DNSEXT Working Group
INTERNET-DRAFT
<draft-ietf-dnsext-delegation-signer-09.txt>

Olafur Gudmundsson
October 2002

Updates: RFC 1035, RFC 2535, RFC 3008, RFC 3090.

Delegation Signer Resource Record

Status of this Memo

Abstract

The delegation signer (DS) resource record is inserted at a zone cut (i.e., a delegation point) to indicate that the delegated zone is digitally signed and that the delegated zone recognizes the indicated key as a valid zone key for the delegated zone. The DS RR is a modification to the DNS Security Extensions definition, motivated by

operational considerations. The intent is to use this resource record as an explicit statement about the delegation, rather than relying on inference.

This document defines the DS RR, gives examples of how it is used and the implications of this record on resolvers. This change is not backwards compatible with RFC 2535.
This document updates RFC1035, RFC2535, RFC3008 and RFC3090.

1 Introduction

Familiarity with the DNS system [RFC1035], DNS security extensions [RFC2535] and DNSSEC terminology [RFC3090] is important.

Experience shows that when the same data can reside in two administratively different DNS zones, the data frequently gets out of sync. The presence of an NS RRset in a zone anywhere other than at the apex indicates a zone cut or delegation. The RDATA of the NS RRset specifies the authoritative servers for the delegated or "child" zone. Based on actual measurements, 10-30% of all delegations on the Internet have differing NS RRsets at parent and child. There are a number of reasons for this, including a lack of communication between parent and child and bogus name servers being listed to meet registry requirements.

DNSSEC [RFC2535,RFC3008,RFC3090] specifies that a child zone needs to have its KEY RRset signed by its parent to create a verifiable chain of KEYs. There has been some debate on where the signed KEY RRset should reside, whether at the child [RFC2535] or at the parent. If the KEY RRset resides at the child, maintaining the signed KEY RRset in the child requires frequent two-way communication between the two parties. First the child transmits the KEY RRset to the parent and then the parent sends the signature(s) to the child. Storing the KEY RRset at the parent simplifies the communication.

DNSSEC [RFC2535] requires that the parent store a NULL KEY record for an unsecure child zone to indicate that the child is unsecure. A NULL KEY record is a waste: an entire signed RRset is used to communicate effectively one bit of information--that the child is unsecure. Chasing down NULL KEY RRsets complicates the resolution process in many cases, because servers for both parent and child need to be queried for the KEY RRset if the child server does not return it. Storing the KEY RRset only in the parent zone simplifies this and would allow the elimination of the NULL KEY RRsets entirely. For large delegation zones the cost of NULL keys is a significant barrier to deployment.

Another complication of the DNSSEC key model is that the KEY record can be used to store public keys for other protocols in addition to DNSSEC keys. There are number of potential problems with this, including:

1. The KEY RRset can become quite large if many applications and protocols store their keys at the zone apex. Possible protocols are IPSEC, HTTP, SMTP, SSH and others that use public key cryptography.
2. The KEY RRset may require frequent updates.
3. The probability of compromised or lost keys, which trigger emergency key rollover procedures, increases.
4. The parent may refuse sign KEY RRsets with non-DNSSEC zone keys.
5. The parent may not meet the child's expectations in turnaround time for resigning the KEY RRset.

Given these and other reasons, there is good reason to explore alternatives to using only KEY records to create a chain of trust.

Some of these problems can be reduced or eliminated by operational rules or protocol changes. To reduce the number of keys at the zone apex, a rule to require applications to store their KEY records at the SRV name for that application is one possibility. Another is to restrict the KEY record to only DNSSEC keys and create a new record type for all non-DNSSEC keys. A third possible solution is to prohibit the storage of non-DNSSEC keys at the zone apex. There are other possible solutions, but they are outside the scope of this document.

1.2 Reserved Words

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

2 Specification of the Delegation key Signer

This section defines the Delegation Signer (DS) RR type and the changes to DNS to accommodate it.

2.1 Delegation Signer Record Model

This document presents a replacement for the DNSSEC KEY record chain of trust [RFC2535] that uses a new RR that resides only at the parent. This record identifies the key(s) that the child uses to self-sign its own KEY RRset.

The chain of trust is now established by verifying the parent KEY RRset, the DS RRset from the parent and the KEY RRset at the child.

This is cryptographically equivalent to using just KEY records.

Communication between the parent and child is greatly reduced, since the child only needs to notify the parent about changes in keys that sign its apex KEY RRset. The parent is ignorant of all other keys in the child's apex KEY RRset. Furthermore, the child maintains full control over the apex KEY RRset and its content. The child can maintain any policies regarding its KEY usage for DNSSEC and other applications and protocols with minimal impact on the parent. Thus if the child wants to have frequent key rollover for its DNS zone keys, the parent does not need to be aware of it: the child can use one key to sign only its apex KEY RRset and other keys to sign the other RRsets in the zone.

This model fits well with a slow roll out of DNSSEC and the islands of security model. In this model, someone who trusts "good.example." can preconfigure a key from "good.example." as a trusted key, and from then on trusts any data signed by that key or that has a chain of trust to that key. If "example." starts advertising DS records, "good.example." does not have to change operations by suspending self-signing. DS records can also be used to identify trusted keys instead of KEY records. Another significant advantage is that the amount of information stored in large delegation zones is reduced: rather than the NULL KEY record at every unsecure delegation required by RFC 2535, only secure delegations require additional information in the form of a signed DS RRset.

The main disadvantage of this approach is that verifying a zone's KEY RRset requires two signature verification operations instead of the one required by RFC 2535. There is no impact on the number of signatures verified for other types of RRsets.

2.2 Protocol Change

All DNS servers and resolvers that support DS MUST support the OK bit [RFC3225] and a larger message size [RFC3226]. In order for a delegation to be considered secure the delegation MUST contain a DS RRset. If a query contains the OK bit, a server returning a referral for the delegation MUST include the following RRsets in the authority section in this order:

```
If DS RRset is present:
     parent NS RRset
     DS and SIG(DS)
If no DS RRset is present:
     parent NS RRset
     parent NXT and SIG(NXT)
```

This increases the size of referral messages and may cause some or all glue to be omitted. If the DS or NXT RRsets with signatures do

not fit in the DNS message, the TC bit MUST be set. Additional section processing is not changed.

A DS RRset accompanying an NS RRset indicates that the child zone is secure. If an NS RRset exists without a DS RRset, the child zone is unsecure (from the parents point of view). DS RRsets MUST NOT appear at non-delegation points or at a zone's apex.

Section 2.2.1 defines the behavior of the corner case of non recursive server that is only authorative for the child. The following section 2.2.2 replaces RFC2535 sections 2.3.4 and 3.4, section 2.2.3 replaces RFC3008 section 2.7, and RFC3090 updates are in section 2.2.4.

2.2.1 Authorative servers special processing

A server that is authorative for both parent and child, a DS aware server will return DS from the parent zone. A non DS aware server is expected to answer:

```
RCODE:             NOERROR
AA bit:            set
Answer Section:    Empty
Authority Section: SOA [+ SIG(SOA) + NXT + SIG(NXT)]
```

This indicates there is no DS at the apex. If the server is DS aware and does not perform recursive lookup for the DS, it MUST return the above answer. The reason is to avoid confusing resolvers that are non DS aware. In the early deployment of DS, most resolvers will be non-DS aware thus send DS queries to child servers rather than parent ones.

A DS aware recursive server that is authorative for the child, MAY perform a recursive query to search for the DS record, if the RD bit is set. If this server has the DS in cache it MUST return it without the AA bit.

2.2.2 RFC2535 2.3.4 and 3.4: Special Considerations at Delegation Points

DNS security views each zone as a unit of data completely under the control of the zone owner with each entry (RRset) signed by a special private key held by the zone manager. But the DNS protocol views the leaf nodes in a zone that are also the apex nodes of a child zone (i.e., delegation points) as "really" belonging to the child zone. The corresponding domain names appear in two master files and might have RRsets signed by both the parent and child zones' keys. A retrieval could get a mixture of these RRsets and SIGs, especially since one server could be serving both the zone above and below a

delegation point [RFC 2181].

Each DS RRset stored in the parent zone MUST be signed by, at least, one of the parent zone's private key. The parent zone MUST NOT contain a KEY RRset at any delegation point. Delegations in the parent MAY contain only the following RR types: NS, DS, NXT and SIG. The NS RRset MUST NOT be signed. The NXT RRset is the exceptional case: it will always appear differently and authoritatively in both the parent and child zones if both are secure.

A secure zone MUST contain a self-signed KEY RRset at its apex. Upon verifying the DS RRset from the parent, a resolver MAY trust any KEY identified in the DS RRset as a valid signer of the child's apex KEY RRset. Resolvers configured to trust one of the keys signing the KEY RRset MAY now treat any data signed by the zone keys in the KEY RRset as secure. In all other cases resolvers MUST consider the zone unsecure. A DS RRset MUST NOT appear at a zone's apex.

An authoritative server queried for type DS MUST return the DS RRset in the answer section.

2.2.3 Signer's Name (replaces RFC3008 section 2.7)

The signer's name field of a SIG RR MUST contain the name of the zone to which the data and signature belong. The combination of signer's name, key tag, and algorithm MUST identify a zone key if the SIG is to be considered material. This document defines a standard policy for DNSSEC validation; local policy may override the standard policy.

There are no restrictions on the signer field of a SIG(0) record. The combination of signer's name, key tag, and algorithm MUST identify a key if this SIG(0) is to be processed.

2.2.4 Changes to RFC3090

A number of sections of RFC3090 need to be updated to reflect the DS record.

2.2.4.1 RFC3090: Updates to section 1: Introduction

Most of the text is still relevant but the words ``NULL key'' are to be replaced with ``missing DS RRset''. In section 1.3 the last three paragraphs discuss the confusion in sections of RFC 2535 that are replaced in section 2.2.1 above. Therefore, these paragraphs are now obsolete.

2.2.4.2 RFC3090 section 2.1: Globally Secured

Rule 2.1.b is replaced by the following rule:

2.1.b. The KEY RRset at a zone's apex MUST be self-signed by a private key whose public counterpart MUST appear in a zone signing KEY RR (2.a) owned by the zone's apex and specifying a mandatory-to-implement algorithm. This KEY RR MUST be identified by a DS RR in a signed DS RRset in the parent zone.

If a zone cannot get its parent to advertise a DS record for it, the child zone cannot be considered globally secured. The only exception to this is the root zone, for which there is no parent zone.

2.2.4.3 RFC3090 section 3: Experimental Status.

The only difference between experimental status and globally secured is the missing DS RRset in the parent zone. All locally secured zones are experimental.

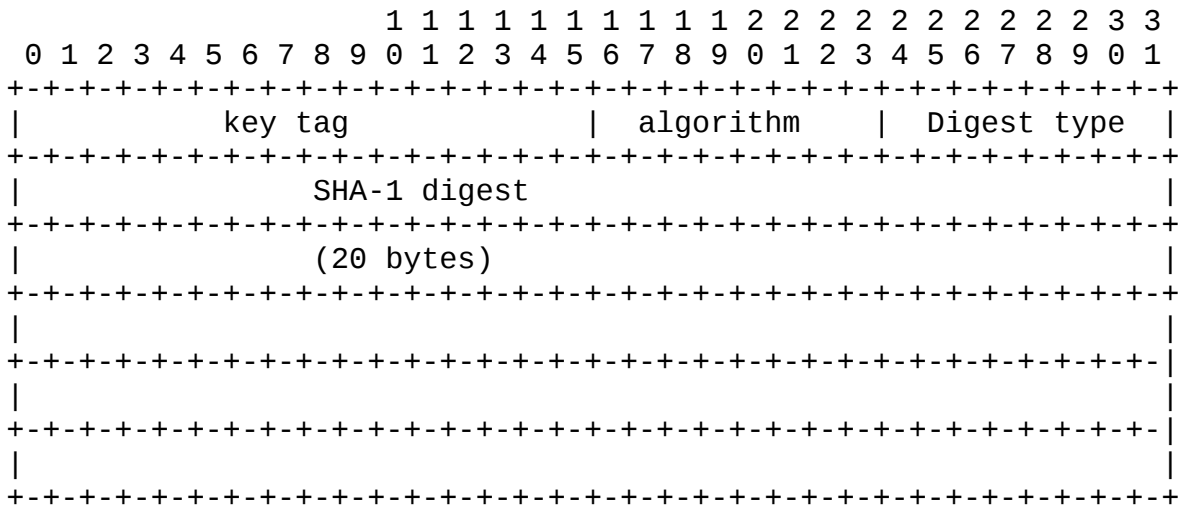
2.3 Comments on Protocol Changes

Over the years there have been various discussions surrounding the DNS delegation model, declaring it to be broken because there is no good way to assert if a delegation exists. In the RFC2535 version of DNSSEC, the presence of the NS bit in the NXT bit map proves there is a delegation at this name. Something more explicit is needed and the DS record addresses this need for secure delegations.

The DS record is a major change to DNS: it is the first resource record that can appear only on the upper side of a delegation. Adding it will cause interoperabilty problems and requires a flag day for DNSSEC. Many old servers and resolvers MUST be upgraded to take advantage of DS. Some old servers will be able to be authoritative for zones with DS records but will not add the NXT or DS records to the authority section. The same is true for caching servers; in fact, some may even refuse to pass on the DS or NXT records.

2.4 Wire Format of the DS record

The DS (type=TDB) record contains these fields: key tag, algorithm, digest type, and the digest of a public key KEY record that is allowed and/or used to sign the child's apex KEY RRset. Other keys MAY sign the child's apex KEY RRset.



The key tag is calculated as specified in RFC2535. Algorithm MUST be an algorithm number assigned in the range 1..251 and the algorithm MUST be allowed to sign DNS data. The digest type is an identifier for the digest algorithm used. The digest is calculated over the canonical name of the delegated domain name followed by the whole RDATA of the KEY record (all four fields).

```
digest = hash( canonical FQDN on KEY RR | KEY_RR_rdata)

KEY_RR_rdata = Flags | Protocol | Algorithm | Public Key
```

Digest type value 0 is reserved, value 1 is SHA-1, and reserving other types requires IETF standards action. For interoperabilty reasons, as few digest algorithms as possible should be reserved. The only reason to reserve additional digest types is to increase security.

DS records MUST point to zone KEY records that are allowed to authenticate DNS data. The indicated KEY record's protocol field MUST be set to 3; flag field bits 0 and 6 MUST be set to 0; bit 7 MUST be set to 1. The value of other bits is not significant for the purposes of this document.

The size of the DS RDATA for type 1 (SHA-1) is 24 bytes, regardless of key size, new digest types probably will have larger digests.

2.4.1 Justifications for Fields

The algorithm and key tag fields are present to allow resolvers to quickly identify the candidate KEY records to examine. SHA-1 is a strong cryptographic checksum: it is computationally infeasible for an attacker to generate a KEY record that has the same SHA-1 digest. Combining the name of the key and the key rdata as input to the digest provides stronger assurance of the binding. Having the key tag in the DS record adds greater assurance than the SHA-1 digest alone, as there are now two different mapping functions that a KEY RR must match.

This format allows concise representation of the keys that the child will use, thus keeping down the size of the answer for the delegation, reducing the probability of DNS message overflow. The SHA-1 hash is strong enough to uniquely identify the key and is similar to the PGP key footprint. The digest type field is present for possible future expansion.

The DS record is well suited to listing trusted keys for islands of security in configuration files.

2.5 Presentation Format of the DS Record

The presentation format of the DS record consists of three numbers (key tag, algorithm and digest type) followed by the digest itself presented in hex:

```
example.   DS  12345 3 1 123456789abcdef67890123456789abcdef67890
```

2.6 Transition Issues for Installed Base

No backwards compatibility with RFC2535 is provided.

RFC2535-compliant resolvers will assume that all DS-secured delegations are locally secure. This is bad, but the DNSEXT Working Group has determined that rather than dealing with both RFC2535-secured zones and DS-secured zones, a rapid adoption of DS is preferable. Thus the only option for early adopters is to upgrade to DS as soon as possible.

2.6.1 Backwards compatibility with RFC2535 and RFC1035

This section documents how a resolver determines the type of delegation.
RFC1035 delegation (in parent) has:

RFC1035 NS

RFC2535 adds the following two cases:

```
Secure RFC2535:   NS + NXT + SIG(NXT)
                  NXT bit map contains: NS SIG NXT
Unsecure RFC2535: NS + KEY + SIG(KEY) + NXT + SIG(NXT)
                  NXT bit map contains: NS SIG KEY NXT
                  KEY must be a NULL key.
```

DNSSEC with DS has the following two states:

```
Secure DS:        NS + DS + SIG(DS)
                  NXT bit map contains: NS SIG NXT DS
Unsecure DS:      NS + NXT + SIG(NXT)
                  NXT bit map contains: NS SIG NXT
```

It is difficult for a resolver to determine if a delegation is secure RFC 2535 or unsecure DS. This could be overcome by adding a flag to the NXT bit map, but only upgraded resolvers would understand this flag, anyway. Having both parent and child signatures for a KEY RRset might allow old resolvers to accept a zone as secure, but the cost of doing this for a long time is much higher than just prohibiting RFC 2535-style signatures at child zone apexes and forcing rapid deployment of DS-enabled servers and resolvers.

RFC 2535 and DS can in theory be deployed in parallel, but this would require resolvers to deal with RFC 2535 configurations forever. This document obsoletes the NULL KEY in parent zones, which is a difficult enough change that a flag day is required.

2.7 KEY and corresponding DS record example

This is a example of a KEY record and corresponding DS record.

```
dskey.example. KEY  256 3 1 (
               AQPwHb4UL1U9RHaU8qP+Ts5bVOU1s7fYbj2b3CCbzNdj
               4+/ECd18yKiyUQqKqQFWW5T3iVc8SJOKnueJHt/Jb/wt
               ) ; key id = 28668
          DS   28668 1  1  49FD46E6C4B45C55D4AC69CBD3CD34AC1AFE51DE
```

3 Resolver

3.1 DS Example

To create a chain of trust, a resolver goes from trusted KEY to DS to KEY.

Assume the key for domain "example." is trusted. Zone "example." contains at least the following records:

```
example.           SOA     <soa stuff>
example.           NS       ns.example.
example.           KEY     <stuff>
example.           NXT      NS SOA KEY SIG NXT secure.example.
example.           SIG(SOA)
example.           SIG(NS)
example.           SIG(NXT)
example.           SIG(KEY)
secure.example.    NS      ns1.secure.example.
secure.example.    DS      tag=12345 alg=3 digest_type=1 <foofoo>
secure.example.    NXT     NS SIG NXT DS unsecure.example.
secure.example.    SIG(NXT)
secure.example.    SIG(DS)
unsecure.example   NS      ns1.unsecure.example.
unsecure.example.  NXT     NS SIG NXT example.
unsecure.example.  SIG(NXT)
```

In zone "secure.example." following records exist:

```
secure.example.    SOA      <soa stuff>
secure.example.    NS       ns1.secure.example.
secure.example.    KEY      <tag=12345 alg=3>
secure.example.    KEY      <tag=54321 alg=5>
secure.example.    NXT      <nxt stuff>
secure.example.    SIG(KEY) <key-tag=12345 alg=3>
secure.example.    SIG(SOA) <key-tag=54321 alg=5>
secure.example.    SIG(NS)  <key-tag=54321 alg=5>
secure.example.    SIG(NXT) <key-tag=54321 alg=5>
```

In this example the private key for "example." signs the DS record for "secure.example.", making that a secure delegation. The DS record states which key is expected to sign the KEY RRset at "secure.example.". Here "secure.example." signs its KEY RRset with the KEY identified in the DS RRset, thus the KEY RRset is validated and trusted.

This example has only one DS record for the child, but parents MUST allow multiple DS records to facilitate key rollover and multiple KEY algorithms.

The resolver determines the security status of "unsecure.example." by examining the parent zone's NXT record for this name. The absence of the DS bit indicates an unsecure delegation. Note the NXT record SHOULD only be examined after verifying the corresponding signature.

3.1 Resolver Cost Estimates for DS Records

From a RFC2535 resolver point of view, for each delegation followed to chase down an answer, one KEY RRset has to be verified. Additional RRsets might also need to be verified based on local policy (e.g., the contents of the NS RRset). Once the resolver gets to the appropriate delegation, validating the answer might require verifying one or more signatures. A simple A record lookup requires at least N delegations to be verified and one RRset. For a DS-enabled resolver, the cost is 2N+1. For an MX record, where the target of the MX record is in the same zone as the MX record, the costs are N+2 and 2N+2, for RFC 2535 and DS, respectively. In the case of negatives answer the same ratios hold true.

The resolver may require an extra query to get the DS record and this may add to the overall cost of the query, but this is never worse than chasing down NULL KEY records from the parent in RFC2535 DNSSEC.

DS adds processing overhead on resolvers and increases the size of delegation answers, but much less than storing signatures in the parent zone.

4 Security Considerations:

This document proposes a change to the validation chain of KEY records in DNSSEC. The change is not believed to reduce security in the overall system. In RFC2535 DNSSEC, the child zone has to communicate keys to its parent and prudent parents will require some authentication with that transaction. The modified protocol will require the same authentication, but allows the child to exert more local control over its own KEY RRset.

There is a remote possibility that an attacker could generate a valid KEY that matches all the DS fields, of a specific DS set, and thus forge data from the child. This possibility is considered impractical, as on average more than

    2 ^ (160 - <Number of keys in DS set>)

keys would have to be generated before a match would be found.

An attacker that wants to match any DS record will have to generate on average at least 2^80 keys.

The DS record represents a change to the DNSSEC protocol and there is an installed base of implementations, as well as textbooks on how to

set up secure delegations. Implementations that do not understand the DS record will not be able to follow the KEY to DS to KEY chain and will consider all zones secured that way as unsecure.

5 IANA Considerations:

IANA needs to allocate an RR type code for DS from the standard RR type space (type 43 requested).

IANA needs to open a new registry for the DS RR type for digest algorithms. Defined types are:

0 is Reserved,
1 is SHA-1.

Adding new reservations requires IETF standards action.

6 Acknowledgments

Over the last few years a number of people have contributed ideas that are captured in this document. The core idea of using one key to sign only the KEY RRset comes from discussions with Bill Manning and Perry Metzger on how to put in a single root key in all resolvers. Alexis Yushin, Brian Wellington, Paul Vixie, Jakob Schlyter, Scott Rose, Edward Lewis, Lars-Johan Liman, Matt Larson, Mark Kosters, Dan Massey, Olaf Kolman, Phillip Hallam-Baker, Miek Gieben, Havard Eidnes, Donald Eastlake 3rd., Randy Bush, David Blacka, Steve Bellovin, Rob Austein, Derek Atkins, Roy Arends, Harald Alvestrand, and others have provided useful comments.

Normative References:

[RFC1035] P. Mockapetris, ``Domain Names - Implementation and Specification'', STD 13, RFC 1035, November 1987.

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

[RFC2535] D. Eastlake, ``Domain Name System Security Extensions'', RFC 2535, March 1999.

[RFC3008] B. Wellington, ``Domain Name System Security (DNSSEC) Signing Authority'', RFC 3008, November 2000.

[RFC3090] E. Lewis `` DNS Security Extension Clarification on Zone Status'', RFC 3090, March 2001.

[RFC3225] D. Conrad, ``Indicating Resolver Support of DNSSEC'', RFC 3225, December 2001.

[RFC3226] O. Gudmundsson, ``DNSSEC and IPv6 A6 aware server/resolver message size requirements'', RFC 3226, December 2001.

Author Address

Olafur Gudmundsson
3826 Legation Street, NW
Washington, DC, 20015
USA
<ogud@ogud.com>

Full Copyright Statement