         DNS Security (DNSSEC) Hashed Authenticated Denial of Existence

Status of This Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

Abstract

   The Domain Name System Security (DNSSEC) Extensions introduced the
   NSEC resource record (RR) for authenticated denial of existence.
   This document introduces an alternative resource record, NSEC3, which
   similarly provides authenticated denial of existence. However, it
   also provides measures against zone enumeration and permits gradual
   expansion of delegation-centric zones.

Table of Contents

1. Introduction

1.1. Rationale

   The DNS Security Extensions included the NSEC RR to provide
   authenticated denial of existence. Though the NSEC RR meets the
   requirements for authenticated denial of existence, it introduces a
   side-effect in that the contents of a zone can be enumerated. This
   property introduces undesired policy issues.

   The enumeration is enabled by the set of NSEC records that exists
   inside a signed zone. An NSEC record lists two names that are
   ordered canonically, in order to show that nothing exists between the
   two names. The complete set of NSEC records lists all the names in a
   zone. It is trivial to enumerate the content of a zone by querying
   for names that do not exist.

   An enumerated zone can be used, for example, as a source of probable
   e-mail addresses for spam, or as a key for multiple WHOIS queries to
   reveal registrant data that many registries may have legal
   obligations to protect. Many registries therefore prohibit the
   copying of their zone data; however, the use of NSEC RRs renders
   these policies unenforceable.

   A second problem is that the cost to cryptographically secure
   delegations to unsigned zones is high, relative to the perceived
   security benefit, in two cases: large, delegation-centric zones, and
   zones where insecure delegations will be updated rapidly. In these
   cases, the costs of maintaining the NSEC RR chain may be extremely
   high and use of the "Opt-Out" convention may be more appropriate (for
   these unsecured zones).

   This document presents the NSEC3 Resource Record which can be used as
   an alternative to NSEC to mitigate these issues.

   Earlier work to address these issues include [DNSEXT-NO], [RFC4956],
   and [DNSEXT-NSEC2v2].

1.2. Requirements

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

1.3. Terminology

   The reader is assumed to be familiar with the basic DNS and DNSSEC
   concepts described in [RFC1034], [RFC1035], [RFC4033], [RFC4034],
   [RFC4035], and subsequent RFCs that update them: [RFC2136],
   [RFC2181], and [RFC2308].

   The following terminology is used throughout this document:

   Zone enumeration: the practice of discovering the full content of a
      zone via successive queries. Zone enumeration was non-trivial
      prior to the introduction of DNSSEC.

   Original owner name: the owner name corresponding to a hashed owner
      name.

   Hashed owner name: the owner name created after applying the hash
      function to an owner name.

   Hash order: the order in which hashed owner names are arranged
      according to their numerical value, treating the leftmost (lowest
      numbered) octet as the most significant octet. Note that this
      order is the same as the canonical DNS name order specified in
      [RFC4034], when the hashed owner names are in base32, encoded with
      an Extended Hex Alphabet [RFC4648].

   Empty non-terminal: a domain name that owns no resource records, but
      has one or more subdomains that do.

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

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

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

   Opt-Out NSEC3 resource record: an NSEC3 resource record that has the
      Opt-Out flag set to 1.

   Opt-Out zone: a zone with at least one Opt-Out NSEC3 RR.

   Closest encloser: the longest existing ancestor of a name. See also
      Section 3.3.1 of [RFC4592].

   Closest provable encloser: the longest ancestor of a name that can
      be proven to exist. Note that this is only different from the
      closest encloser in an Opt-Out zone.

   Next closer name: the name one label longer than the closest
      provable encloser of a name.

   Base32: the "Base 32 Encoding with Extended Hex Alphabet" as
      specified in [RFC4648]. Note that trailing padding characters
      ("=") are not used in the NSEC3 specification.

   To cover: An NSEC3 RR is said to "cover" a name if the hash of the
      name or "next closer" name falls between the owner name and the
      next hashed owner name of the NSEC3. In other words, if it proves
      the nonexistence of the name, either directly or by proving the
      nonexistence of an ancestor of the name.

   To match: An NSEC3 RR is said to "match" a name if the owner name of
      the NSEC3 RR is the same as the hashed owner name of that name.

2. Backwards Compatibility

   This specification describes a protocol change that is not generally
   backwards compatible with [RFC4033], [RFC4034], and [RFC4035]. In
   particular, security-aware resolvers that are unaware of this
   specification (NSEC3-unaware resolvers) may fail to validate the
   responses introduced by this document.

   In order to aid deployment, this specification uses a signaling
   technique to prevent NSEC3-unaware resolvers from attempting to
   validate responses from NSEC3-signed zones.

   This specification allocates two new DNSKEY algorithm identifiers for
   this purpose. Algorithm 6, DSA-NSEC3-SHA1 is an alias for algorithm
   3, DSA. Algorithm 7, RSASHA1-NSEC3-SHA1 is an alias for algorithm 5,
   RSASHA1. These are not new algorithms, they are additional
   identifiers for the existing algorithms.

   Zones signed according to this specification MUST only use these
   algorithm identifiers for their DNSKEY RRs. Because these new
   identifiers will be unknown algorithms to existing, NSEC3-unaware
   resolvers, those resolvers will then treat responses from the NSEC3
   signed zone as insecure, as detailed in Section 5.2 of [RFC4035].

   These algorithm identifiers are used with the NSEC3 hash algorithm
   SHA1. Using other NSEC3 hash algorithms requires allocation of a new
   alias (see Section 12.1.3).

   Security aware resolvers that are aware of this specification MUST
   recognize the new algorithm identifiers and treat them as equivalent
   to the algorithms that they alias.

   A methodology for transitioning from a DNSSEC signed zone to a zone
   signed using NSEC3 is discussed in Section 10.4.

3. The NSEC3 Resource Record

   The NSEC3 Resource Record (RR) provides authenticated denial of
   existence for DNS Resource Record Sets.

   The NSEC3 RR lists RR types present at the original owner name of the
   NSEC3 RR. It includes the next hashed owner name in the hash order
   of the zone. The complete set of NSEC3 RRs in a zone indicates which
   RRSets exist for the original owner name of the RR and form a chain
   of hashed owner names in the zone. This information is used to
   provide authenticated denial of existence for DNS data. To provide
   protection against zone enumeration, the owner names used in the
   NSEC3 RR are cryptographic hashes of the original owner name
   prepended as a single label to the name of the zone. The NSEC3 RR
   indicates which hash function is used to construct the hash, which
   salt is used, and how many iterations of the hash function are
   performed over the original owner name. The hashing technique is
   described fully in Section 5.

   Hashed owner names of unsigned delegations may be excluded from the
   chain. An NSEC3 RR whose span covers the hash of an owner name or
   "next closer" name of an unsigned delegation is referred to as an
   Opt-Out NSEC3 RR and is indicated by the presence of a flag.

   The owner name for the NSEC3 RR is the base32 encoding of the hashed
   owner name prepended as a single label to the name of the zone.

   The type value for the NSEC3 RR is 50.

   The NSEC3 RR RDATA format is class independent and is described
   below.

   The class MUST be the same as the class of the original owner name.

   The NSEC3 RR SHOULD have the same TTL value as the SOA minimum TTL
   field. This is in the spirit of negative caching [RFC2308].

3.1. RDATA Fields

3.1.1. Hash Algorithm

   The Hash Algorithm field identifies the cryptographic hash algorithm
   used to construct the hash-value.

   The values for this field are defined in the NSEC3 hash algorithm
   registry defined in Section 11.

3.1.2. Flags

   The Flags field contains 8 one-bit flags that can be used to indicate
   different processing. All undefined flags must be zero. The only
   flag defined by this specification is the Opt-Out flag.

3.1.2.1. Opt-Out Flag

   If the Opt-Out flag is set, the NSEC3 record covers zero or more
   unsigned delegations.

   If the Opt-Out flag is clear, the NSEC3 record covers zero unsigned
   delegations.

   The Opt-Out Flag indicates whether this NSEC3 RR may cover unsigned
   delegations. It is the least significant bit in the Flags field.
   See Section 6 for details about the use of this flag.

3.1.3. Iterations

   The Iterations field defines the number of additional times the hash
   function has been performed. More iterations result in greater
   resiliency of the hash value against dictionary attacks, but at a
   higher computational cost for both the server and resolver. See
   Section 5 for details of the use of this field, and Section 10.3 for
   limitations on the value.

3.1.4. Salt Length

   The Salt Length field defines the length of the Salt field in octets,
   ranging in value from 0 to 255.

3.1.5. Salt

   The Salt field is appended to the original owner name before hashing
   in order to defend against pre-calculated dictionary attacks. See
   Section 5 for details on how the salt is used.

3.1.6. Hash Length

   The Hash Length field defines the length of the Next Hashed Owner
   Name field, ranging in value from 1 to 255 octets.

3.1.7. Next Hashed Owner Name

   The Next Hashed Owner Name field contains the next hashed owner name
   in hash order. This value is in binary format. Given the ordered
   set of all hashed owner names, the Next Hashed Owner Name field
   contains the hash of an owner name that immediately follows the owner
   name of the given NSEC3 RR. The value of the Next Hashed Owner Name
   field in the last NSEC3 RR in the zone is the same as the hashed
   owner name of the first NSEC3 RR in the zone in hash order. Note
   that, unlike the owner name of the NSEC3 RR, the value of this field
   does not contain the appended zone name.

3.1.8. Type Bit Maps

   The Type Bit Maps field identifies the RRSet types that exist at the
   original owner name of the NSEC3 RR.

3.2. NSEC3 RDATA Wire Format

   The RDATA of the NSEC3 RR is as shown below:

```
                      1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |   Hash Alg.   |     Flags     |            Iterations         |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |  Salt Length  |                     Salt                      /
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |  Hash Length  |             Next Hashed Owner Name            /
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  /                         Type Bit Maps                         /
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
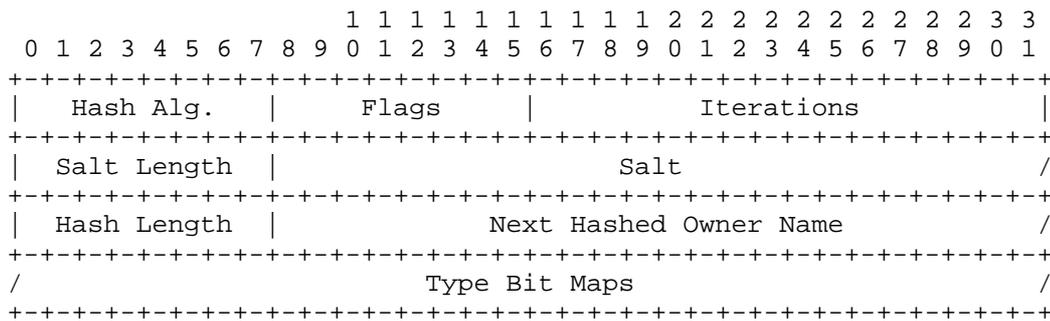
   Hash Algorithm is a single octet.

   Flags field is a single octet, the Opt-Out flag is the least
   significant bit, as shown below:
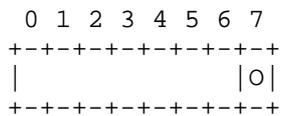
```
   0 1 2 3 4 5 6 7
  +-+-+-+-+-+-+-+-+
  |             |O|
  +-+-+-+-+-+-+-+-+
```

Iterations is represented as a 16-bit unsigned integer, with the most significant bit first.

Salt Length is represented as an unsigned octet. Salt Length represents the length of the Salt field in octets. If the value is zero, the following Salt field is omitted.

Salt, if present, is encoded as a sequence of binary octets. The length of this field is determined by the preceding Salt Length field.

Hash Length is represented as an unsigned octet. Hash Length represents the length of the Next Hashed Owner Name field in octets.

The next hashed owner name is not base32 encoded, unlike the owner name of the NSEC3 RR. It is the unmodified binary hash value. It does not include the name of the containing zone. The length of this field is determined by the preceding Hash Length field.

3.2.1. Type Bit Maps Encoding

The encoding of the Type Bit Maps field is the same as that used by the NSEC RR, described in [RFC4034]. It is explained and clarified here for clarity.

The RR type space is split into 256 window blocks, each representing the low-order 8 bits of the 16-bit RR type space. Each block that has at least one active RR type is encoded using a single octet window number (from 0 to 255), a single octet bitmap length (from 1 to 32) indicating the number of octets used for the bitmap of the window block, and up to 32 octets (256 bits) of bitmap.

Blocks are present in the NSEC3 RR RDATA in increasing numerical order.

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

    where "|" denotes concatenation.

Each bitmap encodes the low-order 8 bits of RR types within the window block, in network bit order. The first bit is bit 0. For window block 0, bit 1 corresponds to RR type 1 (A), bit 2 corresponds to RR type 2 (NS), and so forth. For window block 1, bit 1 corresponds to RR type 257, bit 2 to RR type 258. If a bit is set to 1, it indicates that an RRSet of that type is present for the original owner name of the NSEC3 RR. If a bit is set to 0, it indicates that no RRSet of that type is present for the original owner name of the NSEC3 RR.

   Since bit 0 in window block 0 refers to the non-existing RR type 0,
   it MUST be set to 0. After verification, the validator MUST ignore
   the value of bit 0 in window block 0.

   Bits representing Meta-TYPEs or QTYPEs as specified in Section 3.1 of
   [RFC2929] or within the range reserved for assignment only to QTYPEs
   and Meta-TYPEs MUST be set to 0, since they do not appear in zone
   data. If encountered, they must be ignored upon reading.

   Blocks with no types present MUST NOT be included. Trailing zero
   octets in the bitmap MUST be omitted. The length of the bitmap of
   each block is determined by the type code with the largest numerical
   value, within that block, among the set of RR types present at the
   original owner name of the NSEC3 RR. Trailing octets not specified
   MUST be interpreted as zero octets.

3.3. Presentation Format

   The presentation format of the RDATA portion is as follows:

   o The Hash Algorithm field is represented as an unsigned decimal
      integer. The value has a maximum of 255.

   o The Flags field is represented as an unsigned decimal integer.
      The value has a maximum of 255.

   o The Iterations field is represented as an unsigned decimal
      integer. The value is between 0 and 65535, inclusive.

   o The Salt Length field is not represented.

   o The Salt field is represented as a sequence of case-insensitive
      hexadecimal digits. Whitespace is not allowed within the
      sequence. The Salt field is represented as "-" (without the
      quotes) when the Salt Length field has a value of 0.

   o The Hash Length field is not represented.

   o The Next Hashed Owner Name field is represented as an unpadded
      sequence of case-insensitive base32 digits, without whitespace.

   o The Type Bit Maps field is represented as a sequence of RR type
      mnemonics. When the mnemonic is not known, the TYPE
      representation as described in Section 5 of [RFC3597] MUST be
      used.

4. The NSEC3PARAM Resource Record

   The NSEC3PARAM RR contains the NSEC3 parameters (hash algorithm,
   flags, iterations, and salt) needed by authoritative servers to
   calculate hashed owner names. The presence of an NSEC3PARAM RR at a
   zone apex indicates that the specified parameters may be used by
   authoritative servers to choose an appropriate set of NSEC3 RRs for
   negative responses. The NSEC3PARAM RR is not used by validators or
   resolvers.

   If an NSEC3PARAM RR is present at the apex of a zone with a Flags
   field value of zero, then there MUST be an NSEC3 RR using the same
   hash algorithm, iterations, and salt parameters present at every
   hashed owner name in the zone. That is, the zone MUST contain a
   complete set of NSEC3 RRs with the same hash algorithm, iterations,
   and salt parameters.

   The owner name for the NSEC3PARAM RR is the name of the zone apex.

   The type value for the NSEC3PARAM RR is 51.

   The NSEC3PARAM RR RDATA format is class independent and is described
   below.

   The class MUST be the same as the NSEC3 RRs to which this RR refers.

4.1. RDATA Fields

   The RDATA for this RR mirrors the first four fields in the NSEC3 RR.

4.1.1. Hash Algorithm

   The Hash Algorithm field identifies the cryptographic hash algorithm
   used to construct the hash-value.

   The acceptable values are the same as the corresponding field in the
   NSEC3 RR.

4.1.2. Flag Fields

   The Opt-Out flag is not used and is set to zero.

   All other flags are reserved for future use, and must be zero.

   NSEC3PARAM RRs with a Flags field value other than zero MUST be
   ignored.

4.1.3. Iterations

   The Iterations field defines the number of additional times the hash
   is performed.

   Its acceptable values are the same as the corresponding field in the
   NSEC3 RR.

4.1.4. Salt Length

   The Salt Length field defines the length of the salt in octets,
   ranging in value from 0 to 255.

4.1.5. Salt

   The Salt field is appended to the original owner name before hashing.

4.2. NSEC3PARAM RDATA Wire Format

   The RDATA of the NSEC3PARAM RR is as shown below:

```
                         1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   Hash Alg.   |     Flags     |            Iterations          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Salt Length  |                     Salt                      /
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
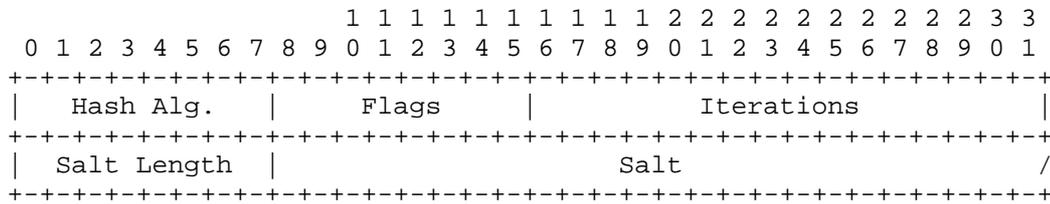
   Hash Algorithm is a single octet.

   Flags field is a single octet.

   Iterations is represented as a 16-bit unsigned integer, with the most
   significant bit first.

   Salt Length is represented as an unsigned octet. Salt Length
   represents the length of the following Salt field in octets. If the
   value is zero, the Salt field is omitted.

   Salt, if present, is encoded as a sequence of binary octets. The
   length of this field is determined by the preceding Salt Length
   field.

4.3. Presentation Format

   The presentation format of the RDATA portion is as follows:

   o The Hash Algorithm field is represented as an unsigned decimal
      integer. The value has a maximum of 255.

   o The Flags field is represented as an unsigned decimal integer.
      The value has a maximum value of 255.

   o The Iterations field is represented as an unsigned decimal
      integer. The value is between 0 and 65535, inclusive.

   o The Salt Length field is not represented.

   o The Salt field is represented as a sequence of case-insensitive
      hexadecimal digits. Whitespace is not allowed within the
      sequence. This field is represented as "-" (without the quotes)
      when the Salt Length field is zero.

5. Calculation of the Hash

   The hash calculation uses three of the NSEC3 RDATA fields: Hash
   Algorithm, Salt, and Iterations.

   Define H(x) to be the hash of x using the Hash Algorithm selected by
   the NSEC3 RR, k to be the number of Iterations, and || to indicate
   concatenation. Then define:

      IH(salt, x, 0) = H(x || salt), and

      IH(salt, x, k) = H(IH(salt, x, k-1) || salt), if k > 0

   Then the calculated hash of an owner name is

      IH(salt, owner name, iterations),

   where the owner name is in the canonical form, defined as:

   The wire format of the owner name where:

   1. The owner name is fully expanded (no DNS name compression) and
       fully qualified;

   2. All uppercase US-ASCII letters are replaced by the corresponding
       lowercase US-ASCII letters;

   3. If the owner name is a wildcard name, the owner name is in its
       original unexpanded form, including the "*" label (no wildcard
       substitution);

   This form is as defined in Section 6.2 of [RFC4034].

   The method to calculate the Hash is based on [RFC2898].

6. Opt-Out

   In this specification, as in [RFC4033], [RFC4034] and [RFC4035], NS
   RRSets at delegation points are not signed and may be accompanied by
   a DS RRSet. With the Opt-Out bit clear, the security status of the
   child zone is determined by the presence or absence of this DS RRSet,
   cryptographically proven by the signed NSEC3 RR at the hashed owner
   name of the delegation. Setting the Opt-Out flag modifies this by
   allowing insecure delegations to exist within the signed zone without
   a corresponding NSEC3 RR at the hashed owner name of the delegation.

   An Opt-Out NSEC3 RR is said to cover a delegation if the hash of the
   owner name or "next closer" name of the delegation is between the
   owner name of the NSEC3 RR and the next hashed owner name.

   An Opt-Out NSEC3 RR does not assert the existence or non-existence of
   the insecure delegations that it may cover. This allows for the
   addition or removal of these delegations without recalculating or re-
   signing RRs in the NSEC3 RR chain. However, Opt-Out NSEC3 RRs do
   assert the (non)existence of other, authoritative RRSets.

   An Opt-Out NSEC3 RR MAY have the same original owner name as an
   insecure delegation. In this case, the delegation is proven insecure
   by the lack of a DS bit in the type map and the signed NSEC3 RR does
   assert the existence of the delegation.

   Zones using Opt-Out MAY contain a mixture of Opt-Out NSEC3 RRs and
   non-Opt-Out NSEC3 RRs. If an NSEC3 RR is not Opt-Out, there MUST NOT
   be any hashed owner names of insecure delegations (nor any other RRs)
   between it and the name indicated by the next hashed owner name in
   the NSEC3 RDATA. If it is Opt-Out, it MUST only cover hashed owner
   names or hashed "next closer" names of insecure delegations.

   The effects of the Opt-Out flag on signing, serving, and validating
   responses are covered in following sections.

7. Authoritative Server Considerations

7.1. Zone Signing

   Zones using NSEC3 must satisfy the following properties:

   o Each owner name within the zone that owns authoritative RRSets
      MUST have a corresponding NSEC3 RR. Owner names that correspond
      to unsigned delegations MAY have a corresponding NSEC3 RR.
      However, if there is not a corresponding NSEC3 RR, there MUST be
      an Opt-Out NSEC3 RR that covers the "next closer" name to the
      delegation. Other non-authoritative RRs are not represented by
      NSEC3 RRs.

   o Each empty non-terminal MUST have a corresponding NSEC3 RR, unless
      the empty non-terminal is only derived from an insecure delegation
      covered by an Opt-Out NSEC3 RR.

   o The TTL value for any NSEC3 RR SHOULD be the same as the minimum
      TTL value field in the zone SOA RR.

   o The Type Bit Maps field of every NSEC3 RR in a signed zone MUST
      indicate the presence of all types present at the original owner
      name, except for the types solely contributed by an NSEC3 RR
      itself. Note that this means that the NSEC3 type itself will
      never be present in the Type Bit Maps.

   The following steps describe a method of proper construction of NSEC3
   RRs. This is not the only such possible method.

   1. Select the hash algorithm and the values for salt and iterations.

   2. For each unique original owner name in the zone add an NSEC3 RR.

       * If Opt-Out is being used, owner names of unsigned delegations
          MAY be excluded.

       * The owner name of the NSEC3 RR is the hash of the original
          owner name, prepended as a single label to the zone name.

       * The Next Hashed Owner Name field is left blank for the moment.

       * If Opt-Out is being used, set the Opt-Out bit to one.

       * For collision detection purposes, optionally keep track of the
          original owner name with the NSEC3 RR.

      * Additionally, for collision detection purposes, optionally
         create an additional NSEC3 RR corresponding to the original
         owner name with the asterisk label prepended (i.e., as if a
         wildcard existed as a child of this owner name) and keep track
         of this original owner name. Mark this NSEC3 RR as temporary.

   3. For each RRSet at the original owner name, set the corresponding
       bit in the Type Bit Maps field.

   4. If the difference in number of labels between the apex and the
       original owner name is greater than 1, additional NSEC3 RRs need
       to be added for every empty non-terminal between the apex and the
       original owner name. This process may generate NSEC3 RRs with
       duplicate hashed owner names. Optionally, for collision
       detection, track the original owner names of these NSEC3 RRs and
       create temporary NSEC3 RRs for wildcard collisions in a similar
       fashion to step 1.

   5. Sort the set of NSEC3 RRs into hash order.

   6. Combine NSEC3 RRs with identical hashed owner names by replacing
       them with a single NSEC3 RR with the Type Bit Maps field
       consisting of the union of the types represented by the set of
       NSEC3 RRs. If the original owner name was tracked, then
       collisions may be detected when combining, as all of the matching
       NSEC3 RRs should have the same original owner name. Discard any
       possible temporary NSEC3 RRs.

   7. In each NSEC3 RR, insert the next hashed owner name by using the
       value of the next NSEC3 RR in hash order. The next hashed owner
       name of the last NSEC3 RR in the zone contains the value of the
       hashed owner name of the first NSEC3 RR in the hash order.

   8. Finally, add an NSEC3PARAM RR with the same Hash Algorithm,
       Iterations, and Salt fields to the zone apex.

   If a hash collision is detected, then a new salt has to be chosen,
   and the signing process restarted.

7.2. Zone Serving

   This specification modifies DNSSEC-enabled DNS responses generated by
   authoritative servers. In particular, it replaces the use of NSEC
   RRs in such responses with NSEC3 RRs.

In the following response cases, the NSEC RRs dictated by DNSSEC
[RFC4035] are replaced with NSEC3 RRs that prove the same facts.
Responses that would not contain NSEC RRs are unchanged by this
specification.

When returning responses containing multiple NSEC3 RRs, all of the
NSEC3 RRs MUST use the same hash algorithm, iteration, and salt
values. The Flags field value MUST be either zero or one.

7.2.1. Closest Encloser Proof

For many NSEC3 responses a proof of the closest encloser is required.
This is a proof that some ancestor of the QNAME is the closest
encloser of QNAME.

This proof consists of (up to) two different NSEC3 RRs:

o An NSEC3 RR that matches the closest (provable) encloser.

o An NSEC3 RR that covers the "next closer" name to the closest
   encloser.

The first NSEC3 RR essentially proposes a possible closest encloser,
and proves that the particular encloser does, in fact, exist. The
second NSEC3 RR proves that the possible closest encloser is the
closest, and proves that the QNAME (and any ancestors between QNAME
and the closest encloser) does not exist.

These NSEC3 RRs are collectively referred to as the "closest encloser
proof" in the subsequent descriptions.

For example, the closest encloser proof for the nonexistent
"alpha.beta.gamma.example." owner name might prove that
"gamma.example." is the closest encloser. This response would
contain the NSEC3 RR that matches "gamma.example.", and would also
contain the NSEC3 RR that covers "beta.gamma.example." (which is the
"next closer" name).

It is possible, when using Opt-Out (Section 6), to not be able to
prove the actual closest encloser because it is, or is part of an
insecure delegation covered by an Opt-Out span. In this case,
instead of proving the actual closest encloser, the closest provable
encloser is used. That is, the closest enclosing authoritative name
is used instead. In this case, the set of NSEC3 RRs used for this
proof is referred to as the "closest provable encloser proof".

7.2.2. Name Error Responses

   To prove the nonexistence of QNAME, a closest encloser proof and an
   NSEC3 RR covering the (nonexistent) wildcard RR at the closest
   encloser MUST be included in the response. This collection of (up
   to) three NSEC3 RRs proves both that QNAME does not exist and that a
   wildcard that could have matched QNAME also does not exist.

   For example, if "gamma.example." is the closest provable encloser to
   QNAME, then an NSEC3 RR covering "*.gamma.example." is included in
   the authority section of the response.

7.2.3. No Data Responses, QTYPE is not DS

   The server MUST include the NSEC3 RR that matches QNAME. This NSEC3
   RR MUST NOT have the bits corresponding to either the QTYPE or CNAME
   set in its Type Bit Maps field.

7.2.4. No Data Responses, QTYPE is DS

   If there is an NSEC3 RR that matches QNAME, the server MUST return it
   in the response. The bits corresponding with DS and CNAME MUST NOT
   be set in the Type Bit Maps field of this NSEC3 RR.

   If no NSEC3 RR matches QNAME, the server MUST return a closest
   provable encloser proof for QNAME. The NSEC3 RR that covers the
   "next closer" name MUST have the Opt-Out bit set (note that this is
   true by definition -- if the Opt-Out bit is not set, something has
   gone wrong).

   If a server is authoritative for both sides of a zone cut at QNAME,
   the server MUST return the proof from the parent side of the zone
   cut.

7.2.5. Wildcard No Data Responses

   If there is a wildcard match for QNAME, but QTYPE is not present at
   that name, the response MUST include a closest encloser proof for
   QNAME and MUST include the NSEC3 RR that matches the wildcard. This
   combination proves both that QNAME itself does not exist and that a
   wildcard that matches QNAME does exist. Note that the closest
   encloser to QNAME MUST be the immediate ancestor of the wildcard RR
   (if this is not the case, then something has gone wrong).

7.2.6. Wildcard Answer Responses

   If there is a wildcard match for QNAME and QTYPE, then, in addition
   to the expanded wildcard RRSet returned in the answer section of the
   response, proof that the wildcard match was valid must be returned.

   This proof is accomplished by proving that both QNAME does not exist
   and that the closest encloser of the QNAME and the immediate ancestor
   of the wildcard are the same (i.e., the correct wildcard matched).

   To this end, the NSEC3 RR that covers the "next closer" name of the
   immediate ancestor of the wildcard MUST be returned. It is not
   necessary to return an NSEC3 RR that matches the closest encloser, as
   the existence of this closest encloser is proven by the presence of
   the expanded wildcard in the response.

7.2.7. Referrals to Unsigned Subzones

   If there is an NSEC3 RR that matches the delegation name, then that
   NSEC3 RR MUST be included in the response. The DS bit in the type
   bit maps of the NSEC3 RR MUST NOT be set.

   If the zone is Opt-Out, then there may not be an NSEC3 RR
   corresponding to the delegation. In this case, the closest provable
   encloser proof MUST be included in the response. The included NSEC3
   RR that covers the "next closer" name for the delegation MUST have
   the Opt-Out flag set to one. (Note that this will be the case unless
   something has gone wrong).

7.2.8. Responding to Queries for NSEC3 Owner Names

   The owner names of NSEC3 RRs are not represented in the NSEC3 RR
   chain like other owner names. As a result, each NSEC3 owner name is
   covered by another NSEC3 RR, effectively negating the existence of
   the NSEC3 RR. This is a paradox, since the existence of an NSEC3 RR
   can be proven by its RRSIG RRSet.

   If the following conditions are all true:

   o the QNAME equals the owner name of an existing NSEC3 RR, and

   o no RR types exist at the QNAME, nor at any descendant of QNAME,

   then the response MUST be constructed as a Name Error response
   (Section 7.2.2). Or, in other words, the authoritative name server
   will act as if the owner name of the NSEC3 RR did not exist.

Note that NSEC3 RRs are returned as a result of an AXFR or IXFR
query.

7.2.9. Server Response to a Run-Time Collision

If the hash of a non-existing QNAME collides with the owner name of
an existing NSEC3 RR, then the server will be unable to return a
response that proves that QNAME does not exist. In this case, the
server MUST return a response with an RCODE of 2 (server failure).

Note that with the hash algorithm specified in this document, SHA-1,
such collisions are highly unlikely.

7.3. Secondary Servers

Secondary servers (and perhaps other entities) need to reliably
determine which NSEC3 parameters (i.e., hash, salt, and iterations)
are present at every hashed owner name, in order to be able to choose
an appropriate set of NSEC3 RRs for negative responses. This is
indicated by an NSEC3PARAM RR present at the zone apex.

If there are multiple NSEC3PARAM RRs present, there are multiple
valid NSEC3 chains present. The server must choose one of them, but
may use any criteria to do so.

7.4. Zones Using Unknown Hash Algorithms

Zones that are signed according to this specification, but are using
an unrecognized NSEC3 hash algorithm value, cannot be effectively
served. Such zones SHOULD be rejected when loading. Servers SHOULD
respond with RCODE=2 (server failure) responses when handling queries
that would fall under such zones.

7.5. Dynamic Update

A zone signed using NSEC3 may accept dynamic updates [RFC2136].
However, NSEC3 introduces some special considerations for dynamic
updates.

Adding and removing names in a zone MUST account for the creation or
removal of empty non-terminals.

o When removing a name with a corresponding NSEC3 RR, any NSEC3 RRs
   corresponding to empty non-terminals created by that name MUST be
   removed. Note that more than one name may be asserting the
   existence of a particular empty non-terminal.

   o When adding a name that requires adding an NSEC3 RR, NSEC3 RRs
      MUST also be added for any empty non-terminals that are created.
      That is, if there is not an existing NSEC3 RR matching an empty
      non-terminal, it must be created and added.

   The presence of Opt-Out in a zone means that some additions or
   delegations of names will not require changes to the NSEC3 RRs in a
   zone.

   o When removing a delegation RRSet, if that delegation does not have
      a matching NSEC3 RR, then it was opted out. In this case, nothing
      further needs to be done.

   o When adding a delegation RRSet, if the "next closer" name of the
      delegation is covered by an existing Opt-Out NSEC3 RR, then the
      delegation MAY be added without modifying the NSEC3 RRs in the
      zone.

   The presence of Opt-Out in a zone means that when adding or removing
   NSEC3 RRs, the value of the Opt-Out flag that should be set in new or
   modified NSEC3 RRs is ambiguous. Servers SHOULD follow this set of
   basic rules to resolve the ambiguity.

   The central concept to these rules is that the state of the Opt-Out
   flag of the covering NSEC3 RR is preserved.

   o When removing an NSEC3 RR, the value of the Opt-Out flag for the
      previous NSEC3 RR (the one whose next hashed owner name is
      modified) should not be changed.

   o When adding an NSEC3 RR, the value of the Opt-Out flag is set to
      the value of the Opt-Out flag of the NSEC3 RR that previously
      covered the owner name of the NSEC3 RR. That is, the now previous
      NSEC3 RR.

   If the zone in question is consistent with its use of the Opt-Out
   flag (that is, all NSEC3 RRs in the zone have the same value for the
   flag) then these rules will retain that consistency. If the zone is
   not consistent in the use of the flag (i.e., a partially Opt-Out
   zone), then these rules will not retain the same pattern of use of
   the Opt-Out flag.

   For zones that partially use the Opt-Out flag, if there is a logical
   pattern for that use, the pattern could be maintained by using a
   local policy on the server.

8. Validator Considerations

8.1. Responses with Unknown Hash Types

   A validator MUST ignore NSEC3 RRs with unknown hash types. The
   practical result of this is that responses containing only such NSEC3
   RRs will generally be considered bogus.

8.2. Verifying NSEC3 RRs

   A validator MUST ignore NSEC3 RRs with a Flag fields value other than
   zero or one.

   A validator MAY treat a response as bogus if the response contains
   NSEC3 RRs that contain different values for hash algorithm,
   iterations, or salt from each other for that zone.

8.3. Closest Encloser Proof

   In order to verify a closest encloser proof, the validator MUST find
   the longest name, X, such that

   o X is an ancestor of QNAME that is matched by an NSEC3 RR present
      in the response. This is a candidate for the closest encloser,
      and

   o The name one label longer than X (but still an ancestor of -- or
      equal to -- QNAME) is covered by an NSEC3 RR present in the
      response.

   One possible algorithm for verifying this proof is as follows:

   1. Set SNAME=QNAME. Clear the flag.

   2. Check whether SNAME exists:

       * If there is no NSEC3 RR in the response that matches SNAME
          (i.e., an NSEC3 RR whose owner name is the same as the hash of
          SNAME, prepended as a single label to the zone name), clear
          the flag.

       * If there is an NSEC3 RR in the response that covers SNAME, set
          the flag.

       * If there is a matching NSEC3 RR in the response and the flag
          was set, then the proof is complete, and SNAME is the closest
          encloser.

          * If there is a matching NSEC3 RR in the response, but the flag
            is not set, then the response is bogus.

   3. Truncate SNAME by one label from the left, go to step 2.

Once the closest encloser has been discovered, the validator MUST
check that the NSEC3 RR that has the closest encloser as the original
owner name is from the proper zone. The DNAME type bit must not be
set and the NS type bit may only be set if the SOA type bit is set.
If this is not the case, it would be an indication that an attacker
is using them to falsely deny the existence of RRs for which the
server is not authoritative.

In the following descriptions, the phrase "a closest (provable)
encloser proof for X" means that the algorithm above (or an
equivalent algorithm) proves that X does not exist by proving that an
ancestor of X is its closest encloser.

8.4. Validating Name Error Responses

A validator MUST verify that there is a closest encloser proof for
QNAME present in the response and that there is an NSEC3 RR that
covers the wildcard at the closest encloser (i.e., the name formed by
prepending the asterisk label to the closest encloser).

8.5. Validating No Data Responses, QTYPE is not DS

The validator MUST verify that an NSEC3 RR that matches QNAME is
present and that both the QTYPE and the CNAME type are not set in its
Type Bit Maps field.

Note that this test also covers the case where the NSEC3 RR exists
because it corresponds to an empty non-terminal, in which case the
NSEC3 RR will have an empty Type Bit Maps field.

8.6. Validating No Data Responses, QTYPE is DS

If there is an NSEC3 RR that matches QNAME present in the response,
then that NSEC3 RR MUST NOT have the bits corresponding to DS and
CNAME set in its Type Bit Maps field.

If there is no such NSEC3 RR, then the validator MUST verify that a
closest provable encloser proof for QNAME is present in the response,
and that the NSEC3 RR that covers the "next closer" name has the Opt-
Out bit set.

8.7. Validating Wildcard No Data Responses

   The validator MUST verify a closest encloser proof for QNAME and MUST
   find an NSEC3 RR present in the response that matches the wildcard
   name generated by prepending the asterisk label to the closest
   encloser. Furthermore, the bits corresponding to both QTYPE and
   CNAME MUST NOT be set in the wildcard matching NSEC3 RR.

8.8. Validating Wildcard Answer Responses

   The verified wildcard answer RRSet in the response provides the
   validator with a (candidate) closest encloser for QNAME. This
   closest encloser is the immediate ancestor to the generating
   wildcard.

   Validators MUST verify that there is an NSEC3 RR that covers the
   "next closer" name to QNAME present in the response. This proves
   that QNAME itself did not exist and that the correct wildcard was
   used to generate the response.

8.9. Validating Referrals to Unsigned Subzones

   The delegation name in a referral is the owner name of the NS RRSet
   present in the authority section of the referral response.

   If there is an NSEC3 RR present in the response that matches the
   delegation name, then the validator MUST ensure that the NS bit is
   set and that the DS bit is not set in the Type Bit Maps field of the
   NSEC3 RR. The validator MUST also ensure that the NSEC3 RR is from
   the correct (i.e., parent) zone. This is done by ensuring that the
   SOA bit is not set in the Type Bit Maps field of this NSEC3 RR.

   Note that the presence of an NS bit implies the absence of a DNAME
   bit, so there is no need to check for the DNAME bit in the Type Bit
   Maps field of the NSEC3 RR.

   If there is no NSEC3 RR present that matches the delegation name,
   then the validator MUST verify a closest provable encloser proof for
   the delegation name. The validator MUST verify that the Opt-Out bit
   is set in the NSEC3 RR that covers the "next closer" name to the
   delegation name.

9. Resolver Considerations

9.1. NSEC3 Resource Record Caching

   Caching resolvers MUST be able to retrieve the appropriate NSEC3 RRs
   when returning responses that contain them. In DNSSEC [RFC4035], in

many cases it is possible to find the correct NSEC RR to return in a
response by name (e.g., when returning a referral, the NSEC RR will
always have the same owner name as the delegation). With this
specification, that will not be true, nor will a cache be able to
calculate the name(s) of the appropriate NSEC3 RR(s).
Implementations may need to use new methods for caching and
retrieving NSEC3 RRs.

9.2. Use of the AD Bit

The AD bit, as defined by [RFC4035], MUST NOT be set when returning a
response containing a closest (provable) encloser proof in which the
NSEC3 RR that covers the "next closer" name has the Opt-Out bit set.

This rule is based on what this closest encloser proof actually
proves: names that would be covered by the Opt-Out NSEC3 RR may or
may not exist as insecure delegations. As such, not all the data in
responses containing such closest encloser proofs will have been
cryptographically verified, so the AD bit cannot be set.

10. Special Considerations

10.1. Domain Name Length Restrictions

Zones signed using this specification have additional domain name
length restrictions imposed upon them. In particular, zones with
names that, when converted into hashed owner names exceed the 255
octet length limit imposed by [RFC1035], cannot use this
specification.

The actual maximum length of a domain name in a particular zone
depends on both the length of the zone name (versus the whole domain
name) and the particular hash function used.

As an example, SHA-1 produces a hash of 160 bits. The base-32
encoding of 160 bits results in 32 characters. The 32 characters are
prepended to the name of the zone as a single label, which includes a
length field of a single octet. The maximum length of the zone name,
when using SHA-1, is 222 octets (255 - 33).

10.2. DNAME at the Zone Apex

The DNAME specification in Section 3 of [RFC2672] has a 'no-
descendants' limitation. If a DNAME RR is present at node N, there
MUST be no data at any descendant of N.

If N is the apex of the zone, there will be NSEC3 and RRSIG types
present at descendants of N. This specification updates the DNAME

   specification to allow NSEC3 and RRSIG types at descendants of the
   apex regardless of the existence of DNAME at the apex.

10.3. Iterations

   Setting the number of iterations used allows the zone owner to choose
   the cost of computing a hash, and therefore the cost of generating a
   dictionary. Note that this is distinct from the effect of salt,
   which prevents the use of a single precomputed dictionary for all
   time.

   Obviously the number of iterations also affects the zone owner's cost
   of signing and serving the zone as well as the validator's cost of
   verifying responses from the zone. We therefore impose an upper
   limit on the number of iterations. We base this on the number of
   iterations that approximates the cost of verifying an RRSet.

   The limits, therefore, are based on the size of the smallest zone
   signing key, rounded up to the nearest table value (or rounded down
   if the key is larger than the largest table value).

   A zone owner MUST NOT use a value higher than shown in the table
   below for iterations for the given key size. A resolver MAY treat a
   response with a higher value as insecure, after the validator has
   verified that the signature over the NSEC3 RR is correct.

              +----------+------------+
              | Key Size | Iterations |
              +----------+------------+
              | 1024 | 150 |
              | 2048 | 500 |
              | 4096 | 2,500 |
              +----------+------------+

   This table is based on an approximation of the ratio between the cost
   of an SHA-1 calculation and the cost of an RSA verification for keys
   of size 1024 bits (150 to 1), 2048 bits (500 to 1), and 4096 bits
   (2500 to 1).

   The ratio between SHA-1 calculation and DSA verification is higher
   (1500 to 1 for keys of size 1024). A higher iteration count degrades
   performance, while DSA verification is already more expensive than
   RSA for the same key size. Therefore the values in the table MUST be
   used independent of the key algorithm.

10.4. Transitioning a Signed Zone from NSEC to NSEC3

   When transitioning an already signed and trusted zone to this
   specification, care must be taken to prevent client validation
   failures during the process.

   The basic procedure is as follows:

   1. Transition all DNSKEYs to DNSKEYs using the algorithm aliases
       described in Section 2. The actual method for safely and
       securely changing the DNSKEY RRSet of the zone is outside the
       scope of this specification. However, the end result MUST be
       that all DS RRs in the parent use the specified algorithm
       aliases.

       After this transition is complete, all NSEC3-unaware clients will
       treat the zone as insecure. At this point, the authoritative
       server still returns negative and wildcard responses that contain
       NSEC RRs.

   2. Add signed NSEC3 RRs to the zone, either incrementally or all at
       once. If adding incrementally, then the last RRSet added MUST be
       the NSEC3PARAM RRSet.

   3. Upon the addition of the NSEC3PARAM RRSet, the server switches to
       serving negative and wildcard responses with NSEC3 RRs according
       to this specification.

   4. Remove the NSEC RRs either incrementally or all at once.

10.5. Transitioning a Signed Zone from NSEC3 to NSEC

   To safely transition back to a DNSSEC [RFC4035] signed zone, simply
   reverse the procedure above:

   1. Add NSEC RRs incrementally or all at once.

   2. Remove the NSEC3PARAM RRSet. This will signal the server to use
       the NSEC RRs for negative and wildcard responses.

   3. Remove the NSEC3 RRs either incrementally or all at once.

   4. Transition all of the DNSKEYs to DNSSEC algorithm identifiers.
       After this transition is complete, all NSEC3-unaware clients will
       treat the zone as secure.

11. IANA Considerations

   Although the NSEC3 and NSEC3PARAM RR formats include a hash algorithm
   parameter, this document does not define a particular mechanism for
   safely transitioning from one NSEC3 hash algorithm to another. When
   specifying a new hash algorithm for use with NSEC3, a transition
   mechanism MUST also be defined.

   This document updates the IANA registry "DOMAIN NAME SYSTEM
   PARAMETERS" (http://www.iana.org/assignments/dns-parameters) in sub-
   registry "TYPES", by defining two new types. Section 3 defines the
   NSEC3 RR type 50. Section 4 defines the NSEC3PARAM RR type 51.

   This document updates the IANA registry "DNS SECURITY ALGORITHM
   NUMBERS -- per [RFC4035]"
   (http://www.iana.org/assignments/dns-sec-alg-numbers). Section 2
   defines the aliases DSA-NSEC3-SHA1 (6) and RSASHA1-NSEC3-SHA1 (7) for
   respectively existing registrations DSA and RSASHA1 in combination
   with NSEC3 hash algorithm SHA1.

   Since these algorithm numbers are aliases for existing DNSKEY
   algorithm numbers, the flags that exist for the original algorithm
   are valid for the alias algorithm.

   This document creates a new IANA registry for NSEC3 flags. This
   registry is named "DNSSEC NSEC3 Flags". The initial contents of this
   registry are:

    0   1   2   3   4   5   6   7
   +---+---+---+---+---+---+---+---+
   | | | | | | | |Opt|
   | | | | | | | |Out|
   +---+---+---+---+---+---+---+---+

      bit 7 is the Opt-Out flag.

      bits 0 - 6 are available for assignment.

   Assignment of additional NSEC3 Flags in this registry requires IETF
   Standards Action [RFC2434].

   This document creates a new IANA registry for NSEC3PARAM flags. This
   registry is named "DNSSEC NSEC3PARAM Flags". The initial contents of
   this registry are:

```
      0   1   2   3   4   5   6   7
    +---+---+---+---+---+---+---+---+
    |   |   |   |   |   |   |   | 0 |
    +---+---+---+---+---+---+---+---+
```

    bit 7 is reserved and must be 0.

    bits 0 - 6 are available for assignment.

Assignment of additional NSEC3PARAM Flags in this registry requires
IETF Standards Action [RFC2434].

Finally, this document creates a new IANA registry for NSEC3 hash
algorithms. This registry is named "DNSSEC NSEC3 Hash Algorithms".
The initial contents of this registry are:

    0 is Reserved.

    1 is SHA-1.

    2-255 Available for assignment.

Assignment of additional NSEC3 hash algorithms in this registry
requires IETF Standards Action [RFC2434].

12. Security Considerations

12.1. Hashing Considerations

12.1.1. Dictionary Attacks

The NSEC3 RRs are still susceptible to dictionary attacks (i.e., the
attacker retrieves all the NSEC3 RRs, then calculates the hashes of
all likely domain names, comparing against the hashes found in the
NSEC3 RRs, and thus enumerating the zone). These are substantially
more expensive than enumerating the original NSEC RRs would have
been, and in any case, such an attack could also be used directly
against the name server itself by performing queries for all likely
names, though this would obviously be more detectable. The expense
of this off-line attack can be chosen by setting the number of
iterations in the NSEC3 RR.

Zones are also susceptible to a pre-calculated dictionary attack --
that is, a list of hashes for all likely names is computed once, then
NSEC3 RR is scanned periodically and compared against the precomputed
hashes. This attack is prevented by changing the salt on a regular
basis.

The salt SHOULD be at least 64 bits long and unpredictable, so that
an attacker cannot anticipate the value of the salt and compute the
next set of dictionaries before the zone is published.

12.1.2. Collisions

Hash collisions between QNAME and the owner name of an NSEC3 RR may
occur. When they do, it will be impossible to prove the non-
existence of the colliding QNAME. However, with SHA-1, this is
highly unlikely (on the order of 1 in 2^160). Note that DNSSEC
already relies on the presumption that a cryptographic hash function
is second pre-image resistant, since these hash functions are used
for generating and validating signatures and DS RRs.

12.1.3. Transitioning to a New Hash Algorithm

Although the NSEC3 and NSEC3PARAM RR formats include a hash algorithm
parameter, this document does not define a particular mechanism for
safely transitioning from one NSEC3 hash algorithm to another. When
specifying a new hash algorithm for use with NSEC3, a transition
mechanism MUST also be defined. It is possible that the only
practical and palatable transition mechanisms may require an
intermediate transition to an insecure state, or to a state that uses
NSEC records instead of NSEC3.

12.1.4. Using High Iteration Values

Since validators should treat responses containing NSEC3 RRs with
high iteration values as insecure, presence of just one signed NSEC3
RR with a high iteration value in a zone provides attackers with a
possible downgrade attack.

The attack is simply to remove any existing NSEC3 RRs from a
response, and replace or add a single (or multiple) NSEC3 RR that
uses a high iterations value to the response. Validators will then
be forced to treat the response as insecure. This attack would be
effective only when all of following conditions are met:

o There is at least one signed NSEC3 RR that uses a high iterations
   value present in the zone.

o The attacker has access to one or more of these NSEC3 RRs. This
   is trivially true when the NSEC3 RRs with high iteration values
   are being returned in typical responses, but may also be true if
   the attacker can access the zone via AXFR or IXFR queries, or any
   other methodology.

Using a high number of iterations also introduces an additional
denial-of-service opportunity against servers, since servers must
calculate several hashes per negative or wildcard response.

12.2. Opt-Out Considerations

The Opt-Out Flag (O) allows for unsigned names, in the form of
delegations to unsigned zones, to exist within an otherwise signed
zone. All unsigned names are, by definition, insecure, and their
validity or existence cannot be cryptographically proven.

In general:

o Resource records with unsigned names (whether existing or not)
   suffer from the same vulnerabilities as RRs in an unsigned zone.
   These vulnerabilities are described in more detail in [RFC3833]
   (note in particular Section 2.3, "Name Chaining" and Section 2.6,
   "Authenticated Denial of Domain Names").

o Resource records with signed names have the same security whether
   or not Opt-Out is used.

Note that with or without Opt-Out, an insecure delegation may be
undetectably altered by an attacker. Because of this, the primary
difference in security when using Opt-Out is the loss of the ability
to prove the existence or nonexistence of an insecure delegation
within the span of an Opt-Out NSEC3 RR.

In particular, this means that a malicious entity may be able to
insert or delete RRs with unsigned names. These RRs are normally NS
RRs, but this also includes signed wildcard expansions (while the
wildcard RR itself is signed, its expanded name is an unsigned name).

Note that being able to add a delegation is functionally equivalent
to being able to add any RR type: an attacker merely has to forge a
delegation to name server under his/her control and place whatever
RRs needed at the subzone apex.

While in particular cases, this issue may not present a significant
security problem, in general it should not be lightly dismissed.
Therefore, it is strongly RECOMMENDED that Opt-Out be used sparingly.
In particular, zone signing tools SHOULD NOT default to using Opt-
Out, and MAY choose to not support Opt-Out at all.

12.3. Other Considerations

   Walking the NSEC3 RRs will reveal the total number of RRs in the zone
   (plus empty non-terminals), and also what types there are. This
   could be mitigated by adding dummy entries, but certainly an upper
   limit can always be found.

13. References

13.1. 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.

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

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

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

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

   [RFC2434]   Narten, T. and H. Alvestrand, "Guidelines for
                     Writing an IANA Considerations Section in RFCs",
                     BCP 26, RFC 2434, October 1998.

   [RFC2929]   Eastlake, D., Brunner-Williams, E., and B. Manning,
                     "Domain Name System (DNS) IANA Considerations",
                     BCP 42, RFC 2929, September 2000.

   [RFC3597]   Gustafsson, A., "Handling of Unknown DNS Resource
                     Record (RR) Types", RFC 3597, September 2003.

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

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

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

   [RFC4648]   Josefsson, S., "The Base16, Base32, and Base64 Data
                        Encodings", RFC 4648, October 2006.

13.2. Informative References

   [DNSEXT-NO]   Josefsson, S., "Authenticating Denial of Existence
                        in DNS with Minimum Disclosure", Work in Progress,
                        July 2000.

   [DNSEXT-NSEC2v2]   Laurie, B., "DNSSEC NSEC2 Owner and RDATA Format",
                        Work in Progress, December 2004.

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

   [RFC2898]   Kaliski, B., "PKCS #5: Password-Based Cryptography
                        Specification Version 2.0", RFC 2898,
                        September 2000.

   [RFC3833]   Atkins, D. and R. Austein, "Threat Analysis of the
                        Domain Name System (DNS)", RFC 3833, August 2004.

   [RFC4592]   Lewis, E., "The Role of Wildcards in the Domain
                        Name System", RFC 4592, July 2006.

   [RFC4956]   Arends, R., Kosters, M., and D. Blacka, "DNS
                        Security (DNSSEC) Opt-In", RFC 4956, July 2007.

Appendix A. Example Zone

   This is a zone showing its NSEC3 RRs. They can also be used as test
   vectors for the hash algorithm.

   The overall TTL and class are specified in the SOA RR, and are
   subsequently omitted for clarity.

   The zone is preceded by a list that contains the hashes of the
   original ownernames.

   ; H(example)   = 0p9mhaveqvm6t7vbl5lop2u3t2rp3tom
   ; H(a.example)   = 35mthgpgcu1qg68fab165klnsnk3dpvl
   ; H(ai.example)   = gjeqe526plbf1g8mklp59enfd789njgi
   ; H(ns1.example)   = 2t7b4g4vsa5smi47k61mv5bv1a22bojr
   ; H(ns2.example)   = q04jkcevqvmu85r014c7dkba38o0ji5r
   ; H(w.example)   = k8udemvp1j2f7eg6jebps17vp3n8i58h
   ; H(*.w.example)   = r53bq7cc2uvmubfu5ocmm6pers9tk9en
   ; H(x.w.example)   = b4um86eghhds6nea196smvmlo4ors995
   ; H(y.w.example)   = ji6neoaepv8b5o6k4ev33abha8ht9fgc
   ; H(x.y.w.example) = 2vptu5timamqttgl4luu9kg21e0aor3s
   ; H(xx.example)   = t644ebqk9bibcna874givr6joj62mlhv
   ; H(2t7b4g4vsa5smi47k61mv5bv1a22bojr.example)
   ;   = kohar7mbb8dc2ce8a9qvl8hon4k53uhi
   example. 3600 IN SOA ns1.example. bugs.x.w.example. 1 3600 300 (
                          3600000 3600)
                  RRSIG   SOA 7 1 3600 20150420235959 20051021000000 (
                          40430 example.
                          Hu25UIyNPmvPIVBrldN+9Mlp9Zql39qaUd8i
                          q4ZLlYWfUUbbAS41pG+68z81q1xhkYAcEyHd
                          VI2LmKusbZsT0Q==)
                  NS   ns1.example.
                  NS   ns2.example.
                  RRSIG   NS 7 1 3600 20150420235959 20051021000000 (
                          40430 example.
                          PVOgtMK1HHeSTau+HwDWC8Ts+6C8qtqd4pQJ
                          qOtdEVgg+MA+ai4fWDEhu3qHJyLcQ9tbD2vv
                          CnMXjtz6SyObxA==)
                  MX   1 xx.example.
                  RRSIG   MX 7 1 3600 20150420235959 20051021000000 (
                          40430 example.
                          GgQ1A9xs47k42VPvpL/a1BWUz/6XsnHkjotw
                          9So8MQtZtl2wJBsnOQsaoHrRCrRbyriEl/GZ
                          n9Mto/Kx+wBo+w==)
                  DNSKEY 256 3 7 AwEAAaetidLzsKWUt4swWR8yu0wPHPiUi8LU (
                          sAD0QPWU+wzt89epO6tHzkMBVDkC7qphQO2h
                          TY4hHn9npWFRw5BYubE=)

```
                DNSKEY  257 3 7 AwEAAcUlFV1vhmqx6NSOUOq2R/dsR7Xm3upJ (
                        j7IommWSpJABVfW8Q0rOvXdM6kzt+TAu92L9
                        AbsUdblMFin8CVF3n4s= )
                RRSIG   DNSKEY 7 1 3600 20150420235959 (
                        20051021000000 12708 example.
                        AuU4juU9RaxescSmStrQks3Gh9FblGBlVU31
                        uzMZ/U/FpsUb8aC6QZS+sTsJXnLnz7flGOsm
                        MGQZf3bH+QsCtg== )
                NSEC3PARAM 1 0 12 aabbccdd
                RRSIG   NSEC3PARAM 7 1 3600 20150420235959 (
                        20051021000000 40430 example.
                        C1Gl8tPZNtnjlrYWDeeUV/sGLCyy/IHie2re
                        rN05XSA3Pq0U3+4VvGWYWdUMfflOdxqnXHwJ
                        TLQsjlkynhG6Cg== )
0p9mhaveqvm6t7vbl5lop2u3t2rp3tom.example. NSEC3 1 1 12 aabbccdd (
                        2t7b4g4vsa5smi47k61mv5bv1a22bojr MX DNSKEY NS
                        SOA NSEC3PARAM RRSIG )
                RRSIG   NSEC3 7 2 3600 20150420235959 20051021000000 (
                        40430 example.
                        OSgWSm26B+cS+dDL8b5QrWr/dEWhtCsKlwKL
                        IBHYH6blRxK9rC0bMJPwQ4mLIuw85H2EY762
                        BOCXJZMnpuwhpA== )
2t7b4g4vsa5smi47k61mv5bv1a22bojr.example. A 192.0.2.127
                RRSIG   A 7 2 3600 20150420235959 20051021000000 (
                        40430 example.
                        h6c++bzhRuWWt2bykN6mjaTNBcXNq5UuL5Ed
                        K+iDP4eY8I0kSiKaCjg3tC1SQkeloMeub2GW
                        k8p6xHMPZumXlw== )
                NSEC3   1 1 12 aabbccdd (
                        2vptu5timamqttgl4luu9kg21e0aor3s A RRSIG )
                RRSIG   NSEC3 7 2 3600 20150420235959 20051021000000 (
                        40430 example.
                        OmBvJ1Vgg1hCKMXHFiNeIYHK9XVW0iLDLwJN
                        4TFoNxZuP03gAXEI634YwOc4YBNITrj413iq
                        NI6mRk/r1dOSUw== )
2vptu5timamqttgl4luu9kg21e0aor3s.example. NSEC3 1 1 12 aabbccdd (
                        35mthgpgcu1qg68fab165klnsnk3dpvl MX RRSIG )
                RRSIG   NSEC3 7 2 3600 20150420235959 20051021000000 (
                        40430 example.
                        KL1V2oFYghNV0Hm7Tf2vpJjM6l+0g1JCcVYG
                        VfI0lKrhPmTsOA96cLEACgo1x8I7kApJX+ob
                        TuktZ+sdsZPY1w== )
35mthgpgcu1qg68fab165klnsnk3dpvl.example. NSEC3 1 1 12 aabbccdd (
                        b4um86eghhds6nea196smvmlo4ors995 NS DS RRSIG )
                RRSIG   NSEC3 7 2 3600 20150420235959 20051021000000 (
                        40430 example.
                        g6jPUUpduAJKRljUsN8gB4UagAX0NxY9shwQ
                        Aynzo8EUWH+z6hEIBlUTPGj15eZll6VhQqgZ
```

```
                           XtAIR3chwgW+SA== )
   a.example.      NS     ns1.a.example.
                   NS     ns2.a.example.
                   DS     58470 5 1 (
                           3079F1593EBAD6DC121E202A8B766A6A4837206C )
                   RRSIG  DS 7 2 3600 20150420235959 20051021000000 (
                           40430 example.
                           XacFcQVHLVzdoc45EJhN616zQ4mEXtE8FzUh
                           M2KWjfy1VfRKD9r1MeVGwwoukOKgJxBPFsWo
                           o722vZ4UZ2dIdA== )
   ns1.a.example. A      192.0.2.5
   ns2.a.example. A      192.0.2.6
   ai.example.    A      192.0.2.9
                   RRSIG  A 7 2 3600 20150420235959 20051021000000 (
                           40430 example.
                           hVe+wKYMlObTRPhX0NL67GxeZfdxqr/QeR6F
                           tfdAj5+FgYxyzPEjIzvKWy00hWIl6wD3Vws+
                           rznEn8sQ64UdqA== )
                   HINFO  "KLH-10" "ITS"
                   RRSIG  HINFO 7 2 3600 20150420235959 20051021000000 (
                           40430 example.
                           Yi42uOq43eyO6qXHNvwwfFnIustWgV5urFcx
                           enkLvs6pKRh00VBjODmf3Z4nMO7IOl6nHSQ1
                           v0wLHpEZG7Xj2w== )
                   AAAA   2001:db8:0:0:0:0:f00:baa9
                   RRSIG  AAAA 7 2 3600 20150420235959 20051021000000 (
                           40430 example.
                           LcdxKaCB5bGZwPDg+3JJ4O02zoMBrjxqlf6W
                           uaHQZZfTUpb9Nf2nxFGe2XRPfR5tpJT6GdRG
                           cHueLuXkMjBArQ== )
   b4um86eghhds6nea196smvmlo4ors995.example. NSEC3 1 1 12 aabbccdd (
                           gjeqe526plbf1g8mklp59enfd789njgi MX RRSIG )
                   RRSIG  NSEC3 7 2 3600 20150420235959 20051021000000 (
                           40430 example.
                           ZkPG3M32lmoHM6pa3D6gZFGB/rhL//Bs3Omh
                           5u4m/CUiwtblEVOaAKKZd7S959OeiX43aLX3
                           pOv0TSTyiTxIZg== )
   c.example.      NS     ns1.c.example.
                   NS     ns2.c.example.
   ns1.c.example. A      192.0.2.7
   ns2.c.example. A      192.0.2.8
   gjeqe526plbf1g8mklp59enfd789njgi.example. NSEC3 1 1 12 aabbccdd (
                           ji6neoaepv8b5o6k4ev33abha8ht9fgc HINFO A AAAA
                           RRSIG )
                   RRSIG  NSEC3 7 2 3600 20150420235959 20051021000000 (
                           40430 example.
                           IVnezTJ9iqblFF97vPSmfXZ5Zozngx3KX3by
                           LTZC4QBH2dFWhf6scrGFZB980AfCxoD9qbbK
```

```
                            Dy+rdGIeRSVNyw== )
   ji6neoaepv8b5o6k4ev33abha8ht9fgc.example. NSEC3 1 1 12 aabbccdd (
                            k8udemvp1j2f7eg6jebps17vp3n8i58h )
                    RRSIG   NSEC3 7 2 3600 20150420235959 20051021000000 (
                            40430 example.
                            gPkFp1s2QDQ6wQzcg1uSebZ61W33rUBDcTj7
                            2F3kQ490fEdp7k1BUIfbcZtPbX3YCpE+sIt0
                            MpzVSKfTwx4uYA== )
   k8udemvp1j2f7eg6jebps17vp3n8i58h.example. NSEC3 1 1 12 aabbccdd (
                            kohar7mbb8dc2ce8a9qvl8hon4k53uhi )
                    RRSIG   NSEC3 7 2 3600 20150420235959 20051021000000 (
                            40430 example.
                            FtXGbvF0+wf8iWkyo73enAuVx03klN+pILBK
                            S6qCcftVtfH4yVzsEZquJ27NHR7ruxJWDNMt
                            Otx7w9WfcIg62A== )
   kohar7mbb8dc2ce8a9qvl8hon4k53uhi.example. NSEC3 1 1 12 aabbccdd (
                            q04jkcevqvmu85r014c7dkba38o0ji5r A RRSIG )
                    RRSIG   NSEC3 7 2 3600 20150420235959 20051021000000 (
                            40430 example.
                            VrDXs2uVW21N08SyQIz88zml+y4ZCInTwgDr
                            6zz43yAg+LFERjOrj3Ojct51ac7Dp4eZbf9F
                            QJazmASFKGxGXg== )
   ns1.example.     A       192.0.2.1
                    RRSIG   A 7 2 3600 20150420235959 20051021000000 (
                            40430 example.
                            bu6kx73n6XEunoVGuRfAgY7EF/AJqHy7hj0j
                            kiqJjB0dOrx3wuz9SaBeGfqWIdn/uta3SavN
                            4FRvZR9SCFHF5Q== )
   ns2.example.     A       192.0.2.2
                    RRSIG   A 7 2 3600 20150420235959 20051021000000 (
                            40430 example.
                            ktQ3TqE0CfRfki0Rb/Ip5BM0VnxelbuejCC4
                            zpLbFKA/7eD7UNAwxMgxJPtbdST+syjYSJaj
                            4IHfeX6n8vfoGA== )
   q04jkcevqvmu85r014c7dkba38o0ji5r.example. NSEC3 1 1 12 aabbccdd (
                            r53bq7cc2uvmubfu5ocmm6pers9tk9en A RRSIG )
                    RRSIG   NSEC3 7 2 3600 20150420235959 20051021000000 (
                            40430 example.
                            hV5I89b+4FHJDATp09g4bbN0R1F845CaXpL3
                            ZxlMKimoPAyqletMlEWwLfFia7sdpSzn+ZlN
                            NlkxWcLsIlMmUg== )
   r53bq7cc2uvmubfu5ocmm6pers9tk9en.example. NSEC3 1 1 12 aabbccdd (
                            t644ebqk9bibcna874givr6joj62mlhv MX RRSIG )
                    RRSIG   NSEC3 7 2 3600 20150420235959 20051021000000 (
                            40430 example.
                            aupviViruXs4bDg9rCbezzBMf9h1ZlDvbW/C
                            ZFKulIGXXLj8B/fsDJarXVDA9bnUoRhEbKp+
                            HF1FWKW7RIJdtQ== )
```

```
   t644ebqk9bibcna874givr6joj62mlhv.example. NSEC3 1 1 12 aabbccdd (
                         0p9mhaveqvm6t7vbl5lop2u3t2rp3tom HINFO A AAAA
                         RRSIG )
                RRSIG    NSEC3 7 2 3600 20150420235959 20051021000000 (
                         40430 example.
                         RAjGECB8P7O+F4Pa4Dx3tC0M+Z3KmlLKImca
                         fb9XWwx+NWUNz7NBEDBQHivIyKPVDkChcePI
                         X1xPl1ATNa+8Dw== )
   *.w.example.  MX       1 ai.example.
                RRSIG    MX 7 2 3600 20150420235959 20051021000000 (
                         40430 example.
                         CikebjQwGQPwijVcxgcZcSJKtfynugtlBiKb
                         9FcBTrmOoyQ4InoWVudhCWsh/URX3lc4WRUM
                         ivEBP6+4KS3ldA== )
   x.w.example.  MX       1 xx.example.
                RRSIG    MX 7 3 3600 20150420235959 20051021000000 (
                         40430 example.
                         IrK3tq/tHFIBF0scHiE/1IwMAvckS/55hAVv
                         QyxTFbkAdDloP3NbZzu+yoSsr3b3OX6qbBpY
                         7WCtwwekLKRAwQ== )
   x.y.w.example. MX      1 xx.example.
                RRSIG    MX 7 4 3600 20150420235959 20051021000000 (
                         40430 example.
                         MqSt5HqJIN8+SLlzTOImrh5h9Xa6gDvAW/Gn
                         nbdPc6Z7nXvCpLPJj/5lCwx3VuzVOjkbvXze
                         8/8Ccl2Zn2hbug== )
   xx.example.   A        192.0.2.10
                RRSIG    A 7 2 3600 20150420235959 20051021000000 (
                         40430 example.
                         T35hBWEZ017VC5u2c4OriKyVn/pu+fVK4AlX
                         YOxJ6iQylfV2HQIKjv6b7DzINB3aF/wjJqgX
                         pQvhq+Ac6+ZiFg== )
                HINFO    "KLH-10" "TOPS-20"
                RRSIG    HINFO 7 2 3600 20150420235959 20051021000000 (
                         40430 example.
                         KimG+rDd+7VA1zRsu0ITNAQUTRlpnsmqWrih
                         FRnU+bRa93v2e5oFNFYCs3Rqgv62K93N7AhW
                         6Jfqj/8NzWjvKg== )
                AAAA     2001:db8:0:0:0:0:f00:baaa
                RRSIG    AAAA 7 2 3600 20150420235959 20051021000000 (
                         40430 example.
                         IXBcXORITNwd8h3gNwyxtYFvAupS/CYWufVe
                         uBUX0O25ivBCULjZjpDxFSxfohb/KA7YRdxE
                         NzYfMItpILl/Xw== )
```

Appendix B. Example Responses

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

B.1. Name Error

   An authoritative name error. The NSEC3 RRs prove that the name does
   not exist and that there is no wildcard RR that should have been
   expanded.

;; Header: QR AA DO RCODE=3
;;
;; Question
a.c.x.w.example.   IN A

;; Answer
;; (empty)

;; Authority

example.   SOA   ns1.example. bugs.x.w.example. 1 3600 300 (
                       3600000 3600)
example.   RRSIG SOA 7 1 3600 20150420235959 20051021000000 (
                       40430 example.
                       Hu25UIyNPmvPIVBrldN+9Mlp9Zql39qaUd8i
                       q4ZLlYWfUUbbAS41pG+68z81q1xhkYAcEyHd
                       VI2LmKusbZsT0Q==)

;; NSEC3 RR that covers the "next closer" name (c.x.w.example)
;; H(c.x.w.example) = 0va5bpr2ou0vk0lbqeeljri88laipsfh

0p9mhaveqvm6t7vbl5lop2u3t2rp3tom.example. NSEC3 1 1 12 aabbccdd (
                       2t7b4g4vsa5smi47k61mv5bv1a22bojr MX DNSKEY NS
                       SOA NSEC3PARAM RRSIG)
0p9mhaveqvm6t7vbl5lop2u3t2rp3tom.example. RRSIG NSEC3 7 2 3600 (
                       20150420235959 20051021000000 40430 example.
                       OSgWSm26B+cS+dDL8b5QrWr/dEWhtCsKlwKL
                       IBHYH6blRxK9rC0bMJPwQ4mLIuw85H2EY762
                       BOCXJZMnpuwhpA==)

;; NSEC3 RR that matches the closest encloser (x.w.example)
;; H(x.w.example) = b4um86eghhds6nea196smvmlo4ors995

b4um86eghhds6nea196smvmlo4ors995.example. NSEC3 1 1 12 aabbccdd (
                       gjeqe526plbf1g8mklp59enfd789njgi MX RRSIG)
b4um86eghhds6nea196smvmlo4ors995.example. RRSIG NSEC3 7 2 3600 (
                       20150420235959 20051021000000 40430 example.

```
                        ZkPG3M32lmoHM6pa3D6gZFGB/rhL//Bs3Omh
                        5u4m/CUiwtblEVOaAKKZd7S959OeiX43aLX3
                        pOv0TSTyiTxIZg== )
```

;; NSEC3 RR that covers wildcard at the closest encloser (*.x.w.example)
;; H(*.x.w.example) = 92pqneegtaue7pjatc3l3qnk738c6v5m

```
35mthgpgcu1qg68fab165klnsnk3dpvl.example. NSEC3 1 1 12 aabbccdd (
                        b4um86eghhds6nea196smvmlo4ors995 NS DS RRSIG )
35mthgpgcu1qg68fab165klnsnk3dpvl.example. RRSIG NSEC3 7 2 3600 (
                        20150420235959 20051021000000 40430 example.
                        g6jPUUpduAJKRljUsN8gB4UagAX0NxY9shwQ
                        Aynzo8EUWH+z6hEIBlUTPGj15eZll6VhQqgZ
                        XtAIR3chwgW+SA== )
```

;; Additional
;; (empty)

   The query returned three NSEC3 RRs that prove that the requested data
   does not exist and that no wildcard expansion applies. The negative
   response is authenticated by verifying the NSEC3 RRs. The
   corresponding RRSIGs indicate that the NSEC3 RRs are signed by an
   "example" DNSKEY of algorithm 7 and with key tag 40430. The resolver
   needs the corresponding DNSKEY RR in order to authenticate this
   answer.

   One of the owner names of the NSEC3 RRs matches the closest encloser.
   One of the NSEC3 RRs prove that there exists no longer name. One of
   the NSEC3 RRs prove that there exists no wildcard RRSets that should
   have been expanded. The closest encloser can be found by applying
   the algorithm in Section 8.3.

   In the above example, the name 'x.w.example' hashes to
   'b4um86eghhds6nea196smvmlo4ors995'. This indicates that this might
   be the closest encloser. To prove that 'c.x.w.example' and
   '*.x.w.example' do not exist, these names are hashed to,
   respectively, '0va5bpr2ou0vk0lbqeeljri88laipsfh' and
   '92pqneegtaue7pjatc3l3qnk738c6v5m'. The first and last NSEC3 RRs
   prove that these hashed owner names do not exist.

B.2. No Data Error

   A "no data" response. The NSEC3 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.        SOA     ns1.example. bugs.x.w.example. 1 3600 300 (
                       3600000 3600 )
example.        RRSIG   SOA 7 1 3600 20150420235959 20051021000000 (
                       40430 example.
                       Hu25UIyNPmvPIVBrldN+9Mlp9Zql39qaUd8i
                       q4ZLlYWfUUbbAS41pG+68z81q1xhkYAcEyHd
                       VI2LmKusbZsT0Q== )

;; NSEC3 RR matches the QNAME and shows that the MX type bit is not set.

2t7b4g4vsa5smi47k61mv5bv1a22bojr.example. NSEC3 1 1 12 aabbccdd (
                       2vptu5timamqttgl4luu9kg21e0aor3s A RRSIG )
2t7b4g4vsa5smi47k61mv5bv1a22bojr.example. RRSIG NSEC3 7 2 3600 (
                       20150420235959 20051021000000 40430 example.
                       OmBvJ1Vgg1hCKMXHFiNeIYHK9XVW0iLDLwJN
                       4TFoNxZuP03gAXEI634YwOc4YBNITrj413iq
                       NI6mRk/r1dOSUw== )
;; Additional
;; (empty)
```

   The query returned an NSEC3 RR that proves that the requested name
   exists ("ns1.example." hashes to "2t7b4g4vsa5smi47k61mv5bv1a22bojr"),
   but the requested RR type does not exist (type MX is absent in the
   type code list of the NSEC3 RR), and was not a CNAME (type CNAME is
   also absent in the type code list of the NSEC3 RR).

B.2.1. No Data Error, Empty Non-Terminal

   A "no data" response because of an empty non-terminal. The NSEC3 RR
   proves that the name exists and that the requested RR type does not.

```
   ;; Header: QR AA DO RCODE=0
   ;;
   ;; Question
   y.w.example.         IN A

   ;; Answer
   ;; (empty)

   ;; Authority
   example.        SOA     ns1.example. bugs.x.w.example. 1 3600 300 (
                           3600000 3600 )
   example.        RRSIG   SOA 7 1 3600 20150420235959 20051021000000 (
                           40430 example.
                           Hu25UIyNPmvPIVBrldN+9Mlp9Zql39qaUd8i
                           q4ZLlYWfUUbbAS41pG+68z81q1xhkYAcEyHd
                           VI2LmKusbZsT0Q== )

   ;; NSEC3 RR matches the QNAME and shows that the A type bit is not set.

   ji6neoaepv8b5o6k4ev33abha8ht9fgc.example. NSEC3 1 1 12 aabbccdd (
                           k8udemvp1j2f7eg6jebps17vp3n8i58h )
   ji6neoaepv8b5o6k4ev33abha8ht9fgc.example. RRSIG NSEC3 7 2 3600 (
                           20150420235959 20051021000000 40430 example.
                           gPkFp1s2QDQ6wQzcg1uSebZ61W33rUBDcTj7
                           2F3kQ490fEdp7k1BUIfbcZtPbX3YCpE+sIt0
                           MpzVSKfTwx4uYA== )

   ;; Additional
   ;; (empty)
```

   The query returned an NSEC3 RR that proves that the requested name
   exists ("y.w.example." hashes to "ji6neoaepv8b5o6k4ev33abha8ht9fgc"),
   but the requested RR type does not exist (Type A is absent in the
   Type Bit Maps field of the NSEC3 RR). Note that, unlike an empty
   non-terminal proof using NSECs, this is identical to a No Data Error.
   This example is solely mentioned to be complete.

B.3. Referral to an Opt-Out Unsigned Zone

   The NSEC3 RRs prove that nothing for this delegation was signed.
   There is no proof that the unsigned delegation exists.

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

   ;; Answer
   ;; (empty)

   ;; Authority
   c.example.      NS      ns1.c.example.
                   NS      ns2.c.example.

   ;; NSEC3 RR that covers the "next closer" name (c.example)
   ;; H(c.example) = 4g6p9u5gvfshp30pqecj98b3maqbn1ck

   35mthgpgcu1qg68fab165klnsnk3dpvl.example. NSEC3 1 1 12 aabbccdd (
                        b4um86eghhds6nea196smvmlo4ors995 NS DS RRSIG )
   35mthgpgcu1qg68fab165klnsnk3dpvl.example. RRSIG NSEC3 7 2 3600 (
                        20150420235959 20051021000000 40430 example.
                        g6jPUUpduAJKRljUsN8gB4UagAX0NxY9shwQ
                        Aynzo8EUWH+z6hEIBlUTPGj15eZll6VhQqgZ
                        XtAIR3chwgW+SA== )

   ;; NSEC3 RR that matches the closest encloser (example)
   ;; H(example) = 0p9mhaveqvm6t7vbl5lop2u3t2rp3tom

   0p9mhaveqvm6t7vbl5lop2u3t2rp3tom.example. NSEC3 1 1 12 aabbccdd (
                        2t7b4g4vsa5smi47k61mv5bv1a22bojr MX DNSKEY NS
                        SOA NSEC3PARAM RRSIG )
   0p9mhaveqvm6t7vbl5lop2u3t2rp3tom.example. RRSIG NSEC3 7 2 3600 (
                        20150420235959 20051021000000 40430 example.
                        OSgWSm26B+cS+dDL8b5QrWr/dEWhtCsKlwKL
                        IBHYH6blRxK9rC0bMJPwQ4mLIuw85H2EY762
                        BOCXJZMnpuwhpA== )

   ;; Additional
   ns1.c.example. A       192.0.2.7
   ns2.c.example. A       192.0.2.8
```

   The query returned a referral to the unsigned "c.example." zone. The
   response contains the closest provable encloser of "c.example" to be
   "example", since the hash of "c.example"
   ("4g6p9u5gvfshp30pqecj98b3maqbn1ck") is covered by the first NSEC3 RR
   and its Opt-Out bit is set.

B.4. Wildcard Expansion

   A query that was answered with a response containing a wildcard
   expansion. The label count in the RRSIG RRSet in the answer section
   indicates that a wildcard RRSet was expanded to produce this
   response, and the NSEC3 RR proves that no "next closer" name exists
   in the zone.

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

   ;; Answer
   a.z.w.example. MX      1 ai.example.
   a.z.w.example. RRSIG   MX 7 2 3600 20150420235959 20051021000000 (
                          40430 example.
                          CikebjQwGQPwijVcxgcZcSJKtfynugtlBiKb
                          9FcBTrmOoyQ4InoWVudhCWsh/URX3lc4WRUM
                          ivEBP6+4KS3ldA== )

   ;; Authority
   example.        NS     ns1.example.
   example.        NS     ns2.example.
   example.        RRSIG  NS 7 1 3600 20150420235959 20051021000000 (
                          40430 example.
                          PVOgtMK1HHeSTau+HwDWC8Ts+6C8qtqd4pQJ
                          qOtdEVgg+MA+ai4fWDEhu3qHJyLcQ9tbD2vv
                          CnMXjtz6SyObxA== )

   ;; NSEC3 RR that covers the "next closer" name (z.w.example)
   ;; H(z.w.example) = qlu7gtfaeh0ek0c05ksfhdpbcgglbe03

   q04jkcevqvmu85r014c7dkba38o0ji5r.example. NSEC3 1 1 12 aabbccdd (
                          r53bq7cc2uvmubfu5ocmm6pers9tk9en A RRSIG )
   q04jkcevqvmu85r014c7dkba38o0ji5r.example. RRSIG NSEC3 7 2 3600 (
                          20150420235959 20051021000000 40430 example.
                          hV5I89b+4FHJDATp09g4bbN0R1F845CaXpL3
                          ZxlMKimoPAyqletMlEWwLfFia7sdpSzn+ZlN
                          NlkxWcLsIlMmUg== )

   ;; Additional
   ai.example.     A      192.0.2.9
   ai.example.     RRSIG  A 7 2 3600 20150420235959 20051021000000 (
                          40430 example.
                          hVe+wKYMlObTRPhX0NL67GxeZfdxqr/QeR6F
                          tfdAj5+FgYxyzPEjIzvKWy00hWIl6wD3Vws+
                          rznEn8sQ64UdqA== )
   ai.example.     AAAA   2001:db8:0:0:0:0:f00:baa9
   ai.example.     RRSIG  AAAA 7 2 3600 20150420235959 20051021000000 (
                          40430 example.
                          LcdxKaCB5bGZwPDg+3JJ4O02zoMBrjxqlf6W
                          uaHQZZfTUpb9Nf2nxFGe2XRPfR5tpJT6GdRG
                          cHueLuXkMjBArQ== )
```

The query returned an answer that was produced as a result of a
wildcard expansion. The answer section contains a wildcard RRSet
expanded as it would be in a traditional DNS response. The RRSIG
Labels field value of 2 indicates that the answer is the result of a
wildcard expansion, as the "a.z.w.example" name contains 4 labels.
This also shows that "w.example" exists, so there is no need for an
NSEC3 RR that matches the closest encloser.

The NSEC3 RR proves that no closer match could have been used to
answer this query.

B.5. Wildcard No Data Error

A "no data" response for a name covered by a wildcard. The NSEC3 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.        SOA     ns1.example. bugs.x.w.example. 1 3600 300 (
                        3600000 3600 )
example.        RRSIG   SOA 7 1 3600 20150420235959 20051021000000 (
                        40430 example.
                        Hu25UIyNPmvPIVBrldN+9Mlp9Zql39qaUd8i
                        q4ZLlYWfUUbbAS41pG+68z81q1xhkYAcEyHd
                        VI2LmKusbZsT0Q== )

;; NSEC3 RR that matches the closest encloser (w.example)
;; H(w.example) = k8udemvp1j2f7eg6jebps17vp3n8i58h

k8udemvp1j2f7eg6jebps17vp3n8i58h.example. NSEC3 1 1 12 aabbccdd (
                        kohar7mbb8dc2ce8a9qvl8hon4k53uhi )
k8udemvp1j2f7eg6jebps17vp3n8i58h.example. RRSIG NSEC3 7 2 3600 (
                        20150420235959 20051021000000 40430 example.
                        FtXGbvF0+wf8iWkyo73enAuVx03klN+pILBK
                        S6qCcftVtfH4yVzsEZquJ27NHR7ruxJWDNMt
                        Otx7w9WfcIg62A== )

;; NSEC3 RR that covers the "next closer" name (z.w.example)
;; H(z.w.example) = qlu7gtfaeh0ek0c05ksfhdpbcgglbe03
```

```
q04jkcevqvmu85r014c7dkba38o0ji5r.example. NSEC3 1 1 12 aabbccdd (
                    r53bq7cc2uvmubfu5ocmm6pers9tk9en A RRSIG )
q04jkcevqvmu85r014c7dkba38o0ji5r.example. RRSIG NSEC3 7 2 3600 (
                    20150420235959 20051021000000 40430 example.
                    hV5I89b+4FHJDATp09g4bbN0R1F845CaXpL3
                    ZxlMKimoPAyqletMlEWwLfFia7sdpSzn+ZlN
                    NlkxWcLsIlMmUg== )
```

```
;; NSEC3 RR that matches a wildcard at the closest encloser.
;; H(*.w.example) = r53bq7cc2uvmubfu5ocmm6pers9tk9en
```

```
r53bq7cc2uvmubfu5ocmm6pers9tk9en.example. NSEC3 1 1 12 aabbccdd (
                    t644ebqk9bibcna874givr6joj62mlhv MX RRSIG )
r53bq7cc2uvmubfu5ocmm6pers9tk9en.example. RRSIG NSEC3 7 2 3600 (
                    20150420235959 20051021000000 40430 example.
                    aupviViruXs4bDg9rCbezzBMf9h1ZlDvbW/C
                    ZFKulIGXXLj8B/fsDJarXVDA9bnUoRhEbKp+
                    HF1FWKW7RIJdtQ== )
```

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

The query returned the NSEC3 RRs that prove that the requested data
does not exist and no wildcard RR applies.

B.6. 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.        SOA     ns1.example. bugs.x.w.example. 1 3600 300 (
                       3600000 3600 )
example.        RRSIG   SOA 7 1 3600 20150420235959 20051021000000 (
                       40430 example.
                       Hu25UIyNPmvPIVBrldN+9Mlp9Zql39qaUd8i
                       q4ZLlYWfUUbbAS41pG+68z81q1xhkYAcEyHd
                       VI2LmKusbZsT0Q== )

;; NSEC3 RR matches the QNAME and shows that the DS type bit is not set.

0p9mhaveqvm6t7vbl5lop2u3t2rp3tom.example. NSEC3 1 1 12 aabbccdd (
                       2t7b4g4vsa5smi47k61mv5bv1a22bojr MX DNSKEY NS
                       SOA NSEC3PARAM RRSIG )
0p9mhaveqvm6t7vbl5lop2u3t2rp3tom.example. RRSIG NSEC3 7 2 3600
                       20150420235959 20051021000000 40430 example.
                       OSgWSm26B+cS+dDL8b5QrWr/dEWhtCsKlwKL
                       IBHYH6blRxK9rC0bMJPwQ4mLIuw85H2EY762
                       BOCXJZMnpuwhpA== )

;; Additional
;; (empty)
```

   The query returned an NSEC3 RR showing that the requested was
   answered by the server authoritative for the zone "example". The
   NSEC3 RR indicates the presence of an SOA RR, showing that this NSEC3
   RR is from the apex of the child, not from the zone cut of the
   parent. Queries for the "example" DS RRSet should be sent to the
   parent servers (which are in this case the root servers).

Appendix C. Special Considerations

   The following paragraphs clarify specific behavior and explain
   special considerations for implementations.

C.1. Salting

   Augmenting original owner names with salt before hashing increases
   the cost of a dictionary of pre-generated hash-values. For every bit
   of salt, the cost of a precomputed dictionary doubles (because there

must be an entry for each word combined with each possible salt
value). The NSEC3 RR can use a maximum of 2040 bits (255 octets) of
salt, multiplying the cost by 2^{2040}. This means that an attacker
must, in practice, recompute the dictionary each time the salt is
changed.

Including a salt, regardless of size, does not affect the cost of
constructing NSEC3 RRs. It does increase the size of the NSEC3 RR.

There MUST be at least one complete set of NSEC3 RRs for the zone
using the same salt value.

The salt SHOULD be changed periodically to prevent pre-computation
using a single salt. It is RECOMMENDED that the salt be changed for
every re-signing.

Note that this could cause a resolver to see RRs with different salt
values for the same zone. This is harmless, since each RR stands
alone (that is, it denies the set of owner names whose hashes, using
the salt in the NSEC3 RR, fall between the two hashes in the NSEC3
RR) -- it is only the server that needs a complete set of NSEC3 RRs
with the same salt in order to be able to answer every possible
query.

There is no prohibition with having NSEC3 RRs with different salts
within the same zone. However, in order for authoritative servers to
be able to consistently find covering NSEC3 RRs, the authoritative
server MUST choose a single set of parameters (algorithm, salt, and
iterations) to use when selecting NSEC3 RRs.

C.2. Hash Collision

   Hash collisions occur when different messages have the same hash
   value. The expected number of domain names needed to give a 1 in 2
   chance of a single collision is about $2^{(n/2)}$ for a hash of length n
   bits (i.e., 2^{80} for SHA-1). Though this probability is extremely
   low, the following paragraphs deal with avoiding collisions and
   assessing possible damage in the event of an attack using hash
   collisions.

C.2.1. Avoiding Hash Collisions During Generation

   During generation of NSEC3 RRs, hash values are supposedly unique.
   In the (academic) case of a collision occurring, an alternative salt
   MUST be chosen and all hash values MUST be regenerated.

C.2.2. Second Preimage Requirement Analysis

   A cryptographic hash function has a second-preimage resistance
   property. The second-preimage resistance property means that it is
   computationally infeasible to find another message with the same hash
   value as a given message, i.e., given preimage X, to find a second
   preimage X' != X such that hash(X) = hash(X'). The work factor for
   finding a second preimage is of the order of 2^160 for SHA-1. To
   mount an attack using an existing NSEC3 RR, an adversary needs to
   find a second preimage.

   Assuming an adversary is capable of mounting such an extreme attack,
   the actual damage is that a response message can be generated that
   claims that a certain QNAME (i.e., the second pre-image) does exist,
   while in reality QNAME does not exist (a false positive), which will
   either cause a security-aware resolver to re-query for the non-
   existent name, or to fail the initial query. Note that the adversary
   can't mount this attack on an existing name, but only on a name that
   the adversary can't choose and that does not yet exist.

Authors' Addresses

   Ben Laurie
   Nominet
   17 Perryn Road
   London W3 7LR
   England

   Phone: +44 20 8735 0686
   EMail: ben@links.org

   Geoffrey Sisson
   Nominet
   Minerva House
   Edmund Halley Road
   Oxford Science Park
   Oxford OX4 4DQ
   UNITED KINGDOM

   Phone: +44 1865 332211
   EMail: geoff-s@panix.com

   Roy Arends
   Nominet
   Minerva House
   Edmund Halley Road
   Oxford Science Park
   Oxford OX4 4DQ
   UNITED KINGDOM

   Phone: +44 1865 332211
   EMail: roy@nominet.org.uk

   David Blacka
   VeriSign, Inc.
   21355 Ridgetop Circle
   Dulles, VA 20166
   US

   Phone: +1 703 948 3200
   EMail: davidb@verisign.com