               Protocol Modifications for the DNS Security Extensions

Status of This Memo

Copyright Notice

Abstract

   This document is part of a family of documents that describe the DNS
   Security Extensions (DNSSEC). The DNS Security Extensions are a
   collection of new resource records and protocol modifications that
   add data origin authentication and data integrity to the DNS. This
   document describes the DNSSEC protocol modifications. This document
   defines the concept of a signed zone, along with the requirements for
   serving and resolving by using DNSSEC. These techniques allow a
   security-aware resolver to authenticate both DNS resource records and
   authoritative DNS error indications.

   This document obsoletes RFC 2535 and incorporates changes from all
   updates to RFC 2535.

Table of Contents

1. Introduction

   The DNS Security Extensions (DNSSEC) are a collection of new resource
   records and protocol modifications that add data origin
   authentication and data integrity to the DNS. This document defines
   the DNSSEC protocol modifications. Section 2 of this document
   defines the concept of a signed zone and lists the requirements for
   zone signing. Section 3 describes the modifications to authoritative
   name server behavior necessary for handling signed zones. Section 4
   describes the behavior of entities that include security-aware
   resolver functions. Finally, Section 5 defines how to use DNSSEC RRs
   to authenticate a response.

1.1. Background and Related Documents

   This document is part of a family of documents defining DNSSEC that
   should be read together as a set.

   [RFC4033] contains an introduction to DNSSEC and definitions of
   common terms; the reader is assumed to be familiar with this
   document. [RFC4033] also contains a list of other documents updated
   by and obsoleted by this document set.

   [RFC4034] defines the DNSSEC resource records.

   The reader is also assumed to be familiar with the basic DNS concepts
   described in [RFC1034], [RFC1035], and the subsequent documents that
   update them; particularly, [RFC2181] and [RFC2308].

   This document defines the DNSSEC protocol operations.

1.2. Reserved Words

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

2. Zone Signing

   DNSSEC introduces the concept of signed zones. A signed zone
   includes DNS Public Key (DNSKEY), Resource Record Signature (RRSIG),
   Next Secure (NSEC), and (optionally) Delegation Signer (DS) records
   according to the rules specified in Sections 2.1, 2.2, 2.3, and 2.4,
   respectively. A zone that does not include these records according
   to the rules in this section is an unsigned zone.

   DNSSEC requires a change to the definition of the CNAME resource
   record ([RFC1035]). Section 2.5 changes the CNAME RR to allow RRSIG
   and NSEC RRs to appear at the same owner name as does a CNAME RR.

   DNSSEC specifies the placement of two new RR types, NSEC and DS,
   which can be placed at the parental side of a zone cut (that is, at a
   delegation point). This is an exception to the general prohibition
   against putting data in the parent zone at a zone cut. Section 2.6
   describes this change.

2.1. Including DNSKEY RRs in a Zone

   To sign a zone, the zone's administrator generates one or more
   public/private key pairs and uses the private key(s) to sign
   authoritative RRsets in the zone. For each private key used to
   create RRSIG RRs in a zone, the zone SHOULD include a zone DNSKEY RR
   containing the corresponding public key. A zone key DNSKEY RR MUST
   have the Zone Key bit of the flags RDATA field set (see Section 2.1.1
   of [RFC4034]). Public keys associated with other DNS operations MAY
   be stored in DNSKEY RRs that are not marked as zone keys but MUST NOT
   be used to verify RRSIGs.

   If the zone administrator intends a signed zone to be usable other
   than as an island of security, the zone apex MUST contain at least
   one DNSKEY RR to act as a secure entry point into the zone. This
   secure entry point could then be used as the target of a secure
   delegation via a corresponding DS RR in the parent zone (see
   [RFC4034]).

2.2. Including RRSIG RRs in a Zone

   For each authoritative RRset in a signed zone, there MUST be at least
   one RRSIG record that meets the following requirements:

   o The RRSIG owner name is equal to the RRset owner name.

   o The RRSIG class is equal to the RRset class.

   o The RRSIG Type Covered field is equal to the RRset type.

   o The RRSIG Original TTL field is equal to the TTL of the RRset.

   o The RRSIG RR's TTL is equal to the TTL of the RRset.

   o The RRSIG Labels field is equal to the number of labels in the
      RRset owner name, not counting the null root label and not
      counting the leftmost label if it is a wildcard.

   o The RRSIG Signer's Name field is equal to the name of the zone
      containing the RRset.

   o The RRSIG Algorithm, Signer's Name, and Key Tag fields identify a
      zone key DNSKEY record at the zone apex.

   The process for constructing the RRSIG RR for a given RRset is
   described in [RFC4034]. An RRset MAY have multiple RRSIG RRs
   associated with it. Note that as RRSIG RRs are closely tied to the
   RRsets whose signatures they contain, RRSIG RRs, unlike all other DNS

RR types, do not form RRsets. In particular, the TTL values among
RRSIG RRs with a common owner name do not follow the RRset rules
described in [RFC2181].

An RRSIG RR itself MUST NOT be signed, as signing an RRSIG RR would
add no value and would create an infinite loop in the signing
process.

The NS RRset that appears at the zone apex name MUST be signed, but
the NS RRsets that appear at delegation points (that is, the NS
RRsets in the parent zone that delegate the name to the child zone's
name servers) MUST NOT be signed. Glue address RRsets associated
with delegations MUST NOT be signed.

There MUST be an RRSIG for each RRset using at least one DNSKEY of
each algorithm in the zone apex DNSKEY RRset. The apex DNSKEY RRset
itself MUST be signed by each algorithm appearing in the DS RRset
located at the delegating parent (if any).

2.3. Including NSEC RRs in a Zone

Each owner name in the zone that has authoritative data or a
delegation point NS RRset MUST have an NSEC resource record. The
format of NSEC RRs and the process for constructing the NSEC RR for a
given name is described in [RFC4034].

The TTL value for any NSEC RR SHOULD be the same as the minimum TTL
value field in the zone SOA RR.

An NSEC record (and its associated RRSIG RRset) MUST NOT be the only
RRset at any particular owner name. That is, the signing process
MUST NOT create NSEC or RRSIG RRs for owner name nodes that were not
the owner name of any RRset before the zone was signed. The main
reasons for this are a desire for namespace consistency between
signed and unsigned versions of the same zone and a desire to reduce
the risk of response inconsistency in security oblivious recursive
name servers.

The type bitmap of every NSEC resource record in a signed zone MUST
indicate the presence of both the NSEC record itself and its
corresponding RRSIG record.

The difference between the set of owner names that require RRSIG
records and the set of owner names that require NSEC records is
subtle and worth highlighting. RRSIG records are present at the
owner names of all authoritative RRsets. NSEC records are present at
the owner names of all names for which the signed zone is
authoritative and also at the owner names of delegations from the

signed zone to its children. Neither NSEC nor RRSIG records are
present (in the parent zone) at the owner names of glue address
RRsets. Note, however, that this distinction is for the most part
visible only during the zone signing process, as NSEC RRsets are
authoritative data and are therefore signed. Thus, any owner name
that has an NSEC RRset will have RRSIG RRs as well in the signed
zone.

The bitmap for the NSEC RR at a delegation point requires special
attention. Bits corresponding to the delegation NS RRset and any
RRsets for which the parent zone has authoritative data MUST be set;
bits corresponding to any non-NS RRset for which the parent is not
authoritative MUST be clear.

2.4. Including DS RRs in a Zone

The DS resource record establishes authentication chains between DNS
zones. A DS RRset SHOULD be present at a delegation point when the
child zone is signed. The DS RRset MAY contain multiple records,
each referencing a public key in the child zone used to verify the
RRSIGs in that zone. All DS RRsets in a zone MUST be signed, and DS
RRsets MUST NOT appear at a zone's apex.

A DS RR SHOULD point to a DNSKEY RR that is present in the child's
apex DNSKEY RRset, and the child's apex DNSKEY RRset SHOULD be signed
by the corresponding private key. DS RRs that fail to meet these
conditions are not useful for validation, but because the DS RR and
its corresponding DNSKEY RR are in different zones, and because the
DNS is only loosely consistent, temporary mismatches can occur.

The TTL of a DS RRset SHOULD match the TTL of the delegating NS RRset
(that is, the NS RRset from the same zone containing the DS RRset).

Construction of a DS RR requires knowledge of the corresponding
DNSKEY RR in the child zone, which implies communication between the
child and parent zones. This communication is an operational matter
not covered by this document.

2.5. Changes to the CNAME Resource Record

If a CNAME RRset is present at a name in a signed zone, appropriate
RRSIG and NSEC RRsets are REQUIRED at that name. A KEY RRset at that
name for secure dynamic update purposes is also allowed ([RFC3007]).
Other types MUST NOT be present at that name.

This is a modification to the original CNAME definition given in
[RFC1034]. The original definition of the CNAME RR did not allow any
other types to coexist with a CNAME record, but a signed zone

requires NSEC and RRSIG RRs for every authoritative name. To resolve
this conflict, this specification modifies the definition of the
CNAME resource record to allow it to coexist with NSEC and RRSIG RRs.

2.6. DNSSEC RR Types Appearing at Zone Cuts

   DNSSEC introduced two new RR types that are unusual in that they can
   appear at the parental side of a zone cut. At the parental side of a
   zone cut (that is, at a delegation point), NSEC RRs are REQUIRED at
   the owner name. A DS RR could also be present if the zone being
   delegated is signed and seeks to have a chain of authentication to
   the parent zone. This is an exception to the original DNS
   specification ([RFC1034]), which states that only NS RRsets could
   appear at the parental side of a zone cut.

   This specification updates the original DNS specification to allow
   NSEC and DS RR types at the parent side of a zone cut. These RRsets
   are authoritative for the parent when they appear at the parent side
   of a zone cut.

2.7. Example of a Secure Zone

   Appendix A shows a complete example of a small signed zone.

3. Serving

   This section describes the behavior of entities that include
   security-aware name server functions. In many cases such functions
   will be part of a security-aware recursive name server, but a
   security-aware authoritative name server has some of the same
   requirements. Functions specific to security-aware recursive name
   servers are described in Section 3.2; functions specific to
   authoritative servers are described in Section 3.1.

   In the following discussion, the terms "SNAME", "SCLASS", and "STYPE"
   are as used in [RFC1034].

   A security-aware name server MUST support the EDNS0 ([RFC2671])
   message size extension, MUST support a message size of at least 1220
   octets, and SHOULD support a message size of 4000 octets. As IPv6
   packets can only be fragmented by the source host, a security aware
   name server SHOULD take steps to ensure that UDP datagrams it
   transmits over IPv6 are fragmented, if necessary, at the minimum IPv6
   MTU, unless the path MTU is known. Please see [RFC1122], [RFC2460],
   and [RFC3226] for further discussion of packet size and fragmentation
   issues.

   A security-aware name server that receives a DNS query that does not
   include the EDNS OPT pseudo-RR or that has the DO bit clear MUST
   treat the RRSIG, DNSKEY, and NSEC RRs as it would any other RRset and
   MUST NOT perform any of the additional processing described below.
   Because the DS RR type has the peculiar property of only existing in
   the parent zone at delegation points, DS RRs always require some
   special processing, as described in Section 3.1.4.1.

   Security aware name servers that receive explicit queries for
   security RR types that match the content of more than one zone that
   it serves (for example, NSEC and RRSIG RRs above and below a
   delegation point where the server is authoritative for both zones)
   should behave self-consistently. As long as the response is always
   consistent for each query to the name server, the name server MAY
   return one of the following:

   o The above-delegation RRsets.
   o The below-delegation RRsets.
   o Both above and below-delegation RRsets.
   o Empty answer section (no records).
   o Some other response.
   o An error.

   DNSSEC allocates two new bits in the DNS message header: the CD
   (Checking Disabled) bit and the AD (Authentic Data) bit. The CD bit
   is controlled by resolvers; a security-aware name server MUST copy
   the CD bit from a query into the corresponding response. The AD bit
   is controlled by name servers; a security-aware name server MUST
   ignore the setting of the AD bit in queries. See Sections 3.1.6,
   3.2.2, 3.2.3, 4, and 4.9 for details on the behavior of these bits.

   A security aware name server that synthesizes CNAME RRs from DNAME
   RRs as described in [RFC2672] SHOULD NOT generate signatures for the
   synthesized CNAME RRs.

3.1. Authoritative Name Servers

   Upon receiving a relevant query that has the EDNS ([RFC2671]) OPT
   pseudo-RR DO bit ([RFC3225]) set, a security-aware authoritative name
   server for a signed zone MUST include additional RRSIG, NSEC, and DS
   RRs, according to the following rules:

   o RRSIG RRs that can be used to authenticate a response MUST be
      included in the response according to the rules in Section 3.1.1.

o NSEC RRs that can be used to provide authenticated denial of
   existence MUST be included in the response automatically according
   to the rules in Section 3.1.3.

o Either a DS RRset or an NSEC RR proving that no DS RRs exist MUST
   be included in referrals automatically according to the rules in
   Section 3.1.4.

These rules only apply to responses where the semantics convey
information about the presence or absence of resource records. That
is, these rules are not intended to rule out responses such as RCODE
4 ("Not Implemented") or RCODE 5 ("Refused").

DNSSEC does not change the DNS zone transfer protocol. Section 3.1.5
discusses zone transfer requirements.

3.1.1. Including RRSIG RRs in a Response

When responding to a query that has the DO bit set, a security-aware
authoritative name server SHOULD attempt to send RRSIG RRs that a
security-aware resolver can use to authenticate the RRsets in the
response. A name server SHOULD make every attempt to keep the RRset
and its associated RRSIG(s) together in a response. Inclusion of
RRSIG RRs in a response is subject to the following rules:

o When placing a signed RRset in the Answer section, the name server
   MUST also place its RRSIG RRs in the Answer section. The RRSIG
   RRs have a higher priority for inclusion than any other RRsets
   that may have to be included. If space does not permit inclusion
   of these RRSIG RRs, the name server MUST set the TC bit.

o When placing a signed RRset in the Authority section, the name
   server MUST also place its RRSIG RRs in the Authority section.
   The RRSIG RRs have a higher priority for inclusion than any other
   RRsets that may have to be included. If space does not permit
   inclusion of these RRSIG RRs, the name server MUST set the TC bit.

o When placing a signed RRset in the Additional section, the name
   server MUST also place its RRSIG RRs in the Additional section.
   If space does not permit inclusion of both the RRset and its
   associated RRSIG RRs, the name server MAY retain the RRset while
   dropping the RRSIG RRs. If this happens, the name server MUST NOT
   set the TC bit solely because these RRSIG RRs didn't fit.

3.1.2. Including DNSKEY RRs in a Response

   When responding to a query that has the DO bit set and that requests
   the SOA or NS RRs at the apex of a signed zone, a security-aware
   authoritative name server for that zone MAY return the zone apex
   DNSKEY RRset in the Additional section. In this situation, the
   DNSKEY RRset and associated RRSIG RRs have lower priority than does
   any other information that would be placed in the additional section.
   The name server SHOULD NOT include the DNSKEY RRset unless there is
   enough space in the response message for both the DNSKEY RRset and
   its associated RRSIG RR(s). If there is not enough space to include
   these DNSKEY and RRSIG RRs, the name server MUST omit them and MUST
   NOT set the TC bit solely because these RRs didn't fit (see Section
   3.1.1).

3.1.3. Including NSEC RRs in a Response

   When responding to a query that has the DO bit set, a security-aware
   authoritative name server for a signed zone MUST include NSEC RRs in
   each of the following cases:

   No Data: The zone contains RRsets that exactly match <SNAME, SCLASS>
      but does not contain any RRsets that exactly match <SNAME, SCLASS,
      STYPE>.

   Name Error: The zone does not contain any RRsets that match <SNAME,
      SCLASS> either exactly or via wildcard name expansion.

   Wildcard Answer: The zone does not contain any RRsets that exactly
      match <SNAME, SCLASS> but does contain an RRset that matches
      <SNAME, SCLASS, STYPE> via wildcard name expansion.

   Wildcard No Data: The zone does not contain any RRsets that exactly
      match <SNAME, SCLASS> and does contain one or more RRsets that
      match <SNAME, SCLASS> via wildcard name expansion, but does not
      contain any RRsets that match <SNAME, SCLASS, STYPE> via wildcard
      name expansion.

   In each of these cases, the name server includes NSEC RRs in the
   response to prove that an exact match for <SNAME, SCLASS, STYPE> was
   not present in the zone and that the response that the name server is
   returning is correct given the data in the zone.

3.1.3.1. Including NSEC RRs: No Data Response

   If the zone contains RRsets matching <SNAME, SCLASS> but contains no
   RRset matching <SNAME, SCLASS, STYPE>, then the name server MUST
   include the NSEC RR for <SNAME, SCLASS> along with its associated
   RRSIG RR(s) in the Authority section of the response (see Section
   3.1.1). If space does not permit inclusion of the NSEC RR or its
   associated RRSIG RR(s), the name server MUST set the TC bit (see
   Section 3.1.1).

   Since the search name exists, wildcard name expansion does not apply
   to this query, and a single signed NSEC RR suffices to prove that the
   requested RR type does not exist.

3.1.3.2. Including NSEC RRs: Name Error Response

   If the zone does not contain any RRsets matching <SNAME, SCLASS>
   either exactly or via wildcard name expansion, then the name server
   MUST include the following NSEC RRs in the Authority section, along
   with their associated RRSIG RRs:

   o An NSEC RR proving that there is no exact match for <SNAME,
      SCLASS>.

   o An NSEC RR proving that the zone contains no RRsets that would
      match <SNAME, SCLASS> via wildcard name expansion.

   In some cases, a single NSEC RR may prove both of these points. If
   it does, the name server SHOULD only include the NSEC RR and its
   RRSIG RR(s) once in the Authority section.

   If space does not permit inclusion of these NSEC and RRSIG RRs, the
   name server MUST set the TC bit (see Section 3.1.1).

   The owner names of these NSEC and RRSIG RRs are not subject to
   wildcard name expansion when these RRs are included in the Authority
   section of the response.

   Note that this form of response includes cases in which SNAME
   corresponds to an empty non-terminal name within the zone (a name
   that is not the owner name for any RRset but that is the parent name
   of one or more RRsets).

3.1.3.3. Including NSEC RRs: Wildcard Answer Response

   If the zone does not contain any RRsets that exactly match <SNAME,
   SCLASS> but does contain an RRset that matches <SNAME, SCLASS, STYPE>
   via wildcard name expansion, the name server MUST include the

wildcard-expanded answer and the corresponding wildcard-expanded
RRSIG RRs in the Answer section and MUST include in the Authority
section an NSEC RR and associated RRSIG RR(s) proving that the zone
does not contain a closer match for <SNAME, SCLASS>. If space does
not permit inclusion of the answer, NSEC and RRSIG RRs, the name
server MUST set the TC bit (see Section 3.1.1).

3.1.3.4. Including NSEC RRs: Wildcard No Data Response

   This case is a combination of the previous cases. The zone does not
   contain an exact match for <SNAME, SCLASS>, and although the zone
   does contain RRsets that match <SNAME, SCLASS> via wildcard
   expansion, none of those RRsets matches STYPE. The name server MUST
   include the following NSEC RRs in the Authority section, along with
   their associated RRSIG RRs:

   o An NSEC RR proving that there are no RRsets matching STYPE at the
      wildcard owner name that matched <SNAME, SCLASS> via wildcard
      expansion.

   o An NSEC RR proving that there are no RRsets in the zone that would
      have been a closer match for <SNAME, SCLASS>.

   In some cases, a single NSEC RR may prove both of these points. If
   it does, the name server SHOULD only include the NSEC RR and its
   RRSIG RR(s) once in the Authority section.

   The owner names of these NSEC and RRSIG RRs are not subject to
   wildcard name expansion when these RRs are included in the Authority
   section of the response.

   If space does not permit inclusion of these NSEC and RRSIG RRs, the
   name server MUST set the TC bit (see Section 3.1.1).

3.1.3.5. Finding the Right NSEC RRs

   As explained above, there are several situations in which a
   security-aware authoritative name server has to locate an NSEC RR
   that proves that no RRsets matching a particular SNAME exist.
   Locating such an NSEC RR within an authoritative zone is relatively
   simple, at least in concept. The following discussion assumes that
   the name server is authoritative for the zone that would have held
   the non-existent RRsets matching SNAME. The algorithm below is
   written for clarity, not for efficiency.

   To find the NSEC that proves that no RRsets matching name N exist in
   the zone Z that would have held them, construct a sequence, S,
   consisting of the owner names of every RRset in Z, sorted into

   canonical order ([RFC4034]), with no duplicate names. Find the name
   M that would have immediately preceded N in S if any RRsets with
   owner name N had existed. M is the owner name of the NSEC RR that
   proves that no RRsets exist with owner name N.

   The algorithm for finding the NSEC RR that proves that a given name
   is not covered by any applicable wildcard is similar but requires an
   extra step. More precisely, the algorithm for finding the NSEC
   proving that no RRsets exist with the applicable wildcard name is
   precisely the same as the algorithm for finding the NSEC RR that
   proves that RRsets with any other owner name do not exist. The part
   that's missing is a method of determining the name of the non-
   existent applicable wildcard. In practice, this is easy, because the
   authoritative name server has already checked for the presence of
   precisely this wildcard name as part of step (1)(c) of the normal
   lookup algorithm described in Section 4.3.2 of [RFC1034].

3.1.4. Including DS RRs in a Response

   When responding to a query that has the DO bit set, a security-aware
   authoritative name server returning a referral includes DNSSEC data
   along with the NS RRset.

   If a DS RRset is present at the delegation point, the name server
   MUST return both the DS RRset and its associated RRSIG RR(s) in the
   Authority section along with the NS RRset.

   If no DS RRset is present at the delegation point, the name server
   MUST return both the NSEC RR that proves that the DS RRset is not
   present and the NSEC RR's associated RRSIG RR(s) along with the NS
   RRset. The name server MUST place the NS RRset before the NSEC RRset
   and its associated RRSIG RR(s).

   Including these DS, NSEC, and RRSIG RRs increases the size of
   referral messages and may cause some or all glue RRs to be omitted.
   If space does not permit inclusion of the DS or NSEC RRset and
   associated RRSIG RRs, the name server MUST set the TC bit (see
   Section 3.1.1).

3.1.4.1. Responding to Queries for DS RRs

   The DS resource record type is unusual in that it appears only on the
   parent zone's side of a zone cut. For example, the DS RRset for the
   delegation of "foo.example" is stored in the "example" zone rather
   than in the "foo.example" zone. This requires special processing
   rules for both name servers and resolvers, as the name server for the
   child zone is authoritative for the name at the zone cut by the
   normal DNS rules but the child zone does not contain the DS RRset.

A security-aware resolver sends queries to the parent zone when
looking for a needed DS RR at a delegation point (see Section 4.2).
However, special rules are necessary to avoid confusing
security-oblivious resolvers which might become involved in
processing such a query (for example, in a network configuration that
forces a security-aware resolver to channel its queries through a
security-oblivious recursive name server). The rest of this section
describes how a security-aware name server processes DS queries in
order to avoid this problem.

The need for special processing by a security-aware name server only
arises when all the following conditions are met:

o   The name server has received a query for the DS RRset at a zone
    cut.

o   The name server is authoritative for the child zone.

o   The name server is not authoritative for the parent zone.

o   The name server does not offer recursion.

In all other cases, the name server either has some way of obtaining
the DS RRset or could not have been expected to have the DS RRset
even by the pre-DNSSEC processing rules, so the name server can
return either the DS RRset or an error response according to the
normal processing rules.

If all the above conditions are met, however, the name server is
authoritative for SNAME but cannot supply the requested RRset. In
this case, the name server MUST return an authoritative "no data"
response showing that the DS RRset does not exist in the child zone's
apex. See Appendix B.8 for an example of such a response.

3.1.5. Responding to Queries for Type AXFR or IXFR

DNSSEC does not change the DNS zone transfer process. A signed zone
will contain RRSIG, DNSKEY, NSEC, and DS resource records, but these
records have no special meaning with respect to a zone transfer
operation.

An authoritative name server is not required to verify that a zone is
properly signed before sending or accepting a zone transfer.
However, an authoritative name server MAY choose to reject the entire
zone transfer if the zone fails to meet any of the signing
requirements described in Section 2. The primary objective of a zone
transfer is to ensure that all authoritative name servers have
identical copies of the zone. An authoritative name server that

chooses to perform its own zone validation MUST NOT selectively
reject some RRs and accept others.

DS RRsets appear only on the parental side of a zone cut and are
authoritative data in the parent zone. As with any other
authoritative RRset, the DS RRset MUST be included in zone transfers
of the zone in which the RRset is authoritative data. In the case of
the DS RRset, this is the parent zone.

NSEC RRs appear in both the parent and child zones at a zone cut and
are authoritative data in both the parent and child zones. The
parental and child NSEC RRs at a zone cut are never identical to each
other, as the NSEC RR in the child zone's apex will always indicate
the presence of the child zone's SOA RR whereas the parental NSEC RR
at the zone cut will never indicate the presence of an SOA RR. As
with any other authoritative RRs, NSEC RRs MUST be included in zone
transfers of the zone in which they are authoritative data. The
parental NSEC RR at a zone cut MUST be included in zone transfers of
the parent zone, and the NSEC at the zone apex of the child zone MUST
be included in zone transfers of the child zone.

RRSIG RRs appear in both the parent and child zones at a zone cut and
are authoritative in whichever zone contains the authoritative RRset
for which the RRSIG RR provides the signature. That is, the RRSIG RR
for a DS RRset or a parental NSEC RR at a zone cut will be
authoritative in the parent zone, and the RRSIG for any RRset in the
child zone's apex will be authoritative in the child zone. Parental
and child RRSIG RRs at a zone cut will never be identical to each
other, as the Signer's Name field of an RRSIG RR in the child zone's
apex will indicate a DNSKEY RR in the child zone's apex whereas the
same field of a parental RRSIG RR at the zone cut will indicate a
DNSKEY RR in the parent zone's apex. As with any other authoritative
RRs, RRSIG RRs MUST be included in zone transfers of the zone in
which they are authoritative data.

3.1.6. The AD and CD Bits in an Authoritative Response

The CD and AD bits are designed for use in communication between
security-aware resolvers and security-aware recursive name servers.
These bits are for the most part not relevant to query processing by
security-aware authoritative name servers.

A security-aware name server does not perform signature validation
for authoritative data during query processing, even when the CD bit
is clear. A security-aware name server SHOULD clear the CD bit when
composing an authoritative response.

   A security-aware name server MUST NOT set the AD bit in a response
   unless the name server considers all RRsets in the Answer and
   Authority sections of the response to be authentic. A security-aware
   name server's local policy MAY consider data from an authoritative
   zone to be authentic without further validation. However, the name
   server MUST NOT do so unless the name server obtained the
   authoritative zone via secure means (such as a secure zone transfer
   mechanism) and MUST NOT do so unless this behavior has been
   configured explicitly.

   A security-aware name server that supports recursion MUST follow the
   rules for the CD and AD bits given in Section 3.2 when generating a
   response that involves data obtained via recursion.

3.2. Recursive Name Servers

   As explained in [RFC4033], a security-aware recursive name server is
   an entity that acts in both the security-aware name server and
   security-aware resolver roles. This section uses the terms "name
   server side" and "resolver side" to refer to the code within a
   security-aware recursive name server that implements the
   security-aware name server role and the code that implements the
   security-aware resolver role, respectively.

   The resolver side follows the usual rules for caching and negative
   caching that would apply to any security-aware resolver.

3.2.1. The DO Bit

   The resolver side of a security-aware recursive name server MUST set
   the DO bit when sending requests, regardless of the state of the DO
   bit in the initiating request received by the name server side. If
   the DO bit in an initiating query is not set, the name server side
   MUST strip any authenticating DNSSEC RRs from the response but MUST
   NOT strip any DNSSEC RR types that the initiating query explicitly
   requested.

3.2.2. The CD Bit

   The CD bit exists in order to allow a security-aware resolver to
   disable signature validation in a security-aware name server's
   processing of a particular query.

   The name server side MUST copy the setting of the CD bit from a query
   to the corresponding response.

   The name server side of a security-aware recursive name server MUST
   pass the state of the CD bit to the resolver side along with the rest

of an initiating query, so that the resolver side will know whether
it is required to verify the response data it returns to the name
server side. If the CD bit is set, it indicates that the originating
resolver is willing to perform whatever authentication its local
policy requires. Thus, the resolver side of the recursive name
server need not perform authentication on the RRsets in the response.
When the CD bit is set, the recursive name server SHOULD, if
possible, return the requested data to the originating resolver, even
if the recursive name server's local authentication policy would
reject the records in question. That is, by setting the CD bit, the
originating resolver has indicated that it takes responsibility for
performing its own authentication, and the recursive name server
should not interfere.

If the resolver side implements a BAD cache (see Section 4.7) and the
name server side receives a query that matches an entry in the
resolver side's BAD cache, the name server side's response depends on
the state of the CD bit in the original query. If the CD bit is set,
the name server side SHOULD return the data from the BAD cache; if
the CD bit is not set, the name server side MUST return RCODE 2
(server failure).

The intent of the above rule is to provide the raw data to clients
that are capable of performing their own signature verification
checks while protecting clients that depend on the resolver side of a
security-aware recursive name server to perform such checks. Several
of the possible reasons why signature validation might fail involve
conditions that may not apply equally to the recursive name server
and the client that invoked it. For example, the recursive name
server's clock may be set incorrectly, or the client may have
knowledge of a relevant island of security that the recursive name
server does not share. In such cases, "protecting" a client that is
capable of performing its own signature validation from ever seeing
the "bad" data does not help the client.

3.2.3. The AD Bit

The name server side of a security-aware recursive name server MUST
NOT set the AD bit in a response unless the name server considers all
RRsets in the Answer and Authority sections of the response to be
authentic. The name server side SHOULD set the AD bit if and only if
the resolver side considers all RRsets in the Answer section and any
relevant negative response RRs in the Authority section to be
authentic. The resolver side MUST follow the procedure described in
Section 5 to determine whether the RRs in question are authentic.
However, for backward compatibility, a recursive name server MAY set
the AD bit when a response includes unsigned CNAME RRs if those CNAME

RRs demonstrably could have been synthesized from an authentic DNAME
RR that is also included in the response according to the synthesis
rules described in [RFC2672].

3.3. Example DNSSEC Responses

See Appendix B for example response packets.

4. Resolving

This section describes the behavior of entities that include
security-aware resolver functions. In many cases such functions will
be part of a security-aware recursive name server, but a stand-alone
security-aware resolver has many of the same requirements. Functions
specific to security-aware recursive name servers are described in
Section 3.2.

4.1. EDNS Support

A security-aware resolver MUST include an EDNS ([RFC2671]) OPT
pseudo-RR with the DO ([RFC3225]) bit set when sending queries.

A security-aware resolver MUST support a message size of at least
1220 octets, SHOULD support a message size of 4000 octets, and MUST
use the "sender's UDP payload size" field in the EDNS OPT pseudo-RR
to advertise the message size that it is willing to accept. A
security-aware resolver's IP layer MUST handle fragmented UDP packets
correctly regardless of whether any such fragmented packets were
received via IPv4 or IPv6. Please see [RFC1122], [RFC2460], and
[RFC3226] for discussion of these requirements.

4.2. Signature Verification Support

A security-aware resolver MUST support the signature verification
mechanisms described in Section 5 and SHOULD apply them to every
received response, except when:

o the security-aware resolver is part of a security-aware recursive
   name server, and the response is the result of recursion on behalf
   of a query received with the CD bit set;

o the response is the result of a query generated directly via some
   form of application interface that instructed the security-aware
   resolver not to perform validation for this query; or

o validation for this query has been disabled by local policy.

A security-aware resolver's support for signature verification MUST
include support for verification of wildcard owner names.

Security-aware resolvers MAY query for missing security RRs in an
attempt to perform validation; implementations that choose to do so
must be aware that the answers received may not be sufficient to
validate the original response. For example, a zone update may have
changed (or deleted) the desired information between the original and
follow-up queries.

When attempting to retrieve missing NSEC RRs that reside on the
parental side at a zone cut, a security-aware iterative-mode resolver
MUST query the name servers for the parent zone, not the child zone.

When attempting to retrieve a missing DS, a security-aware
iterative-mode resolver MUST query the name servers for the parent
zone, not the child zone. As explained in Section 3.1.4.1,
security-aware name servers need to apply special processing rules to
handle the DS RR, and in some situations the resolver may also need
to apply special rules to locate the name servers for the parent zone
if the resolver does not already have the parent's NS RRset. To
locate the parent NS RRset, the resolver can start with the
delegation name, strip off the leftmost label, and query for an NS
RRset by that name. If no NS RRset is present at that name, the
resolver then strips off the leftmost remaining label and retries the
query for that name, repeating this process of walking up the tree
until it either finds the NS RRset or runs out of labels.

4.3. Determining Security Status of Data

A security-aware resolver MUST be able to determine whether it should
expect a particular RRset to be signed. More precisely, a
security-aware resolver must be able to distinguish between four
cases:

Secure: An RRset for which the resolver is able to build a chain of
   signed DNSKEY and DS RRs from a trusted security anchor to the
   RRset. In this case, the RRset should be signed and is subject to
   signature validation, as described above.

Insecure: An RRset for which the resolver knows that it has no chain
   of signed DNSKEY and DS RRs from any trusted starting point to the
   RRset. This can occur when the target RRset lies in an unsigned
   zone or in a descendent of an unsigned zone. In this case, the
   RRset may or may not be signed, but the resolver will not be able
   to verify the signature.

Bogus: An RRset for which the resolver believes that it ought to be
   able to establish a chain of trust but for which it is unable to
   do so, either due to signatures that for some reason fail to
   validate or due to missing data that the relevant DNSSEC RRs
   indicate should be present. This case may indicate an attack but
   may also indicate a configuration error or some form of data
   corruption.

Indeterminate: An RRset for which the resolver is not able to
   determine whether the RRset should be signed, as the resolver is
   not able to obtain the necessary DNSSEC RRs. This can occur when
   the security-aware resolver is not able to contact security-aware
   name servers for the relevant zones.

4.4. Configured Trust Anchors

   A security-aware resolver MUST be capable of being configured with at
   least one trusted public key or DS RR and SHOULD be capable of being
   configured with multiple trusted public keys or DS RRs. Since a
   security-aware resolver will not be able to validate signatures
   without such a configured trust anchor, the resolver SHOULD have some
   reasonably robust mechanism for obtaining such keys when it boots;
   examples of such a mechanism would be some form of non-volatile
   storage (such as a disk drive) or some form of trusted local network
   configuration mechanism.

   Note that trust anchors also cover key material that is updated in a
   secure manner. This secure manner could be through physical media, a
   key exchange protocol, or some other out-of-band means.

4.5. Response Caching

   A security-aware resolver SHOULD cache each response as a single
   atomic entry containing the entire answer, including the named RRset
   and any associated DNSSEC RRs. The resolver SHOULD discard the
   entire atomic entry when any of the RRs contained in it expire. In
   most cases the appropriate cache index for the atomic entry will be
   the triple <QNAME, QTYPE, QCLASS>, but in cases such as the response
   form described in Section 3.1.3.2 the appropriate cache index will be
   the double <QNAME,QCLASS>.

   The reason for these recommendations is that, between the initial
   query and the expiration of the data from the cache, the
   authoritative data might have been changed (for example, via dynamic
   update).

There are two situations for which this is relevant:

1. By using the RRSIG record, it is possible to deduce that an
    answer was synthesized from a wildcard. A security-aware
    recursive name server could store this wildcard data and use it
    to generate positive responses to queries other than the name for
    which the original answer was first received.

2. NSEC RRs received to prove the non-existence of a name could be
    reused by a security-aware resolver to prove the non-existence of
    any name in the name range it spans.

In theory, a resolver could use wildcards or NSEC RRs to generate
positive and negative responses (respectively) until the TTL or
signatures on the records in question expire. However, it seems
prudent for resolvers to avoid blocking new authoritative data or
synthesizing new data on their own. Resolvers that follow this
recommendation will have a more consistent view of the namespace.

4.6. Handling of the CD and AD Bits

A security-aware resolver MAY set a query's CD bit in order to
indicate that the resolver takes responsibility for performing
whatever authentication its local policy requires on the RRsets in
the response. See Section 3.2 for the effect this bit has on the
behavior of security-aware recursive name servers.

A security-aware resolver MUST clear the AD bit when composing query
messages to protect against buggy name servers that blindly copy
header bits that they do not understand from the query message to the
response message.

A resolver MUST disregard the meaning of the CD and AD bits in a
response unless the response was obtained by using a secure channel
or the resolver was specifically configured to regard the message
header bits without using a secure channel.

4.7. Caching BAD Data

While many validation errors will be transient, some are likely to be
more persistent, such as those caused by administrative error
(failure to re-sign a zone, clock skew, and so forth). Since
requerying will not help in these cases, validating resolvers might
generate a significant amount of unnecessary DNS traffic as a result
of repeated queries for RRsets with persistent validation failures.

To prevent such unnecessary DNS traffic, security-aware resolvers MAY
cache data with invalid signatures, with some restrictions.

Conceptually, caching such data is similar to negative caching
([RFC2308]), except that instead of caching a valid negative
response, the resolver is caching the fact that a particular answer
failed to validate. This document refers to a cache of data with
invalid signatures as a "BAD cache".

Resolvers that implement a BAD cache MUST take steps to prevent the
cache from being useful as a denial-of-service attack amplifier,
particularly the following:

o Since RRsets that fail to validate do not have trustworthy TTLs,
   the implementation MUST assign a TTL. This TTL SHOULD be small,
   in order to mitigate the effect of caching the results of an
   attack.

o In order to prevent caching of a transient validation failure
   (which might be the result of an attack), resolvers SHOULD track
   queries that result in validation failures and SHOULD only answer
   from the BAD cache after the number of times that responses to
   queries for that particular <QNAME, QTYPE, QCLASS> have failed to
   validate exceeds a threshold value.

Resolvers MUST NOT return RRsets from the BAD cache unless the
resolver is not required to validate the signatures of the RRsets in
question under the rules given in Section 4.2 of this document. See
Section 3.2.2 for discussion of how the responses returned by a
security-aware recursive name server interact with a BAD cache.

4.8. Synthesized CNAMEs

A validating security-aware resolver MUST treat the signature of a
valid signed DNAME RR as also covering unsigned CNAME RRs that could
have been synthesized from the DNAME RR, as described in [RFC2672],
at least to the extent of not rejecting a response message solely
because it contains such CNAME RRs. The resolver MAY retain such
CNAME RRs in its cache or in the answers it hands back, but is not
required to do so.

4.9. Stub Resolvers

A security-aware stub resolver MUST support the DNSSEC RR types, at
least to the extent of not mishandling responses just because they
contain DNSSEC RRs.

4.9.1. Handling of the DO Bit

   A non-validating security-aware stub resolver MAY include the DNSSEC
   RRs returned by a security-aware recursive name server as part of the
   data that the stub resolver hands back to the application that
   invoked it, but is not required to do so. A non-validating stub
   resolver that seeks to do this will need to set the DO bit in order
   to receive DNSSEC RRs from the recursive name server.

   A validating security-aware stub resolver MUST set the DO bit,
   because otherwise it will not receive the DNSSEC RRs it needs to
   perform signature validation.

4.9.2. Handling of the CD Bit

   A non-validating security-aware stub resolver SHOULD NOT set the CD
   bit when sending queries unless it is requested by the application
   layer, as by definition, a non-validating stub resolver depends on
   the security-aware recursive name server to perform validation on its
   behalf.

   A validating security-aware stub resolver SHOULD set the CD bit,
   because otherwise the security-aware recursive name server will
   answer the query using the name server's local policy, which may
   prevent the stub resolver from receiving data that would be
   acceptable to the stub resolver's local policy.

4.9.3. Handling of the AD Bit

   A non-validating security-aware stub resolver MAY chose to examine
   the setting of the AD bit in response messages that it receives in
   order to determine whether the security-aware recursive name server
   that sent the response claims to have cryptographically verified the
   data in the Answer and Authority sections of the response message.
   Note, however, that the responses received by a security-aware stub
   resolver are heavily dependent on the local policy of the
   security-aware recursive name server. Therefore, there may be little
   practical value in checking the status of the AD bit, except perhaps
   as a debugging aid. In any case, a security-aware stub resolver MUST
   NOT place any reliance on signature validation allegedly performed on
   its behalf, except when the security-aware stub resolver obtained the
   data in question from a trusted security-aware recursive name server
   via a secure channel.

   A validating security-aware stub resolver SHOULD NOT examine the
   setting of the AD bit in response messages, as, by definition, the
   stub resolver performs its own signature validation regardless of the
   setting of the AD bit.

5. Authenticating DNS Responses

   To use DNSSEC RRs for authentication, a security-aware resolver
   requires configured knowledge of at least one authenticated DNSKEY or
   DS RR. The process for obtaining and authenticating this initial
   trust anchor is achieved via some external mechanism. For example, a
   resolver could use some off-line authenticated exchange to obtain a
   zone's DNSKEY RR or to obtain a DS RR that identifies and
   authenticates a zone's DNSKEY RR. The remainder of this section
   assumes that the resolver has somehow obtained an initial set of
   trust anchors.

   An initial DNSKEY RR can be used to authenticate a zone's apex DNSKEY
   RRset. To authenticate an apex DNSKEY RRset by using an initial key,
   the resolver MUST:

   1. verify that the initial DNSKEY RR appears in the apex DNSKEY
       RRset, and that the DNSKEY RR has the Zone Key Flag (DNSKEY RDATA
       bit 7) set; and

   2. verify that there is some RRSIG RR that covers the apex DNSKEY
       RRset, and that the combination of the RRSIG RR and the initial
       DNSKEY RR authenticates the DNSKEY RRset. The process for using
       an RRSIG RR to authenticate an RRset is described in Section 5.3.

   Once the resolver has authenticated the apex DNSKEY RRset by using an
   initial DNSKEY RR, delegations from that zone can be authenticated by
   using DS RRs. This allows a resolver to start from an initial key
   and use DS RRsets to proceed recursively down the DNS tree, obtaining
   other apex DNSKEY RRsets. If the resolver were configured with a
   root DNSKEY RR, and if every delegation had a DS RR associated with
   it, then the resolver could obtain and validate any apex DNSKEY
   RRset. The process of using DS RRs to authenticate referrals is
   described in Section 5.2.

   Section 5.3 shows how the resolver can use DNSKEY RRs in the apex
   DNSKEY RRset and RRSIG RRs from the zone to authenticate any other
   RRsets in the zone once the resolver has authenticated a zone's apex
   DNSKEY RRset. Section 5.4 shows how the resolver can use
   authenticated NSEC RRsets from the zone to prove that an RRset is not
   present in the zone.

   When a resolver indicates support for DNSSEC (by setting the DO bit),
   a security-aware name server should attempt to provide the necessary
   DNSKEY, RRSIG, NSEC, and DS RRsets in a response (see Section 3).
   However, a security-aware resolver may still receive a response that
   lacks the appropriate DNSSEC RRs, whether due to configuration issues
   such as an upstream security-oblivious recursive name server that

accidentally interferes with DNSSEC RRs or due to a deliberate attack
in which an adversary forges a response, strips DNSSEC RRs from a
response, or modifies a query so that DNSSEC RRs appear not to be
requested. The absence of DNSSEC data in a response MUST NOT by
itself be taken as an indication that no authentication information
exists.

A resolver SHOULD expect authentication information from signed
zones. A resolver SHOULD believe that a zone is signed if the
resolver has been configured with public key information for the
zone, or if the zone's parent is signed and the delegation from the
parent contains a DS RRset.

5.1. Special Considerations for Islands of Security

Islands of security (see [RFC4033]) are signed zones for which it is
not possible to construct an authentication chain to the zone from
its parent. Validating signatures within an island of security
requires that the validator have some other means of obtaining an
initial authenticated zone key for the island. If a validator cannot
obtain such a key, it SHOULD switch to operating as if the zones in
the island of security are unsigned.

All the normal processes for validating responses apply to islands of
security. The only difference between normal validation and
validation within an island of security is in how the validator
obtains a trust anchor for the authentication chain.

5.2. Authenticating Referrals

Once the apex DNSKEY RRset for a signed parent zone has been
authenticated, DS RRsets can be used to authenticate the delegation
to a signed child zone. A DS RR identifies a DNSKEY RR in the child
zone's apex DNSKEY RRset and contains a cryptographic digest of the
child zone's DNSKEY RR. Use of a strong cryptographic digest
algorithm ensures that it is computationally infeasible for an
adversary to generate a DNSKEY RR that matches the digest. Thus,
authenticating the digest allows a resolver to authenticate the
matching DNSKEY RR. The resolver can then use this child DNSKEY RR
to authenticate the entire child apex DNSKEY RRset.

Given a DS RR for a delegation, the child zone's apex DNSKEY RRset
can be authenticated if all of the following hold:

o The DS RR has been authenticated using some DNSKEY RR in the
   parent's apex DNSKEY RRset (see Section 5.3).

   o The Algorithm and Key Tag in the DS RR match the Algorithm field
      and the key tag of a DNSKEY RR in the child zone's apex DNSKEY
      RRset, and, when the DNSKEY RR's owner name and RDATA are hashed
      using the digest algorithm specified in the DS RR's Digest Type
      field, the resulting digest value matches the Digest field of the
      DS RR.

   o The matching DNSKEY RR in the child zone has the Zone Flag bit
      set, the corresponding private key has signed the child zone's
      apex DNSKEY RRset, and the resulting RRSIG RR authenticates the
      child zone's apex DNSKEY RRset.

   If the referral from the parent zone did not contain a DS RRset, the
   response should have included a signed NSEC RRset proving that no DS
   RRset exists for the delegated name (see Section 3.1.4). A
   security-aware resolver MUST query the name servers for the parent
   zone for the DS RRset if the referral includes neither a DS RRset nor
   a NSEC RRset proving that the DS RRset does not exist (see Section
   4).

   If the validator authenticates an NSEC RRset that proves that no DS
   RRset is present for this zone, then there is no authentication path
   leading from the parent to the child. If the resolver has an initial
   DNSKEY or DS RR that belongs to the child zone or to any delegation
   below the child zone, this initial DNSKEY or DS RR MAY be used to
   re-establish an authentication path. If no such initial DNSKEY or DS
   RR exists, the validator cannot authenticate RRsets in or below the
   child zone.

   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.

   Note that, for a signed delegation, there are two NSEC RRs associated
   with the delegated name. One NSEC RR resides in the parent zone and
   can be used to prove whether a DS RRset exists for the delegated
   name. The second NSEC RR resides in the child zone and identifies
   which RRsets are present at the apex of the child zone. The parent
   NSEC RR and child NSEC RR can always be distinguished because the SOA
   bit will be set in the child NSEC RR and clear in the parent NSEC RR.
   A security-aware resolver MUST use the parent NSEC RR when attempting
   to prove that a DS RRset does not exist.

   If the resolver does not support any of the algorithms listed in an
   authenticated DS RRset, then the resolver will not be able to verify
   the authentication path to the child zone. In this case, the
   resolver SHOULD treat the child zone as if it were unsigned.

5.3. Authenticating an RRset with an RRSIG RR

   A validator can use an RRSIG RR and its corresponding DNSKEY RR to
   attempt to authenticate RRsets. The validator first checks the RRSIG
   RR to verify that it covers the RRset, has a valid time interval, and
   identifies a valid DNSKEY RR. The validator then constructs the
   canonical form of the signed data by appending the RRSIG RDATA
   (excluding the Signature Field) with the canonical form of the
   covered RRset. Finally, the validator uses the public key and
   signature to authenticate the signed data. Sections 5.3.1, 5.3.2,
   and 5.3.3 describe each step in detail.

5.3.1. Checking the RRSIG RR Validity

   A security-aware resolver can use an RRSIG RR to authenticate an
   RRset if all of the following conditions hold:

   o The RRSIG RR and the RRset MUST have the same owner name and the
      same class.

   o The RRSIG RR's Signer's Name field MUST be the name of the zone
      that contains the RRset.

   o The RRSIG RR's Type Covered field MUST equal the RRset's type.

   o The number of labels in the RRset owner name MUST be greater than
      or equal to the value in the RRSIG RR's Labels field.

   o The validator's notion of the current time MUST be less than or
      equal to the time listed in the RRSIG RR's Expiration field.

   o The validator's notion of the current time MUST be greater than or
      equal to the time listed in the RRSIG RR's Inception field.

   o The RRSIG RR's Signer's Name, Algorithm, and Key Tag fields MUST
      match the owner name, algorithm, and key tag for some DNSKEY RR in
      the zone's apex DNSKEY RRset.

   o The matching DNSKEY RR MUST be present in the zone's apex DNSKEY
      RRset, and MUST have the Zone Flag bit (DNSKEY RDATA Flag bit 7)
      set.

It is possible for more than one DNSKEY RR to match the conditions
above. In this case, the validator cannot predetermine which DNSKEY
RR to use to authenticate the signature, and it MUST try each
matching DNSKEY RR until either the signature is validated or the
validator has run out of matching public keys to try.

Note that this authentication process is only meaningful if the
validator authenticates the DNSKEY RR before using it to validate
signatures. The matching DNSKEY RR is considered to be authentic if:

o the apex DNSKEY RRset containing the DNSKEY RR is considered
   authentic; or

o the RRset covered by the RRSIG RR is the apex DNSKEY RRset itself,
   and the DNSKEY RR either matches an authenticated DS RR from the
   parent zone or matches a trust anchor.

5.3.2. Reconstructing the Signed Data

Once the RRSIG RR has met the validity requirements described in
Section 5.3.1, the validator has to reconstruct the original signed
data. The original signed data includes RRSIG RDATA (excluding the
Signature field) and the canonical form of the RRset. Aside from
being ordered, the canonical form of the RRset might also differ from
the received RRset due to DNS name compression, decremented TTLs, or
wildcard expansion. The validator should use the following to
reconstruct the original signed data:

         signed_data = RRSIG_RDATA | RR(1) | RR(2)... where

            "|" denotes concatenation

            RRSIG_RDATA is the wire format of the RRSIG RDATA fields
               with the Signature field excluded and the Signer's Name
               in canonical form.

            RR(i) = name | type | class | OrigTTL | RDATA length | RDATA

               name is calculated according to the function below

               class is the RRset's class

               type is the RRset type and all RRs in the class

               OrigTTL is the value from the RRSIG Original TTL field

               All names in the RDATA field are in canonical form

          The set of all RR(i) is sorted into canonical order.

       To calculate the name:
          let rrsig_labels = the value of the RRSIG Labels field

          let fqdn = RRset's fully qualified domain name in
                        canonical form

          let fqdn_labels = Label count of the fqdn above.

          if rrsig_labels = fqdn_labels,
             name = fqdn

          if rrsig_labels < fqdn_labels,
             name = "*." | the rightmost rrsig_label labels of the
                        fqdn

          if rrsig_labels > fqdn_labels
             the RRSIG RR did not pass the necessary validation
             checks and MUST NOT be used to authenticate this
             RRset.

   The canonical forms for names and RRsets are defined in [RFC4034].

   NSEC RRsets at a delegation boundary require special processing.
   There are two distinct NSEC RRsets associated with a signed delegated
   name. One NSEC RRset resides in the parent zone, and specifies which
   RRsets are present at the parent zone. The second NSEC RRset resides
   at the child zone and identifies which RRsets are present at the apex
   in the child zone. The parent NSEC RRset and child NSEC RRset can
   always be distinguished as only a child NSEC RR will indicate that an
   SOA RRset exists at the name. When reconstructing the original NSEC
   RRset for the delegation from the parent zone, the NSEC RRs MUST NOT
   be combined with NSEC RRs from the child zone. When reconstructing
   the original NSEC RRset for the apex of the child zone, the NSEC RRs
   MUST NOT be combined with NSEC RRs from the parent zone.

   Note that each of the two NSEC RRsets at a delegation point has a
   corresponding RRSIG RR with an owner name matching the delegated
   name, and each of these RRSIG RRs is authoritative data associated
   with the same zone that contains the corresponding NSEC RRset. If
   necessary, a resolver can tell these RRSIG RRs apart by checking the
   Signer's Name field.

5.3.3. Checking the Signature

   Once the resolver has validated the RRSIG RR as described in Section
   5.3.1 and reconstructed the original signed data as described in
   Section 5.3.2, the validator can attempt to use the cryptographic
   signature to authenticate the signed data, and thus (finally!)
   authenticate the RRset.

   The Algorithm field in the RRSIG RR identifies the cryptographic
   algorithm used to generate the signature. The signature itself is
   contained in the Signature field of the RRSIG RDATA, and the public
   key used to verify the signature is contained in the Public Key field
   of the matching DNSKEY RR(s) (found in Section 5.3.1). [RFC4034]
   provides a list of algorithm types and provides pointers to the
   documents that define each algorithm's use.

   Note that it is possible for more than one DNSKEY RR to match the
   conditions in Section 5.3.1. In this case, the validator can only
   determine which DNSKEY RR is correct by trying each matching public
   key until the validator either succeeds in validating the signature
   or runs out of keys to try.

   If the Labels field of the RRSIG RR is not equal to the number of
   labels in the RRset's fully qualified owner name, then the RRset is
   either invalid or the result of wildcard expansion. The resolver
   MUST verify that wildcard expansion was applied properly before
   considering the RRset to be authentic. Section 5.3.4 describes how
   to determine whether a wildcard was applied properly.

   If other RRSIG RRs also cover this RRset, 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.

   If the resolver accepts the RRset as authentic, the validator MUST
   set the TTL of the RRSIG RR and each RR in the authenticated RRset to
   a value no greater than the minimum of:

   o the RRset's TTL as received in the response;

   o the RRSIG RR's TTL as received in the response;

   o the value in the RRSIG RR's Original TTL field; and

   o the difference of the RRSIG RR's Signature Expiration time and the
      current time.

5.3.4. Authenticating a Wildcard Expanded RRset Positive Response

   If the number of labels in an RRset's owner name is greater than the
   Labels field of the covering RRSIG RR, then the RRset and its
   covering RRSIG RR were created as a result of wildcard expansion.
   Once the validator has verified the signature, as described in
   Section 5.3, it must take additional steps to verify the non-
   existence of an exact match or closer wildcard match for the query.
   Section 5.4 discusses these steps.

   Note that the response received by the resolver should include all
   NSEC RRs needed to authenticate the response (see Section 3.1.3).

5.4. Authenticated Denial of Existence

   A resolver can use authenticated NSEC RRs to prove that an RRset is
   not present in a signed zone. Security-aware name servers should
   automatically include any necessary NSEC RRs for signed zones in
   their responses to security-aware resolvers.

   Denial of existence is determined by the following rules:

   o If the requested RR name matches the owner name of an
      authenticated NSEC RR, then the NSEC RR's type bit map field lists
      all RR types present at that owner name, and a resolver can prove
      that the requested RR type does not exist by checking for the RR
      type in the bit map. If the number of labels in an authenticated
      NSEC RR's owner name equals the Labels field of the covering RRSIG
      RR, then the existence of the NSEC RR proves that wildcard
      expansion could not have been used to match the request.

   o If the requested RR name would appear after an authenticated NSEC
      RR's owner name and before the name listed in that NSEC RR's Next
      Domain Name field according to the canonical DNS name order
      defined in [RFC4034], then no RRsets with the requested name exist
      in the zone. However, it is possible that a wildcard could be
      used to match the requested RR owner name and type, so proving
      that the requested RRset does not exist also requires proving that
      no possible wildcard RRset exists that could have been used to
      generate a positive response.

   In addition, security-aware resolvers MUST authenticate the NSEC
   RRsets that comprise the non-existence proof as described in Section
   5.3.

   To prove the non-existence of an RRset, the resolver must be able to
   verify both that the queried RRset does not exist and that no
   relevant wildcard RRset exists. Proving this may require more than

one NSEC RRset from the zone. If the complete set of necessary NSEC
RRsets is not present in a response (perhaps due to message
truncation), then a security-aware resolver MUST resend the query in
order to attempt to obtain the full collection of NSEC RRs necessary
to verify the non-existence of the requested RRset. As with all DNS
operations, however, the resolver MUST bound the work it puts into
answering any particular query.

Since a validated NSEC RR proves the existence of both itself and its
corresponding RRSIG RR, a validator MUST ignore the settings of the
NSEC and RRSIG bits in an NSEC RR.

5.5. Resolver Behavior When Signatures Do Not Validate

If for whatever reason none of the RRSIGs can be validated, the
response SHOULD be considered BAD. If the validation was being done
to service a recursive query, the name server MUST return RCODE 2 to
the originating client. However, it MUST return the full response if
and only if the original query had the CD bit set. Also see Section
4.7 on caching responses that do not validate.

5.6. Authentication Example

Appendix C shows an example of the authentication process.

6. IANA Considerations

[RFC4034] contains a review of the IANA considerations introduced by
DNSSEC. The following are additional IANA considerations discussed
in this document:

[RFC2535] reserved the CD and AD bits in the message header. The
meaning of the AD bit was redefined in [RFC3655], and the meaning of
both the CD and AD bit are restated in this document. No new bits in
the DNS message header are defined in this document.

[RFC2671] introduced EDNS, and [RFC3225] reserved the DNSSEC OK bit
and defined its use. The use is restated but not altered in this
document.

7. Security Considerations

This document describes how the DNS security extensions use public
key cryptography to sign and authenticate DNS resource record sets.
Please see [RFC4033] for terminology and general security
considerations related to DNSSEC; see [RFC4034] for considerations
specific to the DNSSEC resource record types.

An active attacker who can set the CD bit in a DNS query message or
the AD bit in a DNS response message can use these bits to defeat the
protection that DNSSEC attempts to provide to security-oblivious
recursive-mode resolvers. For this reason, use of these control bits
by a security-aware recursive-mode resolver requires a secure
channel. See Sections 3.2.2 and 4.9 for further discussion.

The protocol described in this document attempts to extend the
benefits of DNSSEC to security-oblivious stub resolvers. However, as
recovery from validation failures is likely to be specific to
particular applications, the facilities that DNSSEC provides for stub
resolvers may prove inadequate. Operators of security-aware
recursive name servers will have to pay close attention to the
behavior of the applications that use their services when choosing a
local validation policy; failure to do so could easily result in the
recursive name server accidentally denying service to the clients it
is intended to support.

8. Acknowledgements

This document was created from the input and ideas of the members of
the DNS Extensions Working Group and working group mailing list. The
editors would like to express their thanks for the comments and
suggestions received during the revision of these security extension
specifications. Although explicitly listing everyone who has
contributed during the decade in which DNSSEC has been under
development would be impossible, [RFC4033] includes a list of some of
the participants who were kind enough to comment on these documents.

9. References

9.1. Normative References

   [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.

   [RFC1122] Braden, R., "Requirements for Internet Hosts -
              Communication Layers", STD 3, RFC 1122, October 1989.

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

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

   [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6
              (IPv6) Specification", RFC 2460, December 1998.

   [RFC2671] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC
              2671, August 1999.

   [RFC2672] Crawford, M., "Non-Terminal DNS Name Redirection", RFC
              2672, August 1999.

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

   [RFC3226] Gudmundsson, O., "DNSSEC and IPv6 A6 aware server/resolver
              message size requirements", RFC 3226, 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 DNS Security Extensions", RFC
              4034, March 2005.

9.2. Informative References

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

   [RFC2535] Eastlake 3rd, D., "Domain Name System Security
              Extensions", RFC 2535, March 1999.

   [RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic
              Update", RFC 3007, November 2000.

   [RFC3655] Wellington, B. and O. Gudmundsson, "Redefinition of DNS
              Authenticated Data (AD) bit", RFC 3655, November 2003.

Appendix A. Signed Zone Example

   The following example shows a (small) complete signed zone.

```
   example.         3600 IN SOA ns1.example. bugs.x.w.example. (
                                 1081539377
                                 3600
                                 300
                                 3600000
                                 3600
                                 )
                    3600 RRSIG  SOA 5 1 3600 20040509183619 (
                                 20040409183619 38519 example.
                                 ONx0k36rcjaxYtcNgq6iQnpNV5+drqYAsC9h
                                 7TSJaHCqbhE67Sr6aH2xDUGcqQWu/n0UVzrF
                                 vkgO9ebarZ0GWDKcuwlM6eNB5SiX2K74l5LW
                                 DA7S/Un/IbtDq4Ay8NMNLQI7Dw7n4p8/rjkB
                                 jV7j86HyQgM5e7+miRAz8V01b0I= )
                    3600 NS     ns1.example.
                    3600 NS     ns2.example.
                    3600 RRSIG  NS 5 1 3600 20040509183619 (
                                 20040409183619 38519 example.
                                 gl13F00f2U0R+SWiXXLHwsMY+qStYy5k6zfd
                                 EuivWc+wd1fmbNCyql0Tk7lHTX6UOxc8AgNf
                                 4ISFve8XqF4q+o9qlnqIzmppU3LiNeKT4FZ8
                                 RO5urFOvoMRTbQxW3U0hXWuggE4g3ZpsHv48
                                 0HjMeRaZB/FRPGfJPajngcq6Kwg= )
                    3600 MX     1 xx.example.
                    3600 RRSIG  MX 5 1 3600 20040509183619 (
                                 20040409183619 38519 example.
                                 HyDHYVT5KHSZ7HtO/vypumPmSZQrcOP3tzWB
                                 2qaKkHVPfau/DgLgS/IKENkYOGL95G4N+NzE
                                 VyNU8dcTOckT+ChPcGeVjguQ7a3Ao9Z/ZkUO
                                 6gmmUW4b89rz1PUxW4jzUxj66PTwoVtUU/iM
                                 W6OISukd1EQt7a0kygkg+PEDxdI= )
                    3600 NSEC   a.example. NS SOA MX RRSIG NSEC DNSKEY
                    3600 RRSIG  NSEC 5 1 3600 20040509183619 (
                                 20040409183619 38519 example.
                                 O0k558jHhyrC97ISHnislm4kLMW48C7U7cBm
                                 FTfhke5iVqNRVTB1STLMpgpbDIC9hcryoO0V
                                 Z9ME5xPzUEhbvGnHd5sfzgFVeGxr5Nyyq4tW
                                 SDBgIBiLQUv1ivy29vhXy7WgR62dPrZ0PWvm
                                 jfFJ5arXf4nPxp/kEowGgBRzY/U= )
                    3600 DNSKEY 256 3 5 (
                                 AQOy1bZVvpPqhg4j7EJoM9rI3ZmyEx2OzDBV
                                 rZy/lvI5CQePxXHZS4i8dANH4DX3tbHol61e
                                 k8EFMcsGXxKciJFHyhl94C+NwILQdzsUlSFo
                                 vBZsyl/NX6yEbtw/xN9ZNcrbYvgjjZ/UVPZI
```

```
                             ySFNsgEYvh0z2542lzMKR4Dh8uZffQ==
                             )
             3600 DNSKEY 257 3 5 (
                             AQOeX7+baTmvpVHb2CcLnL1dMRWbuscRvHXl
                             LnXwDzvqp4tZVKp1sZMepFb8MvxhhW3y/0QZ
                             syCjczGJ1qk8vJe52iOhInKROVLRwxGpMfzP
                             RLMlGybr51bOV/1se0ODacj3DomyB4QB5gKT
                             Yot/K9alk5/j8vfd4jWCWD+E1Sze0Q==
                             )
             3600 RRSIG  DNSKEY 5 1 3600 20040509183619 (
                             20040409183619 9465 example.
                             ZxgauAuIj+k1YoVEOSlZfx41fcmKzTFHoweZ
                             xYnz99JVQZJ33wFS0Q0jcP7VXKkaElXk9nYJ
                             XevO/7nAbo88iWsMkSpSR6jWzYYKwfrBI/L9
                             hjYmyVO9m6FjQ7uwM4dCP/bIuV/DKqOAK9NY
                             NC3AHfvCV1Tp4VKDqxqG7R5tTVM= )
             3600 RRSIG  DNSKEY 5 1 3600 20040509183619 (
                             20040409183619 38519 example.
                             eGL0s90glUqcOmloo/2y+bSzyEfKVOQViD9Z
                             DNhLz/Yn9CQZlDVRJffACQDAUhXpU/oP34ri
                             bKBpysRXosczFrKqS5Oa0bzMOfXCXup9qHAp
                             eFIku28Vqfr8Nt7cigZLxjK+u0Ws/4lIRjKk
                             7z5OXogYVaFzHKillDt3HRxHIZM= )
a.example.   3600 IN NS  ns1.a.example.
             3600 IN NS  ns2.a.example.
             3600 DS     57855 5 1 (
                             B6DCD485719ADCA18E5F3D48A2331627FDD3
                             636B )
             3600 RRSIG  DS 5 2 3600 20040509183619 (
                             20040409183619 38519 example.
                             oXIKit/QtdG64J/CB+Gi8dOvnwRvqrto1AdQ
                             oRkAN15FP3iZ7suB7gvTBmXzCjL7XUgQVcoH
                             kdhyCuzp8W9qJHgRUSwKKkczSyuL64nhgjuD
                             EML8l9wlWVsl7PR2VnZduM9bLyBhaaPmRKX/
                             Fm+v6ccF2EGNLRiY08kdkz+XHHo= )
             3600 NSEC   ai.example. NS DS RRSIG NSEC
             3600 RRSIG  NSEC 5 2 3600 20040509183619 (
                             20040409183619 38519 example.
                             cOlYgqJLqlRqmBQ3iap2SyIsK4O5aqpKSoba
                             U9fQ5SMApZmHfq3AgLflkrkXRXvgxTQSKkG2
                             039/cRUs6Jk/25+fi7Xr5nOVJsb0lq4zsB3I
                             BBdjyGDAHE0F5ROJj87996vJupdm1fbH481g
                             sdkOW6Zyqtz3Zos8N0BBkEx+2G4= )
ns1.a.example. 3600 IN A   192.0.2.5
ns2.a.example. 3600 IN A   192.0.2.6
ai.example.  3600 IN A    192.0.2.9
             3600 RRSIG  A 5 2 3600 20040509183619 (
                             20040409183619 38519 example.
```

```
                           pAOtzLP2MU0tDJUwHOKE5FPIIHmdYsCgTb5B
                           ERGgpnJluA9ixOyf6xxVCgrEJW0WNZSsJicd
                           hBHXfDmAGKUajUUlYSAH8tS4ZnrhyymIvk3u
                           ArDu2wfT130e9UHnumaHHMpUTosKe22PblOy
                           6zrTpg9FkS0XGVmYRvOTNYx2HvQ= )
              3600 HINFO   "KLH-10" "ITS"
              3600 RRSIG   HINFO 5 2 3600 20040509183619 (
                           20040409183619 38519 example.
                           Iq/RGCbBdKzcYzlGE4ovbr5YcB+ezxbZ9W0l
                           e/7WqyvhOO9J16HxhhL7VY/IKmTUY0GGdcfh
                           ZEOCkf4lEykZF9NPok1/R/fWrtzNp8jobuY7
                           AZEcZadp1WdDF3jc2/ndCa5XZhLKD3JzOsBw
                           FvL8sqlS5QS6FY/ijFEDnI4RkZA= )
              3600 AAAA    2001:db8::f00:baa9
              3600 RRSIG   AAAA 5 2 3600 20040509183619 (
                           20040409183619 38519 example.
                           nLcpFuXdT35AcE+EoafOUkl69KB+/e56XmFK
                           kewXG2IadYLKAOBIoR5+VoQV3XgTcofTJNsh
                           1rnF6Eav2zpZB3byI6yo2bwY8MNkr4A7cL9T
                           cMmDwV/hWFKsbGBsj8xSCN/caEL2CWY/5XP2
                           sZM6QjBBLmukH30+w1z3h8PUP2o= )
              3600 NSEC    b.example. A HINFO AAAA RRSIG NSEC
              3600 RRSIG   NSEC 5 2 3600 20040509183619 (
                           20040409183619 38519 example.
                           QoshyPevLcJ/xcRpEtMft1uoIrcrieVcc9pG
                           CScIn5Glnib40T6ayVOimXwdSTZ/8ISXGj4p
                           P8Sh0PlA6olZQ84L453/BUqB8BpdOGky4hsN
                           3AGcLEv1Gr0QMvirQaFcjzOECfnGyBm+wpFL
                           AhS+JOVfDI/79QtyTI0SaDWcg8U= )
b.example.    3600 IN NS   ns1.b.example.
              3600 IN NS   ns2.b.example.
              3600 NSEC    ns1.example. NS RRSIG NSEC
              3600 RRSIG   NSEC 5 2 3600 20040509183619 (
                           20040409183619 38519 example.
                           GNuxHn844wfmUhPzGWKJCPY5ttEX/RfjDoOx
                           9ueK1PtYkOWKOOdiJ/PJKCYB3hYX+858dDWS
                           xb2qnV/LSTCNVBnkm6owOpysY97MVj5VQEWs
                           0lm9tFoqjcptQkmQKYPrwUnCSNwvvclSF1xZ
                           vhRXgWT7OuFXldoCG6TfVFMs9xE= )
ns1.b.example. 3600 IN A   192.0.2.7
ns2.b.example. 3600 IN A   192.0.2.8
ns1.example.  3600 IN A    192.0.2.1
              3600 RRSIG   A 5 2 3600 20040509183619 (
                           20040409183619 38519 example.
                           F1C9HVhIcs10cZU09G5yIVfKJy5yRQQ3qVet
                           5pGhp82pzhAOMZ3K22JnmK4c+IjUeFp/to06
                           im5FVpHtbFisdjyPq84bhTv8vrXt5AB1wNB+
                           +iAqvIfdgW4sFNC6oADb1hK8QNauw9VePJhK
```

```
                                v/iVXSYC0b7mPSU+EOlknFpVECs= )
                    3600 NSEC   ns2.example. A RRSIG NSEC
                    3600 RRSIG  NSEC 5 2 3600 20040509183619 (
                                20040409183619 38519 example.
                                I4hj+Kt6+8rCcHcUdolks2S+Wzri9h3fHas8
                                1rGN/eILdJHN7JpV6lLGPIh/8fIBkfvdyWnB
                                jjf1q3O7JgYO1UdI7FvBNWqaaEPJK3UkddBq
                                ZIaLi8Qr2XHkjq38BeQsbp8X0+6h4ETWSGT8
                                IZaIGBLryQWGLw6Y6X8dqhlnxJM= )
  ns2.example.      3600 IN A   192.0.2.2
                    3600 RRSIG  A 5 2 3600 20040509183619 (
                                20040409183619 38519 example.
                                V7cQRw1TR+knlaL1z/psxlS1PcD37JJDaCMq
                                Qo6/u1qFQu6x+wuDHRH22Ap9ulJPQjFwMKOu
                                yfPGQPC8KzGdE3vt5snFEAoE1Vn3mQqtu7SO
                                6amIjk13Kj/jyJ4nGmdRIc/3cM3ipXFhNTKq
                                rdhx8SZ0yy4ObIRzIzvBFLiSS8o= )
                    3600 NSEC   *.w.example. A RRSIG NSEC
                    3600 RRSIG  NSEC 5 2 3600 20040509183619 (
                                20040409183619 38519 example.
                                N0QzHvaJf5NRw1rE9uxS1Ltb2LZ73Qb9bKGE
                                VyaISkqzGpP3jYJXZJPVTq4UVEsgT3CgeHvb
                                3QbeJ5Dfb2V9NGCHj/OvF/LBxFFWwhLwzngH
                                l+bQAgAcMsLu/nL3nDi1y/JSQjAcdZNDl4bw
                                Ymx28EtgIpo9A0qmP08rMBqs1Jw= )
  *.w.example.      3600 IN MX  1 ai.example.
                    3600 RRSIG  MX 5 2 3600 20040509183619 (
                                20040409183619 38519 example.
                                OMK8rAZlepfzLWW75Dxd63jy2wswESzxDKG2
                                f9AMN1CytCd10cYISAxfAdvXSZ7xujKAtPbc
                                tvOQ2ofO7AZJ+d01EeeQTVBPq4/6KCWhqe2X
                                TjnkVLNvvhnc0u28aoSsG0+4InvkkOHknKxw
                                4kX18MMR34i8lC36SR5xBni8vHI= )
                    3600 NSEC   x.w.example. MX RRSIG NSEC
                    3600 RRSIG  NSEC 5 2 3600 20040509183619 (
                                20040409183619 38519 example.
                                r/mZnRC3I/VIcrelgIcteSxDhtsdlTDt8ng9
                                HSBlABOlzLxQtfgTnn8f+aOwJIAFe1Ee5RvU
                                5cVhQJNP5XpXMJHfyps8tVvfxSAXfahpYqtx
                                91gsmcV/1V9/bZAG55CefP9cM4Z9Y9NT9XQ8
                                s1InQ2UoIv6tJEaaKkP701j8OLA= )
  x.w.example.      3600 IN MX  1 xx.example.
                    3600 RRSIG  MX 5 3 3600 20040509183619 (
                                20040409183619 38519 example.
                                Il2WTZ+Bkv+OytBx4LItNW5mjB4RCwhOO8y1
                                XzPHZmZUTVYL7LaA63f6T9ysVBzJRI3KRjAP
                                H3U1qaYnDoN1DrWqmi9RJe4FoObkbcdm7P3I
                                kx70ePCoFgRz1Yq+bVVXCvGuAU4xALv3W/Y1
```

```
                              jNSlwZ2mSWKHfxFQxPtLj8s32+k= )
                   3600 NSEC   x.y.w.example. MX RRSIG NSEC
                   3600 RRSIG  NSEC 5 3 3600 20040509183619 (
                              20040409183619 38519 example.
                              aRbpHftxggzgMXdDlym9SsADqMZovZZl2QWK
                              vw8J0tZEUNQByH5Qfnf5N1FqH/pS46UA7A4E
                              mcWBN9PUA1pdPY6RVeaRlZlCr1IkVctvbtaI
                              NJuBba/VHm+pebTbKcAPIvL9tBOoh+to1h6e
                              IjgiM8PXkBQtxPq37wDKALkyn7Q= )
  x.y.w.example. 3600 IN MX   1 xx.example.
                   3600 RRSIG  MX 5 4 3600 20040509183619 (
                              20040409183619 38519 example.
                              k2bJHbwP5LH5qN4is39UiPzjAWYmJA38Hhia
                              t7i9t7nbX/e0FPnvDSQXzcK7UL+zrVA+3MDj
                              q1ub4q3SZgcbLMgexxIW3Va//LVrxkP6Xupq
                              GtOB9prkK54QTl/qZTXfMQpW480YOvVknhvb
                              +gLcMZBnHJ326nb/TOOmrqNmQQE= )
                   3600 NSEC   xx.example. MX RRSIG NSEC
                   3600 RRSIG  NSEC 5 4 3600 20040509183619 (
                              20040409183619 38519 example.
                              OvE6WUzN2ziieJcvKPWbCAyXyP6ef8cr6Csp
                              ArVSTzKSquNwbezZmkU7E34o5lmb6CWSSSpg
                              xw098kNUFnHcQf/LzY2zqRomubrNQhJTiDTX
                              a0ArunJQCzPjOYq5t0SLjm6qp6McJI1AP5Vr
                              QoKqJDCLnoAlcPOPKAm/jJkn3jk= )
  xx.example.    3600 IN A    192.0.2.10
                   3600 RRSIG  A 5 2 3600 20040509183619 (
                              20040409183619 38519 example.
                              kBF4YxMGWF0D8r0cztL+2fWWOvN1U/GYSpYP
                              7SoKoNQ4fZKyk+weWGlKLIUM+uE1zjVTPXoa
                              0Z6WG0oZp46rkl1EzMcdMgoaeUzzAJ2BMq+Y
                              VdxG9IK1yZkYGY9AgbTOGPoAgbJyO9EPULsx
                              kbIDV6GPPSZVusnZU6OMgdgzHV4= )
                   3600 HINFO  "KLH-10" "TOPS-20"
                   3600 RRSIG  HINFO 5 2 3600 20040509183619 (
                              20040409183619 38519 example.
                              GY2PLSXmMHkWHfLdggiox8+chWpeMNJLkML0
                              t+U/SXSUsoUdR91KNdNUkTDWamwcF8oFRjhq
                              BcPZ6EqrF+vl5v5oGuvSF7U52epfVTC+wWF8
                              3yCUeUw8YklhLWlvk8gQ15YKth0ITQy8/wI+
                              RgNvuwbioFSEuv2pNlkq0goYxNY= )
                   3600 AAAA   2001:db8::f00:baaa
                   3600 RRSIG  AAAA 5 2 3600 20040509183619 (
                              20040409183619 38519 example.
                              Zzj0yodDxcBLnnOIwDsuKo5WqiaK24DlKg9C
                              aGaxDFiKgKobUj2jilYQHpGFn2poFRetZd4z
                              ulyQkssz2QHrVrPuTMS22knudCiwP4LWpVTr
                              U4zfeA+rDz9stmSBP/4PekH/x2IoAYnwctd/
```

```
                         xS9cL2QgW7FChw16mzlkH6/vsfs= )
            3600 NSEC    example. A HINFO AAAA RRSIG NSEC
            3600 RRSIG   NSEC 5 2 3600 20040509183619 (
                         20040409183619 38519 example.
                         ZFWUln6Avc8bmGl5GFjD3BwT530DUZKHNuoY
                         9A8lgXYyrxu+pqgFiRVbyZRQvVB5pccEOT3k
                         mvHgEa/HzbDB4PIYY79W+VHrgOxzdQGGCZzi
                         asXrpSGOWwSOElghPnMIi8xdF7qtCntr382W
                         GghLahumFIpg4MO3LS/prgzVVWo= )
```

   The apex DNSKEY set includes two DNSKEY RRs, and the DNSKEY RDATA
   Flags indicate that each of these DNSKEY RRs is a zone key. One of
   these DNSKEY RRs also has the SEP flag set and has been used to sign
   the apex DNSKEY RRset; this is the key that should be hashed to
   generate a DS record to be inserted into the parent zone. The other
   DNSKEY is used to sign all the other RRsets in the zone.

   The zone includes a wildcard entry, "*.w.example". Note that the
   name "*.w.example" is used in constructing NSEC chains, and that the
   RRSIG covering the "*.w.example" MX RRset has a label count of 2.

   The zone also includes two delegations. The delegation to
   "b.example" includes an NS RRset, glue address records, and an NSEC
   RR; note that only the NSEC RRset is signed. The delegation to
   "a.example" provides a DS RR; note that only the NSEC and DS RRsets
   are signed.

Appendix B. Example Responses

   The examples in this section show response messages using the signed
   zone example in Appendix A.

B.1. Answer

   A successful query to an authoritative server.

   ;; Header: QR AA DO RCODE=0
   ;;
   ;; Question
   x.w.example.   IN MX

   ;; Answer
   x.w.example.   3600 IN MX 1 xx.example.
   x.w.example.   3600 RRSIG MX 5 3 3600 20040509183619 (
                         20040409183619 38519 example.
                         Il2WTZ+Bkv+OytBx4LItNW5mjB4RCwhOO8y1
                         XzPHZmZUTVYL7LaA63f6T9ysVBzJRI3KRjAP
                         H3U1qaYnDoN1DrWqmi9RJe4FoObkbcdm7P3I
```

```
                             kx70ePCoFgRz1Yq+bVVXCvGuAU4xALv3W/Y1
                             jNSlwZ2mSWKHfxFQxPtLj8s32+k= )

     ;; Authority
     example.        3600 NS     ns1.example.
     example.        3600 NS     ns2.example.
     example.        3600 RRSIG  NS 5 1 3600 20040509183619 (
                             20040409183619 38519 example.
                             gl13F00f2U0R+SWiXXLHwsMY+qStYy5k6zfd
                             EuivWc+wd1fmbNCyql0Tk7lHTX6UOxc8AgNf
                             4ISFve8XqF4q+o9qlnqIzmppU3LiNeKT4FZ8
                             RO5urFOvoMRTbQxW3U0hXWuggE4g3ZpsHv48
                             0HjMeRaZB/FRPGfJPajngcq6Kwg= )

     ;; Additional
     xx.example.     3600 IN A   192.0.2.10
     xx.example.     3600 RRSIG  A 5 2 3600 20040509183619 (
                             20040409183619 38519 example.
                             kBF4YxMGWF0D8r0cztL+2fWWOvN1U/GYSpYP
                             7SoKoNQ4fZKyk+weWGlKLIUM+uE1zjVTPXoa
                             0Z6WG0oZp46rkl1EzMcdMgoaeUzzAJ2BMq+Y
                             VdxG9IK1yZkYGY9AgbTOGPoAgbJyO9EPULsx
                             kbIDV6GPPSZVusnZU6OMgdgzHV4= )
     xx.example.     3600 AAAA   2001:db8::f00:baaa
     xx.example.     3600 RRSIG  AAAA 5 2 3600 20040509183619 (
                             20040409183619 38519 example.
                             Zzj0yodDxcBLnnOIwDsuKo5WqiaK24DlKg9C
                             aGaxDFiKgKobUj2jilYQHpGFn2poFRetZd4z
                             ulyQkssz2QHrVrPuTMS22knudCiwP4LWpVTr
                             U4zfeA+rDz9stmSBP/4PekH/x2IoAYnwctd/
                             xS9cL2QgW7FChw16mzlkH6/vsfs= )
     ns1.example.    3600 IN A   192.0.2.1
     ns1.example.    3600 RRSIG  A 5 2 3600 20040509183619 (
                             20040409183619 38519 example.
                             F1C9HVhIcs10cZU09G5yIVfKJy5yRQQ3qVet
                             5pGhp82pzhAOMZ3K22JnmK4c+IjUeFp/to06
                             im5FVpHtbFisdjyPq84bhTv8vrXt5AB1wNB+
                             +iAqvIfdgW4sFNC6oADb1hK8QNauw9VePJhK
                             v/iVXSYC0b7mPSU+EOlknFpVECs= )
     ns2.example.    3600 IN A   192.0.2.2
     ns2.example.    3600 RRSIG  A 5 2 3600 20040509183619 (
                             20040409183619 38519 example.
                             V7cQRw1TR+knlaL1z/psxlS1PcD37JJDaCMq
                             Qo6/u1qFQu6x+wuDHRH22Ap9ulJPQjFwMKOu
                             yfPGQPC8KzGdE3vt5snFEAoE1Vn3mQqtu7SO
                             6amIjk13Kj/jyJ4nGmdRIc/3cM3ipXFhNTKq
                             rdhx8SZ0yy4ObIRzIzvBFLiSS8o= )
```

B.2. Name Error

   An authoritative name error. The NSEC RRs prove that the name does
   not exist and that no covering wildcard exists.

   ;; Header: QR AA DO RCODE=3
   ;;
   ;; Question
   ml.example.   IN A

   ;; Answer
   ;; (empty)

   ;; Authority
   example.   3600 IN SOA ns1.example. bugs.x.w.example. (
                                 1081539377
                                 3600
                                 300
                                 3600000
                                 3600
)
   example.   3600 RRSIG SOA 5 1 3600 20040509183619 (
                                 20040409183619 38519 example.
                                 ONx0k36rcjaxYtcNgq6iQnpNV5+drqYAsC9h
                                 7TSJaHCqbhE67Sr6aH2xDUGcqQWu/n0UVzrF
                                 vkgO9ebarZ0GWDKcuwlM6eNB5SiX2K74l5LW
                                 DA7S/Un/IbtDq4Ay8NMNLQI7Dw7n4p8/rjkB
                                 jV7j86HyQgM5e7+miRAz8V01b0I=)
   b.example.   3600 NSEC   ns1.example. NS RRSIG NSEC
   b.example.   3600 RRSIG NSEC 5 2 3600 20040509183619 (
                                 20040409183619 38519 example.
                                 GNuxHn844wfmUhPzGWKJCPY5ttEX/RfjDoOx
                                 9ueK1PtYkOWKOOdiJ/PJKCYB3hYX+858dDWS
                                 xb2qnV/LSTCNVBnkm6owOpysY97MVj5VQEWs
                                 0lm9tFoqjcptQkmQKYPrwUnCSNwvvclSF1xZ
                                 vhRXgWT7OuFXldoCG6TfVFMs9xE=)
   example.   3600 NSEC   a.example. NS SOA MX RRSIG NSEC DNSKEY
   example.   3600 RRSIG NSEC 5 1 3600 20040509183619 (
                                 20040409183619 38519 example.
                                 O0k558jHhyrC97ISHnislm4kLMW48C7U7cBm
                                 FTfhke5iVqNRVTB1STLMpgpbDIC9hcryoO0V
                                 Z9ME5xPzUEhbvGnHd5sfzgFVeGxr5Nyyq4tW
                                 SDBgIBiLQUv1ivy29vhXy7WgR62dPrZ0PWvm
                                 jfFJ5arXf4nPxp/kEowGgBRzY/U=)

   ;; Additional
   ;; (empty)

B.3. No Data Error

   A "no data" response. The NSEC RR proves that the name exists and
   that the requested RR type does not.

   ;; Header: QR AA DO RCODE=0
   ;;
   ;; Question
   ns1.example.   IN MX

   ;; Answer
   ;; (empty)

   ;; Authority
   example.   3600 IN SOA ns1.example. bugs.x.w.example. (
                                1081539377
                                3600
                                300
                                3600000
                                3600
)
   example.   3600 RRSIG SOA 5 1 3600 20040509183619 (
                                20040409183619 38519 example.
                                ONx0k36rcjaxYtcNgq6iQnpNV5+drqYAsC9h
                                7TSJaHCqbhE67Sr6aH2xDUGcqQWu/n0UVzrF
                                vkgO9ebarZ0GWDKcuwlM6eNB5SiX2K74l5LW
                                DA7S/Un/IbtDq4Ay8NMNLQI7Dw7n4p8/rjkB
                                jV7j86HyQgM5e7+miRAz8V01b0I=)
   ns1.example.   3600 NSEC   ns2.example. A RRSIG NSEC
   ns1.example.   3600 RRSIG NSEC 5 2 3600 20040509183619 (
                                20040409183619 38519 example.
                                I4hj+Kt6+8rCcHcUdolks2S+Wzri9h3fHas8
                                1rGN/eILdJHN7JpV6lLGPIh/8fIBkfvdyWnB
                                jjf1q3O7JgYO1UdI7FvBNWqaaEPJK3UkddBq
                                ZIaLi8Qr2XHkjq38BeQsbp8X0+6h4ETWSGT8
                                IZaIGBLryQWGLw6Y6X8dqhlnxJM=)

   ;; Additional
   ;; (empty)

B.4. Referral to Signed Zone

   Referral to a signed zone. The DS RR contains the data which the
   resolver will need to validate the corresponding DNSKEY RR in the
   child zone's apex.

   ;; Header: QR DO RCODE=0
   ;;

```
    ;; Question
    mc.a.example.       IN MX

    ;; Answer
    ;; (empty)

    ;; Authority
    a.example.      3600 IN NS  ns1.a.example.
    a.example.      3600 IN NS  ns2.a.example.
    a.example.      3600 DS     57855 5 1 (
                                B6DCD485719ADCA18E5F3D48A2331627FDD3
                                636B )
    a.example.      3600 RRSIG  DS 5 2 3600 20040509183619 (
                                20040409183619 38519 example.
                                oXIKit/QtdG64J/CB+Gi8dOvnwRvqrto1AdQ
                                oRkAN15FP3iZ7suB7gvTBmXzCjL7XUgQVcoH
                                kdhyCuzp8W9qJHgRUSwKKkczSyuL64nhgjuD
                                EML8l9wlWVsl7PR2VnZduM9bLyBhaaPmRKX/
                                Fm+v6ccF2EGNLRiY08kdkz+XHHo= )

    ;; Additional
    ns1.a.example. 3600 IN A   192.0.2.5
    ns2.a.example. 3600 IN A   192.0.2.6
```

B.5. Referral to Unsigned Zone

   Referral to an unsigned zone. The NSEC RR proves that no DS RR for
   this delegation exists in the parent zone.

```
    ;; Header: QR DO RCODE=0
    ;;
    ;; Question
    mc.b.example.       IN MX

    ;; Answer
    ;; (empty)

    ;; Authority
    b.example.      3600 IN NS  ns1.b.example.
    b.example.      3600 IN NS  ns2.b.example.
    b.example.      3600 NSEC   ns1.example. NS RRSIG NSEC
    b.example.      3600 RRSIG  NSEC 5 2 3600 20040509183619 (
                                20040409183619 38519 example.
                                GNuxHn844wfmUhPzGWKJCPY5ttEX/RfjDoOx
                                9ueK1PtYkOWKOOdiJ/PJKCYB3hYX+858dDWS
                                xb2qnV/LSTCNVBnkm6owOpysY97MVj5VQEWs
                                0lm9tFoqjcptQkmQKYPrwUnCSNwvvclSF1xZ
                                vhRXgWT7OuFXldoCG6TfVFMs9xE= )
```

```
   ;; Additional
   ns1.b.example. 3600 IN A   192.0.2.7
   ns2.b.example. 3600 IN A   192.0.2.8
```

B.6. Wildcard Expansion

   A successful query that was answered via wildcard expansion. The
   label count in the answer's RRSIG RR indicates that a wildcard RRset
   was expanded to produce this response, and the NSEC RR proves that no
   closer match exists in the zone.

```
   ;; Header: QR AA DO RCODE=0
   ;;
   ;; Question
   a.z.w.example.      IN MX

   ;; Answer
   a.z.w.example. 3600 IN MX  1 ai.example.
   a.z.w.example. 3600 RRSIG  MX 5 2 3600 20040509183619 (
                               20040409183619 38519 example.
                               OMK8rAZlepfzLWW75Dxd63jy2wswESzxDKG2
                               f9AMN1CytCd10cYISAxfAdvXSZ7xujKAtPbc
                               tvOQ2ofO7AZJ+d01EeeQTVBPq4/6KCWhqe2X
                               TjnkVLNvvhnc0u28aoSsG0+4InvkkOHknKxw
                               4kX18MMR34i8lC36SR5xBni8vHI= )

   ;; Authority
   example.       3600 NS    ns1.example.
   example.       3600 NS    ns2.example.
   example.       3600 RRSIG NS 5 1 3600 20040509183619 (
                               20040409183619 38519 example.
                               gl13F00f2U0R+SWiXXLHwsMY+qStYy5k6zfd
                               EuivWc+wd1fmbNCyql0Tk7lHTX6UOxc8AgNf
                               4ISFve8XqF4q+o9qlnqIzmppU3LiNeKT4FZ8
                               RO5urFOvoMRTbQxW3U0hXWuggE4g3ZpsHv48
                               0HjMeRaZB/FRPGfJPajngcq6Kwg= )
   x.y.w.example. 3600 NSEC   xx.example. MX RRSIG NSEC
   x.y.w.example. 3600 RRSIG  NSEC 5 4 3600 20040509183619 (
                               20040409183619 38519 example.
                               OvE6WUzN2ziieJcvKPWbCAyXyP6ef8cr6Csp
                               ArVSTzKSquNwbezZmkU7E34o5lmb6CWSSSpg
                               xw098kNUFnHcQf/LzY2zqRomubrNQhJTiDTX
                               a0ArunJQCzPjOYq5t0SLjm6qp6McJI1AP5Vr
                               QoKqJDCLnoAlcPOPKAm/jJkn3jk= )

   ;; Additional
   ai.example.    3600 IN A   192.0.2.9
   ai.example.    3600 RRSIG  A 5 2 3600 20040509183619 (
```

```
                            20040409183619 38519 example.
                            pAOtzLP2MU0tDJUwHOKE5FPIIHmdYsCgTb5B
                            ERGgpnJluA9ixOyf6xxVCgrEJW0WNZSsJicd
                            hBHXfDmAGKUajUUlYSAH8tS4Znrhyymlvk3u
                            ArDu2wfT130e9UHnumaHHMpUTosKe22PblOy
                            6zrTpg9FkS0XGVmYRvOTNYx2HvQ= )
     ai.example.     3600 AAAA   2001:db8::f00:baa9
     ai.example.     3600 RRSIG  AAAA 5 2 3600 20040509183619 (
                            20040409183619 38519 example.
                            nLcpFuXdT35AcE+EoafOUkl69KB+/e56XmFK
                            kewXG2IadYLKAOBIoR5+VoQV3XgTcofTJNsh
                            1rnF6Eav2zpZB3byI6yo2bwY8MNkr4A7cL9T
                            cMmDwV/hWFKsbGBsj8xSCN/caEL2CWY/5XP2
                            sZM6QjBBLmukH30+w1z3h8PUP2o= )
```

B.7. Wildcard No Data Error

   A "no data" response for a name covered by a wildcard. The NSEC RRs
   prove that the matching wildcard name does not have any RRs of the
   requested type and that no closer match exists in the zone.

```
   ;; Header: QR AA DO RCODE=0
   ;;
   ;; Question
   a.z.w.example.        IN AAAA

   ;; Answer
   ;; (empty)

   ;; Authority
   example.        3600 IN SOA ns1.example. bugs.x.w.example. (
                            1081539377
                            3600
                            300
                            3600000
                            3600
                            )
   example.        3600 RRSIG  SOA 5 1 3600 20040509183619 (
                            20040409183619 38519 example.
                            ONx0k36rcjaxYtcNgq6iQnpNV5+drqYAsC9h
                            7TSJaHCqbhE67Sr6aH2xDUGcqQWu/n0UVzrF
                            vkgO9ebarZ0GWDKcuwlM6eNB5SiX2K74l5LW
                            DA7S/Un/IbtDq4Ay8NMNLQI7Dw7n4p8/rjkB
                            jV7j86HyQgM5e7+miRAz8V01b0I= )
   x.y.w.example. 3600 NSEC    xx.example. MX RRSIG NSEC
   x.y.w.example. 3600 RRSIG   NSEC 5 4 3600 20040509183619 (
                            20040409183619 38519 example.
                            OvE6WUzN2ziieJcvKPWbCAyXyP6ef8cr6Csp
```

```
                               ArVSTzKSquNwbezZmkU7E34o5lmb6CWSSSpg
                               xw098kNUFnHcQf/LzY2zqRomubrNQhJTiDTX
                               a0ArunJQCzPjOYq5t0SLjm6qp6McJI1AP5Vr
                               QoKqJDCLnoAlcPOPKAm/jJkn3jk= )
     *.w.example.    3600 NSEC   x.w.example. MX RRSIG NSEC
     *.w.example.    3600 RRSIG  NSEC 5 2 3600 20040509183619 (
                               20040409183619 38519 example.
                               r/mZnRC3I/VIcrelgIcteSxDhtsdlTDt8ng9
                               HSBlABOlzLxQtfgTnn8f+aOwJIAFe1Ee5RvU
                               5cVhQJNP5XpXMJHfyps8tVvfxSAXfahpYqtx
                               91gsmcV/1V9/bZAG55CefP9cM4Z9Y9NT9XQ8
                               s1InQ2UoIv6tJEaaKkP701j8OLA= )

     ;; Additional
     ;; (empty)
```

B.8. DS Child Zone No Data Error

   A "no data" response for a QTYPE=DS query that was mistakenly sent to
   a name server for the child zone.

```
   ;; Header: QR AA DO RCODE=0
   ;;
   ;; Question
   example.            IN DS

   ;; Answer
   ;; (empty)

   ;; Authority
   example.        3600 IN SOA ns1.example. bugs.x.w.example. (
                               1081539377
                               3600
                               300
                               3600000
                               3600
                               )
   example.        3600 RRSIG  SOA 5 1 3600 20040509183619 (
                               20040409183619 38519 example.
                               ONx0k36rcjaxYtcNgq6iQnpNV5+drqYAsC9h
                               7TSJaHCqbhE67Sr6aH2xDUGcqQWu/n0UVzrF
                               vkgO9ebarZ0GWDKcuwlM6eNB5SiX2K74l5LW
                               DA7S/Un/IbtDq4Ay8NMNLQI7Dw7n4p8/rjkB
                               jV7j86HyQgM5e7+miRAz8V01b0I= )
   example.        3600 NSEC   a.example. NS SOA MX RRSIG NSEC DNSKEY
   example.        3600 RRSIG  NSEC 5 1 3600 20040509183619 (
                               20040409183619 38519 example.
                               O0k558jHhyrC97ISHnislm4kLMW48C7U7cBm
```

                              FTfhke5iVqNRVTB1STLMpgpbDIC9hcryoO0V
                              Z9ME5xPzUEhbvGnHd5sfzgFVeGxr5Nyyq4tW
                              SDBgIBiLQUv1ivy29vhXy7WgR62dPrZ0PWvm
                              jfFJ5arXf4nPxp/kEowGgBRzY/U=)

```
   ;; Additional
   ;; (empty)
```

Appendix C. Authentication Examples

   The examples in this section show how the response messages in
   Appendix B are authenticated.

C.1. Authenticating an Answer

   The query in Appendix B.1 returned an MX RRset for "x.w.example.com".
   The corresponding RRSIG indicates that the MX RRset was signed by an
   "example" DNSKEY with algorithm 5 and key tag 38519. The resolver
   needs the corresponding DNSKEY RR in order to authenticate this
   answer. The discussion below describes how a resolver might obtain
   this DNSKEY RR.

   The RRSIG indicates the original TTL of the MX RRset was 3600, and,
   for the purpose of authentication, the current TTL is replaced by
   3600. The RRSIG labels field value of 3 indicates that the answer
   was not the result of wildcard expansion. The "x.w.example.com" MX
   RRset is placed in canonical form, and, assuming the current time
   falls between the signature inception and expiration dates, the
   signature is authenticated.

C.1.1. Authenticating the Example DNSKEY RR

   This example shows the logical authentication process that starts
   from the a configured root DNSKEY (or DS RR) and moves down the tree
   to authenticate the desired "example" DNSKEY RR. Note that the
   logical order is presented for clarity. An implementation may choose
   to construct the authentication as referrals are received or to
   construct the authentication chain only after all RRsets have been
   obtained, or in any other combination it sees fit. The example here
   demonstrates only the logical process and does not dictate any
   implementation rules.

   We assume the resolver starts with a configured DNSKEY RR for the
   root zone (or a configured DS RR for the root zone). The resolver
   checks whether this configured DNSKEY RR is present in the root
   DNSKEY RRset (or whether the DS RR matches some DNSKEY in the root
   DNSKEY RRset), whether this DNSKEY RR has signed the root DNSKEY
   RRset, and whether the signature lifetime is valid. If all these

   conditions are met, all keys in the DNSKEY RRset are considered
   authenticated. The resolver then uses one (or more) of the root
   DNSKEY RRs to authenticate the "example" DS RRset. Note that the
   resolver may have to query the root zone to obtain the root DNSKEY
   RRset or "example" DS RRset.

   Once the DS RRset has been authenticated using the root DNSKEY, the
   resolver checks the "example" DNSKEY RRset for some "example" DNSKEY
   RR that matches one of the authenticated "example" DS RRs. If such a
   matching "example" DNSKEY is found, the resolver checks whether this
   DNSKEY RR has signed the "example" DNSKEY RRset and the signature
   lifetime is valid. If these conditions are met, all keys in the
   "example" DNSKEY RRset are considered authenticated.

   Finally, the resolver checks that some DNSKEY RR in the "example"
   DNSKEY RRset uses algorithm 5 and has a key tag of 38519. This
   DNSKEY is used to authenticate the RRSIG included in the response.
   If multiple "example" DNSKEY RRs match this algorithm and key tag,
   then each DNSKEY RR is tried, and the answer is authenticated if any
   of the matching DNSKEY RRs validate the signature as described above.

C.2. Name Error

   The query in Appendix B.2 returned NSEC RRs that prove that the
   requested data does not exist and no wildcard applies. The negative
   reply is authenticated by verifying both NSEC RRs. The NSEC RRs are
   authenticated in a manner identical to that of the MX RRset discussed
   above.

C.3. No Data Error

   The query in Appendix B.3 returned an NSEC RR that proves that the
   requested name exists, but the requested RR type does not exist. The
   negative reply is authenticated by verifying the NSEC RR. The NSEC
   RR is authenticated in a manner identical to that of the MX RRset
   discussed above.

C.4. Referral to Signed Zone

   The query in Appendix B.4 returned a referral to the signed
   "a.example." zone. The DS RR is authenticated in a manner identical
   to that of the MX RRset discussed above. This DS RR is used to
   authenticate the "a.example" DNSKEY RRset.

   Once the "a.example" DS RRset has been authenticated using the
   "example" DNSKEY, the resolver checks the "a.example" DNSKEY RRset
   for some "a.example" DNSKEY RR that matches the DS RR. If such a
   matching "a.example" DNSKEY is found, the resolver checks whether

      this DNSKEY RR has signed the "a.example" DNSKEY RRset and whether
      the signature lifetime is valid. If all these conditions are met,
      all keys in the "a.example" DNSKEY RRset are considered
      authenticated.

C.5. Referral to Unsigned Zone

      The query in Appendix B.5 returned a referral to an unsigned
      "b.example." zone. The NSEC proves that no authentication leads from
      "example" to "b.example", and the NSEC RR is authenticated in a
      manner identical to that of the MX RRset discussed above.

C.6. Wildcard Expansion

      The query in Appendix B.6 returned an answer that was produced as a
      result of wildcard expansion. The answer section contains a wildcard
      RRset expanded as it would be in a traditional DNS response, and the
      corresponding RRSIG indicates that the expanded wildcard MX RRset was
      signed by an "example" DNSKEY with algorithm 5 and key tag 38519.
      The RRSIG indicates that the original TTL of the MX RRset was 3600,
      and, for the purpose of authentication, the current TTL is replaced
      by 3600. The RRSIG labels field value of 2 indicates that the answer
      is the result of wildcard expansion, as the "a.z.w.example" name
      contains 4 labels. The name "a.z.w.w.example" is replaced by
      "*.w.example", the MX RRset is placed in canonical form, and,
      assuming that the current time falls between the signature inception
      and expiration dates, the signature is authenticated.

      The NSEC proves that no closer match (exact or closer wildcard) could
      have been used to answer this query, and the NSEC RR must also be
      authenticated before the answer is considered valid.

C.7. Wildcard No Data Error

      The query in Appendix B.7 returned NSEC RRs that prove that the
      requested data does not exist and no wildcard applies. The negative
      reply is authenticated by verifying both NSEC RRs.

C.8. DS Child Zone No Data Error

      The query in Appendix B.8 returned NSEC RRs that shows the requested
      was answered by a child server ("example" server). The NSEC RR
      indicates the presence of an SOA RR, showing that the answer is from
      the child . Queries for the "example" DS RRset should be sent to the
      parent servers ("root" servers).

Authors' Addresses

   Roy Arends
   Telematica Instituut
   Brouwerijstraat 1
   7523 XC Enschede
   NL

   EMail: roy.arends@telin.nl

   Rob Austein
   Internet Systems Consortium
   950 Charter Street
   Redwood City, CA 94063
   USA

   EMail: sra@isc.org

   Matt Larson
   VeriSign, Inc.
   21345 Ridgetop Circle
   Dulles, VA 20166-6503
   USA

   EMail: mlarson@verisign.com

   Dan Massey
   Colorado State University
   Department of Computer Science
   Fort Collins, CO 80523-1873

   EMail: massey@cs.colostate.edu

   Scott Rose
   National Institute for Standards and Technology
   100 Bureau Drive
   Gaithersburg, MD 20899-8920
   USA

   EMail: scott.rose@nist.gov