in RFC 1035. It is the file containing all the resource records and directives for the zone.

A.1.2. Zone Delegation (Zone Cut)

The terms "zone delegation" and "zone cut" may generally be used interchangeably.

A zone delegation is the act of putting in NS records in a zone at a place other than the apex.

The zone delegation is the parent side of a zone cut.

A zone cut creates a new zone. The apex of the new zone has the same owner name as the zone delegation.

A.1.3. $ORIGIN

Most DNS authority servers include the ability to directly refer to a zone master file, with an implied $ORIGIN. For those, no $ORIGIN needs to be used.

For completeness, zone master files with explicit $ORIGIN are also used in examples. This is in keeping with RFC 1035.

A.1.4. $INCLUDE

In the instances where $ORIGIN use is needed, it is also necessary to include common content via the $INCLUDE directive. This is also RFC 1035 compliant.

A.2. Basic Constructs

The following examples show ways to replace CNAME/DNAME usage with other existing DNS items. Rationale for usage, and differences from CNAME/DNAME are explained for each.

A.2.1. Zone Cut as a Replacement for a CNAME

The following example shows a simple CNAME, where the CNAME target (for convenience) is a sibling of the CNAME itself.

```
zone file for example.com:

$ORIGIN example.com.
@        SOA ns1.example.com. postmaster (
                    42 7200 600 3600000 60 )
     NS ns1.example.com
alias  CNAME real.example.com
real   TXT "This is real data, maintained in a single location."
```

Querying for type TXT at alias.example.com, will return the CNAME RR plus the TXT RR for real.example.com (the CNAME target).

The following example shows the use of a zone cut, and the inclusion of data from a single location, as an alternative.


```
zone file for example.com:

$ORIGIN example.com.
@        SOA ns1.example.com. postmaster (
```

```
                        42 7200 600 3600000 60 )
          NS ns1.example.com
     alias NS ns1.example.com ; all zones are served by ns1
     real  NS ns1.example.com ; normally there are multiple NS's

     server configuration (with implicit $ORIGIN of zone.name):
     zone alias.example.com { master; file="czone-data.example.com";};
     zone real.example.com { master; file="czone-data.example.com";};




     Or, server configuration (without implicit $ORIGIN):
     zone alias.example.com { master; file="alias.example.com";};
     zone real.example.com { master; file="real.example.com";};

     zone file for alias.example.com:

     $ORIGIN alias.example.com.
     $INCLUDE "czone-data.example.com"

     zone file for real.example.com:

     $ORIGIN real.example.com.
     $INCLUDE "czone-data.example.com"

     file czone-data.example.com:

@         SOA ns1.example.com. postmaster ( 42 7200 600 3600000 60 )
          NS ns1.example.com
          TXT "This is real data, maintained in a single location."
```

Querying for the TXT record for either alias.example.com, or real.example.com, requires first following the delegation, and then gets a response which does not include any CNAME.

The main difference here is, that the DNS node "alias.example.com" is a real, first-class entry, which can be used interchangeably with "real.example.com". E.g., it can be used as an MX target.

The other important difference is, as the data is all authoritative and self-contained, it is not possible to have a broken CNAME. If the necessary linkage is incorrect, the zone file will not load.

This means that maintaining aliased data in this way, returns values in a consistent manner, in a deterministic path and timeframe. Data kept this way can be validated, via zone loading and zone checking functions included with most modern DNS authority servers.

A.2.2. Zone Cut as a Replacement for a DNAME

The following example shows a simple DNAME, where the DNAME target (for convenience) is a sibling of the DNAME itself.

```
zone file for example.com:

$ORIGIN example.com.
@        SOA ns1.example.com. postmaster (
                    42 7200 600 3600000 60 )
     NS ns1.example.com
alias  DNAME real.example.com
content.real   TXT "This is real data, kept only here."
```

Querying for type TXT at alias.example.com, will return the CNAME RR plus the TXT RR for real.example.com (the CNAME target).

The following example shows the use of a zone cut, and the inclusion of data from a single location, as an alternative.

```
zone file for example.com:

$ORIGIN example.com.
@        SOA ns1.example.com. postmaster (
                    42 7200 600 3600000 60 )
     NS ns1.example.com
alias NS ns1.example.com ; all zones are served by ns1
real  NS ns1.example.com ; normally there are multiple NS's

server configuration (with implicit $ORIGIN of zone.name):
zone alias.example.com { master; file="dzone.example.com";};
zone real.example.com { master; file="dzone.example.com";};

Or, server configuration (without implicit $ORIGIN):
zone alias.example.com { master; file="alias.example.com";};
zone real.example.com { master; file="real.example.com";};

zone file for alias.example.com:

$ORIGIN alias.example.com.
$INCLUDE "dzone.example.com"

zone file for real.example.com:

$ORIGIN real.example.com.
$INCLUDE "dzone.example.com"

file dzone.example.com:

@        SOA ns1.example.com. postmaster ( 42 7200 600 3600000 60 )
         NS ns1.example.com
content  TXT "This is real data, maintained in a single location."
```

Querying for the TXT record for either content.alias.example.com, or content.real.example.com, requires first following the delegation, and then gets a response which does not include any DNAME or synthesized CNAME.

The main difference here is, that the DNS node

"content.alias.example.com" is a real, first-class entry, which can be used interchangeably with "content.real.example.com". E.g., it can be used as an MX target.

The other important difference is, as the data is all authoritative and self-contained, it is not possible to have a broken DNAME. If the necessary linkage is incorrect, the zone file will not load.

This means that maintaining aliased data in this way, returns values in a consistent manner, in a deterministic path and timeframe. Data kept this way can be validated, via zone loading and zone checking functions included with most modern DNS authority servers.

A.2.3. Identical Zones with Non-Identical Child Zones

This example demonstrates identical zones, where there are child zones (of all identical parents) whose content differ. The owner name of the child zones, relative to the parent apex name, is identical.

Note Well: The following example is not possible with CNAME or DNAME usage.

The query re-writing that occurs in CNAME/DNAME obscures zone cuts as a distinction for any portion of the DNS tree below the CNAME/DNAME.

```
zone file for example.com:

$ORIGIN example.com.
@        SOA ns1.example.com. postmaster (
                    42 7200 600 3600000 60 )
     NS ns1.example.com
alias NS ns1.example.com ; all zones are served by NS1
real  NS ns1.example.com ; normally there are multiple NS's

server configuration (with implicit $ORIGIN of zone.name):
zone alias.example.com {
  master; file="has-a-child.example.com";};
zone real.example.com {
  master; file="has-a-child.example.com";};
zone child.alias.example.com {
  master; file="child.alias.example.com";};
zone child.real.example.com {
  master; file="child.real.example.com";};

(Non-implicit $ORIGIN may be used, similar to previous examples.)

file has-a-child.example.com:

@        SOA ns1.example.com. postmaster (
                    42 7200 600 3600000 60 )
         NS ns1.example.com
```

```
content  TXT "identical"
; this zone cut exists in all aliased zones
child    NS ns1.example.com

    file child.alias.example.com:

@        SOA ns1.example.com. postmaster (
                      42 7200 600 3600000 60 )
         NS ns1.example.com
         TXT "divergent"

    file child.real.example.com:

@        SOA ns1.example.com. postmaster (
                      42 7200 600 3600000 60 )
         NS ns1.example.com
         TXT "unique"
```

TXT Queries for content.alias.example.com and content.real.example.com, both return "identical" results.

TXT Queries for child.alias.example.com, and child.real.example.com, return "divergent" and "unique" results respectively, which come from their respective zones.

If real.example.com were a very large zone, with very few members whose values needed to differ from alias.example.com, this mechanism would be quite useful for maintaining the data.

Appendix B. Construction Examples, Unsigned

The following examples show how to construct a hierarchy of zones, with all zones at a given depth being identical.

The zones are unsigned, and can be implemented using existing DNS elements. No DS2 is required.

B.1. Objective, In Lay Terms

In these example, we are using the constructs available, to build DNS zones with specific characteristics.

What we want, is for all the zones (under consideration) to be "the same".

What we mean by "the same" is that:
Every node in any zone, is in every zone;
Every combination of class and type, at each node, exists at that node in all the zones;
The value of each tuple (name,class,type) in each zone is identical.

In addition, whenever we use zone names (or placeholders for names), like Ai.Bj, we mean all combinations of the members of set {A} and the set {B}, combined, as {A}.{B}. The members of set {A} are A1, A2, A3, etc., and the members of set {B} are B1, B2, B3., etc. The letters "i" and "j" simply indicate that there is no relationship between the number of members in sets {A} and {B}.

Additionally, A1, A2, etc., are placeholder names for what might be actual names. For instance, A1 could be "foo.example.com.", and A2 could be "bar.example.net."

B.2. Building a Set of 'same' Zones

First, we need the zone content that will be used to populate the zones. Let's say we have a single existing zone, that we want to turn into a set of 'same' zones.

Zone file for "myzone.example.com.", the file myzone.zone:

```
myzone.example.com.   SOA ns1.example.com postmaster (
                42 7200 600 3600000 60 )
                NS ns1.example.com
mytext.myzone.example.com.        TXT "Example Text"
mymailer.myzone.example.com.      A 127.0.0.1
www.myzone.example.com.           A 127.0.0.2
radius.myzone.example.com.        A 127.0.0.3
```

First we de-canonicalize the labels within the zone:

```
Zone file for "myzone.example.com.", the file myzone.zone:

myzone.example.com.   SOA ns1.example.com postmaster (
                42 7200 600 3600000 60 )
                NS ns1.example.com
mytext          TXT "Example Text"
mymailer        A 127.0.0.1
www             A 127.0.0.2
radius          A 127.0.0.3
```

Then we turn the zone name itself into a reference to the $ORIGIN, which itself will be supplied when the zone is loaded:

```
Zone file  for "myzone.example.com.", the file myzone.zone:

@   SOA ns1.example.com postmaster (
                42 7200 600 3600000 60 )
                NS ns1.example.com
mytext          TXT "Example Text"
mymailer        A 127.0.0.1
www             A 127.0.0.2
radius          A 127.0.0.3
```

Finally, we modify the configuation file to use this file for both myzone.example.com., and any other zones we want to have be the same:

```
Configuration file (named.conf, for BIND) elements needed
for myzone.example.com and additional zones:

zone "myzone.example.com."{master; file="myzone.zone";};
zone "myzone.example.net."{master; file="myzone.zone";};
zone "other.example.org."{master; file="myzone.zone";};
```


Each time a zone file is loaded, it does so with an implicit $ORIGIN of the zone name. This causes all the zones that have @ as the zone name (SOA owner) to be identical, if they are read from the same file.

B.3. Building a Two-Deep Example

For this example, we will build several zones with the same content, by re-using the same file. For this example, we will save some steps by using the same zone file as before, 'mysone.zone'.

Since the zones will be two deep, we need another set of zones above those, i.e. one deep, who exist only to contain the two-deep zones.

In DNS parlance, these would, if inside a single zone, be considered "empty non-terminals". However, they are not inside a single zone, but rather are delegation-only zones. The "real" data is in their children, and the parent zones contain no other data besides that needed for the delegations (i.e. NS records and possibly glue records).

B.3.1. First Level Zone File

This is a "thin" zone file, containing only the labels by which the second level zones are to be known.

Zone file for first level zones, "first-level.zone":

```
@   SOA ns1.example.com postmaster (
        42 7200 600 3600000 60 )
     NS ns1.example.com
red    NS ns1.example.com
green  NS ns1.example.com
blue   NS ns1.example.com
```

B.3.2. Server Conf for First Level Zones

This establishes the parent names for the first level zones, from which the delegations to the second level occur:

Zone conf file elements for the first level zones:

```
zone "myzone.example.com."{master; file="first-level.zone";};
zone "myzone.example.net."{master; file="first-level.zone";};
zone "other.example.org."{master; file="first-level.zone";};
```

B.3.3. Server Conf for Second Level Zones

This establishes the fully qualified zone names for the second level zones, in all permutations and combinations:

Zone conf file elements for the first level zones:

```
zone "red.myzone.example.com."{master; file="myzone.zone";};
zone "red.myzone.example.net."{master; file="myzone.zone";};
zone "red.other.example.org."{master; file="myzone.zone";};
zone "green.myzone.example.com."{master; file="myzone.zone";};
zone "green.myzone.example.net."{master; file="myzone.zone";};
zone "green.other.example.org."{master; file="myzone.zone";};
zone "blue.myzone.example.com."{master; file="myzone.zone";};
zone "blue.myzone.example.net."{master; file="myzone.zone";};
zone "blue.other.example.org."{master; file="myzone.zone";};
```

While this requires setting up all the zones, there is only one zone file to maintain, and all the zones inherit the common contents.

Appendix C. Construction Examples, Signed

The following examples are exactly the same as the examples above, except that they are signed.

The zone files are identical, and require the use of DS2 signatures.

C.1. Signed Single Level Zones

The procedure here is:

- Create the first-instance zone, fully qualified.
- Sign the zone, and capture the DS record.
- De-canonicalize the zone file, and use the DS as a basis for the DS2 records.

The DS2 records will need to be signed by their respective parent zones. The signatures are omitted for clarity.

C.1.1. Create the Single Level Zone

We need to use the Canonical Signer Name for the zone name.

```
Zone file for "myzone.example.com.", the file myzone.zone:

myzone.example.com.   SOA ns1.example.com postmaster (
                42 7200 600 3600000 60 )
                NS ns1.example.com
mytext.myzone.example.com.        TXT "Example Text"
mymailer.myzone.example.com.      A 127.0.0.1
www.myzone.example.com.           A 127.0.0.2
radius.myzone.example.com.        A 127.0.0.3
```

C.1.2. Sign the zone

C.1.3. De-canonicalize the zone

C.1.4. Add DS2 Records to Parent Zones

C.2. Signed Two Level Zones

The procedure here is:
Create the first-instance child zone, fully qualified.
Sign the child zone, and capture the DS record.
De-canonicalize the child zone file, and use the DS as a basis for the DS2 records on the child delegations.
Create the first-instance parent zone, fully qualified, and include the DS2 records.
Sign the parent zone, and capture the DS record.
De-canonicalize the parent zone file, and use the DS as a basis for the DS2 records in the parent zones.
The parents' DS2 records will need to be signed by their respective parent zones. Those signatures are omitted for clarity.

C.2.1. Create the Child Zone

We need to use the Canonical Signer Name for the zone name.

```
Zone file for "red.myzone.example.com.", the file myzone.zone:

red.myzone.example.com.   SOA ns1.example.com postmaster (
                42 7200 600 3600000 60 )
                NS ns1.example.com
mytext.red.myzone.example.com.        TXT "Example Text"
mymailer.red.myzone.example.com.      A 127.0.0.1
```

```
www.red.myzone.example.com.           A 127.0.0.2
radius.red.myzone.example.com.        A 127.0.0.3
```


C.2.2. Sign the child zone

C.2.3. De-canonicalize the child zone

C.2.4. Create the Canonical First-Level Zone And Add DS2 Records

Zone file for "myzone.example.com.", "first-level.zone":

```
myzone.example.com.   SOA ns1.example.com postmaster (
        42 7200 600 3600000 60 )
     NS ns1.example.com
red.myzone.example.com.    NS ns1.example.com
                   DS2 red.myzone.example.com. (
; rest of DS record contents go here
                   )
green.myzone.example.com.  NS ns1.example.com
                   DS2 red.myzone.example.com. (
; rest of DS record contents go here
                   )
blue.myzone.example.com.   NS ns1.example.com
                   DS2 red.myzone.example.com. (
; rest of DS record contents go here
                   )
```

C.2.5. Sign the first-level zone

C.2.6. De-canonicalize the first-level zone

C.2.7. Add DS2 Records to Parent Zones

Appendix D. Use Cases

D.1. All the same

Use Case: identical zone content, where zones have more than one name.

Example:

```
colors.example.com
colours.example.com
```


D.2. Two (ore more) levels of sameness

Use Case: identical zone content, where the zones owner names contain two or more labels with equivalent labels (synonyms/homonyms/IDNs).

Example:

```
colours.iso-with-accents.example.com
colors.iso-with-accents.example.com
colours.iso-without-accents.example.com
colors.iso-without-accents.example.com
colours.ascii.example.com
colors.ascii.example.com
```

D.3. Groupings of sameness with additional singletons, beneath identical zones

Use Case: Clusters of content where the content of each cluster is identical, beneath zones which are identical.

Example: Language-specific content, grouped by country, underneath aliases for some company's zones.

[NB - Better examples of sameness and singletons are needed - same name, different content]

Zone Name	CSN
(Parents)	
big-company.example.com	big-company.example.com
trade-mark.example.com	big-company.example.com
product-name.example.com	big-company.example.com
(Children)	
en.big-company.example.com	en.big-company.example.com
en.trade-mark.example.com...	..en.big-company.example.com
en.product-name.example.com.	.en.big-company.example.com
canada-en.big-company.example.com	en.big-company.example.com
usa.big-company.example.com	en.big-company.example.com
uk.big-company.example.com	en.big-company.example.com
fr.big-company.example.com	fr.big-company.example.com
canada-fr.big-company.example.com	fr.big-company.example.com
france.big-company.example.com	fr.big-company.example.com
sw.big-company.example.com	sw.big-company.example.com
(etc)	
(Grandchildren)	

```
secure-server.en.big-company.example.com
secure-server.en.trade-mark.example.com
(etc...)
[Delegations are made to unique zones when unique values for
 arbitrary-depth labels are required.]
[E.g. for SSL certificates, unique IP addresses, and such.]
```

In the above table, each of the delegations to the children is signed by the CSN.

Each zone uses a master zone file corresponding to the CSN on a 1:1 basis.

If two zones have the same CSN, they are loaded from the same master zone file.

D.4. All different below one or more levels of all the same

Use Case: Two large zones, whose content is identical except for one node in the zone.

Use Case detail: The "exception" node is isolated via a zone cut, and all the grandchildren zones are unique.

The children zones are identical, and signed once with delegations signed by DS2 RRs. The grandchildren zones are unique, signed by their respective owner names. The delegations from the (identical)

parents of the grandchildren, require multiple DS RRs, one for each grandchild's signature.

Note Well: The grandchildren in this case, MUST be served by the same set of nameservers, since the NS records at the zone cut are identical.

```
bigzone.iso-with-accents.example.com
bigzone.ascii.example.com
unique-node.bigzone.iso-with-accents.example.com
unique-node.bigzone.ascii.example.com
```

Appendix E. Signed Zone Examples

Appendix F. Example Responses

Author's Address

Brian Dickson
Brian Dickson
6261 Lawrence St,
Halifax, NS B3L 1J8
Canada

Email: brian.peter.dickson@gmail.com