        Clarifications and Implementation Notes for DNS Security (DNSSEC)

Abstract

   This document is a collection of technical clarifications to the DNS
   Security (DNSSEC) document set. It is meant to serve as a resource
   to implementors as well as a collection of DNSSEC errata that existed
   at the time of writing.

   This document updates the core DNSSEC documents (RFC 4033, RFC 4034,
   and RFC 4035) as well as the NSEC3 specification (RFC 5155). It also
   defines NSEC3 and SHA-2 (RFC 4509 and RFC 5702) as core parts of the
   DNSSEC specification.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 5741.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc6840.

Table of Contents

1. Introduction and Terminology

   This document lists some additions, clarifications, and corrections
   to the core DNSSEC specification, as originally described in
   [RFC4033], [RFC4034], and [RFC4035], and later amended by [RFC5155].
   (See Section 2 for more recent additions to that core document set.)

   It is intended to serve as a resource for implementors and as a
   repository of items existing at the time of writing that need to be
   addressed when advancing the DNSSEC documents along the Standards
   Track.

1.1. Structure of This Document

   The clarifications and changes to DNSSEC are sorted according to
   their importance, starting with ones which could, if ignored, lead to
   security problems and progressing down to clarifications that are
   expected to have little operational impact.

1.2. 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
   [RFC2119].

2. Important Additions to DNSSEC

   This section lists some documents that are now considered core DNSSEC
   protocol documents in addition to those originally specified in
   Section 10 of [RFC4033].

2.1. NSEC3 Support

   [RFC5155] describes the use and behavior of the NSEC3 and NSEC3PARAM
   records for hashed denial of existence. Validator implementations
   are strongly encouraged to include support for NSEC3 because a number
   of highly visible zones use it. Validators that do not support
   validation of responses using NSEC3 will be hampered in validating
   large portions of the DNS space.

   [RFC5155] is now considered part of the DNS Security Document Family
   as described by Section 10 of [RFC4033].

Note that the algorithm identifiers defined in [RFC5155] (DSA-NSEC3-
SHA1 and RSASHA1-NSEC3-SHA1) and [RFC5702] (RSASHA256 and RSASHA512)
signal that a zone might be using NSEC3, rather than NSEC. The zone
may be using either, and validators supporting these algorithms MUST
support both NSEC3 and NSEC responses.

2.2. SHA-2 Support

[RFC4509] describes the use of SHA-256 as a digest algorithm in
Delegation Signer (DS) RRs. [RFC5702] describes the use of the
RSASHA256 and RSASHA512 algorithms in DNSKEY and RRSIG RRs.
Validator implementations are strongly encouraged to include support
for these algorithms for DS, DNSKEY, and RRSIG records.

Both [RFC4509] and [RFC5702] are now considered part of the DNS
Security Document Family as described by Section 10 of [RFC4033].

3. Scaling Concerns

3.1. Implement a BAD Cache

Section 4.7 of [RFC4035] permits security-aware resolvers to
implement a BAD cache. That guidance has changed: security-aware
resolvers SHOULD implement a BAD cache as described in [RFC4035].

This change in guidance is based on operational experience with
DNSSEC administrative errors leading to significant increases in DNS
traffic, with an accompanying realization that such events are more
likely and more damaging than originally supposed. An example of one
such event is documented in "Rolling Over DNSSEC Keys" [Huston].

4. Security Concerns

This section provides clarifications that, if overlooked, could lead
to security issues.

4.1. Clarifications on Nonexistence Proofs

Section 5.4 of [RFC4035] under-specifies the algorithm for checking
nonexistence proofs. In particular, the algorithm as presented would
allow a validator to interpret an NSEC or NSEC3 RR from an ancestor
zone as proving the nonexistence of an RR in a child zone.

An "ancestor delegation" NSEC RR (or NSEC3 RR) is one with:

o the NS bit set,

o the Start of Authority (SOA) bit clear, and

o a signer field that is shorter than the owner name of the NSEC RR,
   or the original owner name for the NSEC3 RR.

Ancestor delegation NSEC or NSEC3 RRs MUST NOT be used to assume
nonexistence of any RRs below that zone cut, which include all RRs at
that (original) owner name other than DS RRs, and all RRs below that
owner name regardless of type.

Similarly, the algorithm would also allow an NSEC RR at the same
owner name as a DNAME RR, or an NSEC3 RR at the same original owner
name as a DNAME, to prove the nonexistence of names beneath that
DNAME. An NSEC or NSEC3 RR with the DNAME bit set MUST NOT be used
to assume the nonexistence of any subdomain of that NSEC/NSEC3 RR's
(original) owner name.

4.2. Validating Responses to an ANY Query

[RFC4035] does not address how to validate responses when QTYPE=*.
As described in Section 6.2.2 of [RFC1034], a proper response to
QTYPE=* may include a subset of the RRsets at a given name. That is,
it is not necessary to include all RRsets at the QNAME in the
response.

When validating a response to QTYPE=*, all received RRsets that match
QNAME and QCLASS MUST be validated. If any of those RRsets fail
validation, the answer is considered Bogus. If there are no RRsets
matching QNAME and QCLASS, that fact MUST be validated according to
the rules in Section 5.4 of [RFC4035] (as clarified in this
document). To be clear, a validator must not expect to receive all
records at the QNAME in response to QTYPE=*.

4.3. Check for CNAME

Section 5 of [RFC4035] says nothing explicit about validating
responses based on (or that should be based on) CNAMEs. When
validating a NOERROR/NODATA response, validators MUST check the CNAME
bit in the matching NSEC or NSEC3 RR's type bitmap in addition to the
bit for the query type.

Without this check, an attacker could successfully transform a
positive CNAME response into a NOERROR/NODATA response by (for
example) simply stripping the CNAME RRset from the response. A naive
validator would then note that the QTYPE was not present in the
matching NSEC/NSEC3 RR, but fail to notice that the CNAME bit was
set; thus, the response should have been a positive CNAME response.

4.4. Insecure Delegation Proofs

   Section 5.2 of [RFC4035] specifies that a validator, when proving a
   delegation is not secure, needs to check for the absence of the DS
   and SOA bits in the NSEC (or NSEC3) type bitmap. The validator also
   MUST check for the presence of the NS bit in the matching NSEC (or
   NSEC3) RR (proving that there is, indeed, a delegation), or
   alternately make sure that the delegation is covered by an NSEC3 RR
   with the Opt-Out flag set.

   Without this check, an attacker could reuse an NSEC or NSEC3 RR
   matching a non-delegation name to spoof an unsigned delegation at
   that name. This would claim that an existing signed RRset (or set of
   signed RRsets) is below an unsigned delegation, thus not signed and
   vulnerable to further attack.

5. Interoperability Concerns

5.1. Errors in Canonical Form Type Code List

   When canonicalizing DNS names (for both ordering and signing), DNS
   names in the RDATA section of NSEC resource records are not converted
   to lowercase. DNS names in the RDATA section of RRSIG resource
   records are converted to lowercase.

   The guidance in the above paragraph differs from what has been
   published before but is consistent with current common practice.
   Item 3 of Section 6.2 of [RFC4034] says that names in both of these
   RR types should be converted to lowercase. The earlier [RFC3755]
   says that they should not. Current practice follows neither document
   fully.

   Section 6.2 of [RFC4034] also erroneously lists HINFO as a record
   that needs conversion to lowercase, and twice at that. Since HINFO
   records contain no domain names, they are not subject to case
   conversion.

5.2. Unknown DS Message Digest Algorithms

   Section 5.2 of [RFC4035] includes rules for how to handle delegations
   to zones that are signed with entirely unsupported public key
   algorithms, as indicated by the key algorithms shown in those zones'
   DS RRsets. It does not explicitly address how to handle DS records
   that use unsupported message digest algorithms. In brief, DS records
   using unknown or unsupported message digest algorithms MUST be
   treated the same way as DS records referring to DNSKEY RRs of unknown
   or unsupported public key algorithms.

The existing text says:

    If the validator does not support any of the algorithms listed in
    an authenticated DS RRset, then the resolver has no supported
    authentication path leading from the parent to the child. The
    resolver should treat this case as it would the case of an
    authenticated NSEC RRset proving that no DS RRset exists, as
    described above.

In other words, when determining the security status of a zone, a
validator disregards any authenticated DS records that specify
unknown or unsupported DNSKEY algorithms. If none are left, the zone
is treated as if it were unsigned.

This document modifies the above text to additionally disregard
authenticated DS records using unknown or unsupported message digest
algorithms.

5.3. Private Algorithms

As discussed above, Section 5.2 of [RFC4035] requires that validators
make decisions about the security status of zones based on the public
key algorithms shown in the DS records for those zones. In the case
of private algorithms, as described in Appendix A.1.1 of [RFC4034],
the eight-bit algorithm field in the DS RR is not conclusive about
what algorithm(s) is actually in use.

If no private algorithms appear in the DS RRset, or if any supported
algorithm appears in the DS RRset, no special processing is needed.
Furthermore, if the validator implementation does not support any
private algorithms, or only supports private algorithms using an
algorithm number not present in the DS RRset, no special processing
is needed.

In the remaining cases, the security status of the zone depends on
whether or not the resolver supports any of the private algorithms in
use (provided that these DS records use supported message digest
algorithms, as discussed in Section 5.2 of this document). In these
cases, the resolver MUST retrieve the corresponding DNSKEY for each
private algorithm DS record and examine the public key field to
determine the algorithm in use. The security-aware resolver MUST
ensure that the hash of the DNSKEY RR's owner name and RDATA matches
the digest in the DS RR as described in Section 5.2 of [RFC4035],
authenticating the DNSKEY. If all of the retrieved and authenticated
DNSKEY RRs use unknown or unsupported private algorithms, then the
zone is treated as if it were unsigned.

   Note that if none of the private algorithm DS RRs can be securely
   matched to DNSKEY RRs and no other DS establishes that the zone is
   secure, the referral should be considered Bogus data as discussed in
   [RFC4035].

   This clarification facilitates the broader use of private algorithms,
   as suggested by [RFC4955].

5.4. Caution about Local Policy and Multiple RRSIGs

   When multiple RRSIGs cover a given RRset, Section 5.3.3 of [RFC4035]
   suggests that "the local resolver security policy determines whether
   the resolver also has to test these RRSIG RRs and how to resolve
   conflicts if these RRSIG RRs lead to differing results".

   This document specifies that a resolver SHOULD accept any valid RRSIG
   as sufficient, and only determine that an RRset is Bogus if all
   RRSIGs fail validation.

   If a resolver adopts a more restrictive policy, there's a danger that
   properly signed data might unnecessarily fail validation due to cache
   timing issues. Furthermore, certain zone management techniques, like
   the Double Signature Zone Signing Key Rollover method described in
   Section 4.2.1.2 of [RFC6781], will not work reliably. Such a
   resolver is also vulnerable to malicious insertion of gibberish
   signatures.

5.5. Key Tag Calculation

   Appendix B.1 of [RFC4034] incorrectly defines the Key Tag field
   calculation for algorithm 1. It correctly says that the Key Tag is
   the most significant 16 of the least significant 24 bits of the
   public key modulus. However, [RFC4034] then goes on to incorrectly
   say that this is fourth-to-last and third-to-last octets of the
   public key modulus. It is, in fact, the third-to-last and second-to-
   last octets.

5.6. Setting the DO Bit on Replies

   As stated in Section 3 of [RFC3225], the DNSSEC OK (DO) bit of the
   query MUST be copied in the response. However, in order to
   interoperate with implementations that ignore this rule on sending,
   resolvers MUST ignore the DO bit in responses.

5.7. Setting the AD Bit on Queries

   The semantics of the Authentic Data (AD) bit in the query were
   previously undefined. Section 4.6 of [RFC4035] instructed resolvers
   to always clear the AD bit when composing queries.

   This document defines setting the AD bit in a query as a signal
   indicating that the requester understands and is interested in the
   value of the AD bit in the response. This allows a requester to
   indicate that it understands the AD bit without also requesting
   DNSSEC data via the DO bit.

5.8. Setting the AD Bit on Replies

   Section 3.2.3 of [RFC4035] describes under which conditions a
   validating resolver should set or clear the AD bit in a response. In
   order to interoperate with legacy stub resolvers and middleboxes that
   neither understand nor ignore the AD bit, validating resolvers SHOULD
   only set the AD bit when a response both meets the conditions listed
   in Section 3.2.3 of [RFC4035], and the request contained either a set
   DO bit or a set AD bit.

5.9. Always Set the CD Bit on Queries

   When processing a request with the Checking Disabled (CD) bit set, a
   resolver SHOULD attempt to return all response data, even data that
   has failed DNSSEC validation. Section 3.2.2 of [RFC4035] requires a
   resolver processing a request with the CD bit set to set the CD bit
   on its upstream queries.

   This document further specifies that validating resolvers SHOULD set
   the CD bit on every upstream query. This is regardless of whether
   the CD bit was set on the incoming query or whether it has a trust
   anchor at or above the QNAME.

   [RFC4035] is ambiguous about what to do when a cached response was
   obtained with the CD bit unset, a case that only arises when the
   resolver chooses not to set the CD bit on all upstream queries, as
   specified above. In the typical case, no new query is required, nor
   does the cache need to track the state of the CD bit used to make a
   given query. The problem arises when the cached response is a server
   failure (RCODE 2), which may indicate that the requested data failed
   DNSSEC validation at an upstream validating resolver. ([RFC2308]
   permits caching of server failures for up to five minutes.) In these
   cases, a new query with the CD bit set is required.

   Appendix B discusses more of the logic behind the recommendation
   presented in this section.

5.10. Nested Trust Anchors

   A DNSSEC validator may be configured such that, for a given response,
   more than one trust anchor could be used to validate the chain of
   trust to the response zone. For example, imagine a validator
   configured with trust anchors for "example." and "zone.example."
   When the validator is asked to validate a response to
   "www.sub.zone.example.", either trust anchor could apply.

   When presented with this situation, DNSSEC validators have a choice
   of which trust anchor(s) to use. Which to use is a matter of
   implementation choice. Appendix C discusses several possible
   algorithms.

   It is possible and advisable to expose the choice of policy as a
   configuration option. As a default, it is suggested that validators
   implement the "Accept Any Success" policy described in Appendix C.2
   while exposing other policies as configuration options.

   The "Accept Any Success" policy is to try all applicable trust
   anchors until one gives a validation result of Secure, in which case
   the final validation result is Secure. If and only if all applicable
   trust anchors give a result of Insecure, the final validation result
   is Insecure. If one or more trust anchors lead to a Bogus result and
   there is no Secure result, then the final validation result is Bogus.

5.11. Mandatory Algorithm Rules

   The last paragraph of Section 2.2 of [RFC4035] includes rules
   describing which algorithms must be used to sign a zone. Since these
   rules have been confusing, they are restated using different language
   here:

      The DS RRset and DNSKEY RRset are used to signal which algorithms
      are used to sign a zone. The presence of an algorithm in either a
      zone's DS or DNSKEY RRset signals that that algorithm is used to
      sign the entire zone.

      A signed zone MUST include a DNSKEY for each algorithm present in
      the zone's DS RRset and expected trust anchors for the zone. The
      zone MUST also be signed with each algorithm (though not each key)
      present in the DNSKEY RRset. It is possible to add algorithms at
      the DNSKEY that aren't in the DS record, but not vice versa. If
      more than one key of the same algorithm is in the DNSKEY RRset, it
      is sufficient to sign each RRset with any subset of these DNSKEYs.
      It is acceptable to sign some RRsets with one subset of keys (or
      key) and other RRsets with a different subset, so long as at least

        one DNSKEY of each algorithm is used to sign each RRset.
        Likewise, if there are DS records for multiple keys of the same
        algorithm, any subset of those may appear in the DNSKEY RRset.

    This requirement applies to servers, not validators. Validators
    SHOULD accept any single valid path. They 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. A validator
    MAY have a configuration option to perform a signature completeness
    test to support troubleshooting.

5.12. Ignore Extra Signatures from Unknown Keys

    Validating resolvers MUST disregard RRSIGs in a zone that do not
    (currently) have a corresponding DNSKEY in the zone. Similarly, a
    validating resolver MUST disregard RRSIGs with algorithm types that
    don't exist in the DNSKEY RRset.

    Good key rollover and algorithm rollover practices, as discussed in
    RFC 6781 and its successor documents and as suggested by the rules in
    the previous section, may require that such RRSIGs be present in a
    zone.

6. Minor Corrections and Clarifications

6.1. Finding Zone Cuts

    Appendix C.8 of [RFC4035] discusses sending DS queries to the servers
    for a parent zone but does not state how to find those servers.
    Specific instructions can be found in Section 4.2 of [RFC4035].

6.2. Clarifications on DNSKEY Usage

    It is possible to use different DNSKEYs to sign different subsets of
    a zone, constrained only by the rules in Section 5.11. It is even
    possible to use a different DNSKEY for each RRset in a zone, subject
    only to practical limits on the size of the DNSKEY RRset and the
    above rules. However, be aware that there is no way to tell
    resolvers what a particular DNSKEY is supposed to be used for -- any
    DNSKEY in the zone's signed DNSKEY RRset may be used to authenticate
    any RRset in the zone. For example, if a weaker or less trusted
    DNSKEY is being used to authenticate NSEC RRsets or all dynamically
    updated records, that same DNSKEY can also be used to sign any other
    RRsets from the zone.

    Furthermore, note that the SEP bit setting has no effect on how a
    DNSKEY may be used -- the validation process is specifically
    prohibited from using that bit by Section 2.1.2 of [RFC4034]. It is

   possible to use a DNSKEY without the SEP bit set as the sole secure
   entry point to the zone, yet use a DNSKEY with the SEP bit set to
   sign all RRsets in the zone (other than the DNSKEY RRset). It is
   also possible to use a single DNSKEY, with or without the SEP bit
   set, to sign the entire zone, including the DNSKEY RRset itself.

6.3. Errors in Examples

   The text in Appendix C.1 of [RFC4035] refers to the examples in
   Appendix B.1 as "x.w.example.com" while B.1 uses "x.w.example". This
   is painfully obvious in the second paragraph where it states that the
   RRSIG labels field value of 3 indicates that the answer was not the
   result of wildcard expansion. This is true for "x.w.example" but not
   for "x.w.example.com", which of course has a label count of 4
   (antithetically, a label count of 3 would imply the answer was the
   result of a wildcard expansion).

   The first paragraph of Appendix C.6 of [RFC4035] also has a minor
   error: the reference to "a.z.w.w.example" should instead be
   "a.z.w.example", as in the previous line.

6.4. Errors in RFC 5155

   An NSEC3 record that matches an Empty Non-Terminal effectively has no
   type associated with it. This NSEC3 record has an empty type bit
   map. Section 3.2.1 of [RFC5155] contains the statement:

      Blocks with no types present MUST NOT be included.

   However, the same section contains a regular expression:

      Type Bit Maps Field = (Window Block # | Bitmap Length | Bitmap)+

   The plus sign in the regular expression indicates that there is one
   or more of the preceding element. This means that there must be at
   least one window block. If this window block has no types, it
   contradicts with the first statement. Therefore, the correct text in
   Section 3.2.1 of [RFC5155] should be:

      Type Bit Maps Field = (Window Block # | Bitmap Length | Bitmap)*

7. Security Considerations

   This document adds SHA-2 and NSEC3 support to the core DNSSEC
   protocol. Security considerations for those features are discussed
   in the documents defining them. Additionally, this document
   addresses some ambiguities and omissions in the core DNSSEC documents
   that, if not recognized and addressed in implementations, could lead

   to security failures. In particular, the validation algorithm
   clarifications in Section 4 are critical for preserving the security
   properties DNSSEC offers. Furthermore, failure to address some of
   the interoperability concerns in Section 5 could limit the ability to
   later change or expand DNSSEC, including adding new algorithms.

   The recommendation in Section 5.9 to always set the CD bit has
   security implications. By setting the CD bit, a resolver will not
   benefit from more stringent validation rules or a more complete set
   of trust anchors at an upstream validator.

8. References

8.1. Normative References

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

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

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

   [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.

   [RFC4509] Hardaker, W., "Use of SHA-256 in DNSSEC Delegation Signer
              (DS) Resource Records (RRs)", RFC 4509, May 2006.

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

   [RFC5702] Jansen, J., "Use of SHA-2 Algorithms with RSA in DNSKEY
              and RRSIG Resource Records for DNSSEC", RFC 5702,
              October 2009.

8.2. Informative References

   [Huston]   Michaelson, G., Wallstrom, P., Arends, R., and G. Huston,
               "Rolling Over DNSSEC Keys", Internet Protocol
               Journal, Vol. 13, No.1, pp. 2-16, March 2010.

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

   [RFC3755]   Weiler, S., "Legacy Resolver Compatibility for Delegation
               Signer (DS)", RFC 3755, May 2004.

   [RFC4955]   Blacka, D., "DNS Security (DNSSEC) Experiments", RFC 4955,
               July 2007.

   [RFC5011]   StJohns, M., "Automated Updates of DNS Security (DNSSEC)
               Trust Anchors", STD 74, RFC 5011, September 2007.

   [RFC5074]   Weiler, S., "DNSSEC Lookaside Validation (DLV)", RFC 5074,
               November 2007.

   [RFC6781]   Kolkman, O., Mekking, W., and R. Gieben, "DNSSEC
               Operational Practices, Version 2", RFC 6781,
               December 2012.

Appendix A.   Acknowledgments

   The editors would like the thank Rob Austein for his previous work as
   an editor of this document.

   The editors are extremely grateful to those who, in addition to
   finding errors and omissions in the DNSSEC document set, have
   provided text suitable for inclusion in this document.

   The lack of specificity about handling private algorithms, as
   described in Section 5.3, and the lack of specificity in handling ANY
   queries, as described in Section 4.2, were discovered by David
   Blacka.

   The error in algorithm 1 key tag calculation, as described in
   Section 5.5, was found by Abhijit Hayatnagarkar. Donald Eastlake
   contributed text for Section 5.5.

   The bug relating to delegation NSEC RR's in Section 4.1 was found by
   Roy Badami. Roy Arends found the related problem with DNAME.

   The errors in the [RFC4035] examples were found by Roy Arends, who
   also contributed text for Section 6.3 of this document.

   Text on the mandatory algorithm rules was derived from suggestions by
   Matthijs Mekking and Ed Lewis.

   The CD bit logic was analyzed in depth by David Blacka, Olafur
   Gudmundsson, Mike St. Johns, and Andrew Sullivan.

   The editors would like to thank Alfred Hoenes, Ed Lewis, Danny Mayer,
   Olafur Gudmundsson, Suzanne Woolf, Rickard Bellgrim, Mike St. Johns,
   Mark Andrews, Wouter Wijngaards, Matthijs Mekking, Andrew Sullivan,
   Jeremy Reed, Paul Hoffman, Mohan Parthasarathy, Florian Weimer,
   Warren Kumari and Scott Rose for their contributions to this
   document.

Appendix B.   Discussion of Setting the CD Bit

   [RFC4035] may be read as relying on the implicit assumption that
   there is at most one validating system between the stub resolver and
   the authoritative server for a given zone. It is entirely possible,
   however, for more than one validator to exist between a stub resolver
   and an authoritative server. If these different validators have
   disjoint trust anchors configured, then it is possible that each
   would be able to validate some portion of the DNS tree, but neither

is able to validate all of it. Accordingly, it might be argued that
it is desirable not to set the CD bit on upstream queries, because
that allows for maximal validation.

In Section 5.9 of this document, it is recommended to set the CD bit
on an upstream query even when the incoming query arrives with CD=0.
This is for two reasons: it encourages a more predictable validation
experience as only one validator is always doing the validation, and
it ensures that all DNSSEC data that exists may be available from the
local cache should a query with CD=1 arrive.

As a matter of policy, it is possible to set the CD bit differently
than suggested in Section 5.9. A different choice will, of course,
not always yield the benefits listed above. It is beyond the scope
of this document to outline all of the considerations and counter
considerations for all possible policies. Nevertheless, it is
possible to describe three approaches and their underlying philosophy
of operation. These are laid out in the tables below.

The table that describes each model has five columns. The first
column indicates the value of the CD bit that the resolver receives
(for instance, on the name server side in an iterative resolver, or
as local policy or from the API in the case of a stub). The second
column indicates whether the query needs to be forwarded for
resolution (F) or can be satisfied from a local cache (C). The third
column is a line number, so that it can be referred to later in the
table. The fourth column indicates any relevant conditions at the
resolver, for example, whether the resolver has a covering trust
anchor, and so on. If there are no parameters here, the column is
empty. The fifth and final column indicates what action the resolver
takes.

The tables differentiate between "cached data" and "cached RCODE=2".
This is a shorthand; the point is that one has to treat RCODE=2
(server failure) as special, because it might indicate a validation
failure somewhere upstream. The distinction is really between
"cached RCODE=2" and "cached everything else".

The tables are probably easiest to think of in terms of describing
what happens when a stub resolver sends a query to an intermediate
resolver, but they are perfectly general and can be applied to any
validating resolver.

Model 1: "always set"

This model is so named because the validating resolver sets the CD
bit on queries it makes regardless of whether it has a covering trust
anchor for the query. The general philosophy represented by this

table is that only one resolver should be responsible for validation
irrespective of the possibility that an upstream resolver may be
present with trust anchors that cover different or additional QNAMEs.
It is the model recommended in Section 5.9 of this document.

```
CD F/C    line      conditions            action
=================================================================
1   F     A1                              Set CD=1 on upstream query
0   F     A2                              Set CD=1 on upstream query
1   C     A3                              Return the cache contents
                                           (data or RCODE=2)
0   C     A4        no covering TA        Return cache contents
                                           (data or RCODE=2)
0   C     A5        covering TA           Validate cached result and
                                            return it
```

Model 2: "never set when receiving CD=0"

This model is so named because it sets CD=0 on upstream queries for
all received CD=0 queries, even if it has a covering trust anchor.
The general philosophy represented by this table is that more than
one resolver may take responsibility for validating a QNAME and that
a validation failure for a QNAME by any resolver in the chain is a
validation failure for the query. Using this model is NOT
RECOMMENDED.

```
CD F/C    line      conditions            action
=================================================================
1   F     N1                              Set CD=1 on upstream query
0   F     N2                              Set CD=0 on upstream query
1   C     N3        cached data           Return cached data
1   C     N4        cached RCODE=2        Treat as line N1
0   C     N5        no covering TA        Return cache contents
                                           (data or RCODE=2)
0   C     N6        covering TA &         Treat as line N2
                    cached data was
                    generated with CD=1
0   C     N7        covering TA &         Validate and return
                    cached data was
                    generated with CD=0
```

Model 3: "sometimes set"

This model is so named because it sets the CD bit on upstream queries
triggered by received CD=0 queries, based on whether the validator
has a trust anchor configured that covers the query. If there is no
covering trust anchor, the resolver clears the CD bit in the upstream

query. If there is a covering trust anchor, the resolver sets CD=1
and performs validation itself. The general philosophy represented
by this table is that a resolver should try and validate QNAMEs for
which it has trust anchors and should not preclude validation by
other resolvers for QNAMEs for which it does not have covering trust
anchors. Using this model is NOT RECOMMENDED.

```
CD F/C   line      conditions         action
==================================================================
1  F     S1                           Set CD=1 on upstream query
0  F     S2        covering TA        Set CD=1 on upstream query
0  F     S3        no covering TA     Set CD=0 on upstream query
1  C     S4        cached data        Return cached data
1  C     S5        cached RCODE=2     Treat as line S1
0  C     S6        cached data was    Return cache contents
                    generated with
                    CD=0
0  C     S7        cached data was    Validate & return cache
                    generated with     contents
                    CD=1 &
                    covering TA
0  C     S8        cached RCODE=2     Return cache contents
0  C     S9        cached data        Treat as line S3
                    was generated
                    with CD=1 &
                    no covering
                    TA
```

Appendix C. Discussion of Trust Anchor Preference Options

   This section presents several different policies for validating
   resolvers to use when they have a choice of trust anchors available
   for validating a given answer.

C.1. Closest Encloser

   One policy is to choose the trust anchor closest to the QNAME of the
   response. For example, consider a validator configured with trust
   anchors for "example." and "zone.example." When asked to validate a
   response for "www.sub.zone.example.", a validator using the "Closest
   Encloser" policy would choose the "zone.example." trust anchor.

   This policy has the advantage of allowing the operator to trivially
   override a parent zone's trust anchor with one that the operator can
   validate in a stronger way, perhaps because the resolver operator is

affiliated with the zone in question. This policy also minimizes the
number of public key operations needed, which is of benefit in
resource-constrained environments.

This policy has the disadvantage of giving the user some unexpected
and unnecessary validation failures when sub-zone trust anchors are
neglected. As a concrete example, consider a validator that
configured a trust anchor for "zone.example." in 2009 and one for
"example." in 2011. In 2012, "zone.example." rolls its Key Signing
Key (KSK) and updates its DS records, but the validator operator
doesn't update its trust anchor. With the "Closest Encloser" policy,
the validator gets validation failures.

C.2. Accept Any Success

Another policy is to try all applicable trust anchors until one gives
a validation result of Secure, in which case the final validation
result is Secure. If and only if all applicable trust anchors give a
result of Insecure, the final validation result is Insecure. If one
or more trust anchors lead to a Bogus result and there is no Secure
result, then the final validation result is Bogus.

This has the advantage of causing the fewest validation failures,
which may deliver a better user experience. If one trust anchor is
out of date (as in our above example), the user may still be able to
get a Secure validation result (and see DNS responses).

This policy has the disadvantage of making the validator subject to
the compromise of the weakest of these trust anchors, while making it
relatively painless to keep old trust anchors configured in
perpetuity.

C.3. Preference Based on Source

When the trust anchors have come from different sources (e.g.,
automated updates ([RFC5011]), one or more DNSSEC Lookaside
Validation (DLV) registries ([RFC5074]), and manual configuration), a
validator may wish to choose between them based on the perceived
reliability of those sources. The order of precedence might be
exposed as a configuration option.

For example, a validator might choose to prefer trust anchors found
in a DLV registry over those manually configured on the theory that
the manually configured ones will not be as aggressively maintained.

   Conversely, a validator might choose to prefer manually configured
   trust anchors over those obtained from a DLV registry on the theory
   that the manually configured ones have been more carefully
   authenticated.

   Or the validator might do something more complex: prefer a sub-set of
   manually configured trust anchors (based on a configuration option),
   then trust anchors that have been updated using the mechanism in
   [RFC5011], then trust anchors from one DLV registry, then trust
   anchors from a different DLV registry, then the rest of the manually
   configured trust anchors.

Authors' Addresses

   Samuel Weiler (editor)
   SPARTA, Inc.
   7110 Samuel Morse Drive
   Columbia, MD 21046
   US

   EMail: weiler@tislabs.com

   David Blacka (editor)
   Verisign, Inc.
   12061 Bluemont Way
   Reston, VA 20190
   US

   EMail: davidb@verisign.com