Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 2, 2019

J. Vcelak
CZ.NIC
S. Goldberg
Boston University
D. Papadopoulos
HKUST
S. Huque
Salesforce
D. Lawrence
Dyn

December 29, 2018

NSEC5, DNSSEC Authenticated Denial of Existence
draft-vcelak-nsec5-08

Abstract

The Domain Name System Security Extensions (DNSSEC) introduced two resource records (RR) for authenticated denial of existence: the NSEC RR and the NSEC3 RR. This document introduces NSEC5 as an alternative mechanism for DNSSEC authenticated denial of existence. NSEC5 uses verifiable random functions (VRFs) to prevent offline enumeration of zone contents. NSEC5 also protects the integrity of the zone contents even if an adversary compromises one of the authoritative servers for the zone. Integrity is preserved because NSEC5 does not require private zone-signing keys to be present on all authoritative servers for the zone, in contrast to DNSSEC online signing schemes like NSEC3 White Lies.

Ed note

Text inside square brackets ([]) is additional background information, answers to frequently asked questions, general musings, etc. They will be removed before publication. This document is being collaborated on in GitHub at <https://github.com/fcelda/nsec5-draft>. The most recent version of the document, open issues, etc should all be available there. The authors gratefully accept pull requests.

Status of This Memo

Table of Contents

1. Introduction

1.1. Rationale

NSEC5 provides an alternative mechanism for authenticated denial of existence for the DNS Security Extensions (DNSSEC). NSEC5 has two key security properties. First, NSEC5 protects the integrity of the

zone contents even if an adversary compromises one of the authoritative servers for the zone. Second, NSEC5 prevents offline zone enumeration, where an adversary makes a small number of online DNS queries and then processes them offline in order to learn all of the names in a zone. Zone enumeration can be used to identify routers, servers or other "things" that could then be targeted in more complex attacks. An enumerated zone can also be a source of probable email addresses for spam, or as a "key for multiple WHOIS queries to reveal registrant data that many registries may have legal obligations to protect" [RFC5155].

All other DNSSEC mechanisms for authenticated denial of existence either fail to preserve integrity against a compromised server, or fail to prevent offline zone enumeration.

When offline signing with NSEC is used [RFC4034], an NSEC chain of all existing domain names in the zone is constructed and signed offline. The chain is made of resource records (RRs), where each RR represents two consecutive domain names in canonical order present in the zone. The authoritative server proves the non-existence of a name by presenting a signed NSEC RR which covers the name. Because the authoritative server does not need not to know the private zone-signing key, the integrity of the zone is protected even if the server is compromised. However, the NSEC chain allows for easy zone enumeration: N queries to the server suffice to learn all N names in the zone (see e.g., [nmap-nsec-enum], [nsec3map], and [ldns-walk]).

When offline signing with NSEC3 is used [RFC5155], the original names in the NSEC chain are replaced by their cryptographic hashes. Offline signing ensures that NSEC3 preserves integrity even if an authoritative server is compromised. However, offline zone enumeration is still possible with NSEC3 (see e.g., [nsec3walker], [nsec3gpu]), and is part of standard network reconnaissance tools (e.g., [nmap-nsec3-enum], [nsec3map]).

When online signing is used, the authoritative server holds the private zone-signing key and uses this key to synthesize NSEC or NSEC3 responses on the fly (e.g. NSEC3 White Lies (NSEC3-WL) or Minimally-Covering NSEC, both described in [RFC7129]). Because the synthesized response only contains information about the queried name (but not about any other name in the zone), offline zone enumeration is not possible. However, because the authoritative server holds the private zone-signing key, integrity is lost if the authoritative server is compromised.

Scheme	Integrity vs network attacks?	Integrity vs compromised auth. server?	Prevents offline zone enumeration?	Online crypto?
Unsigned	NO	NO	YES	NO
NSEC	YES	YES	NO	NO
NSEC3	YES	YES	NO	NO
NSEC3-WL	YES	NO	YES	YES
NSEC5	YES	YES	YES	YES

NSEC5 prevents offline zone enumeration and also protects integrity even if a zone's authoritative server is compromised. To do this, NSEC5 replaces the unkeyed cryptographic hash function used in NSEC3 with a verifiable random function (VRF) [I-D.irtf-cfrg-vrf] [MRV99]. A VRF is the public-key version of a keyed cryptographic hash. Only the holder of the private VRF key can compute the hash, but anyone with public VRF key can verify the correctness of the hash.

The public VRF key is distributed in an NSEC5KEY RR, similar to a DNSKEY RR, and is used to verify NSEC5 hash values. The private VRF key is present on all authoritative servers for the zone, and is used to compute hash values. For every query that elicits a negative response, the authoritative server hashes the query on the fly using the private VRF key, and also returns the corresponding precomputed NSEC5 record(s). In contrast to the online signing approach [RFC7129], the private key that is present on all authoritative servers for NSEC5 cannot be used to modify the zone contents.

Like online signing approaches, NSEC5 requires the authoritative server to perform online public key cryptographic operations for every query eliciting a denying response. This is necessary; [nsec5] proved that online cryptography is required to prevent offline zone enumeration while still protecting the integrity of zone contents against network attacks.

NSEC5 is not intended to replace NSEC or NSEC3. It is an alternative mechanism for authenticated denial of existence. This document specifies NSEC5 based on the VRFs in [I-D.irtf-cfrg-vrf] over the FIPS 186-3 P-256 elliptic curve and over the the Ed25519 elliptic curve. A formal cryptographic proof of security for NSEC5 is in [nsec5ecc].

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], and [RFC4035]; subsequent RFCs that update them in [RFC2136], [RFC2181], [RFC2308], [RFC5155], and [RFC7129]; and DNS terms in [RFC7719].

The reader should also be familiar with verifiable random functions (VRFs) as defined in [I-D.irtf-cfrg-vrf].

The following terminology is used through this document:

Base32hex: The "Base 32 Encoding with Extended Hex Alphabet" as specified in [RFC4648]. The padding characters ("=") are not used in the NSEC5 specification.

Base64: The "Base 64 Encoding" as specified in [RFC4648].

QNAME: The domain name being queried (query name).

Private NSEC5 key: The private key for the verifiable random function (VRF).

Public NSEC5 key: The public key for the VRF.

NSEC5 proof: A VRF proof. The holder of the private NSEC5 key (e.g., authoritative server) can compute the NSEC5 proof for an input domain name. Anyone who knows the public VRF key can verify that the NSEC5 proof corresponds to the input domain name.

NSEC5 hash: A cryptographic digest of an NSEC5 proof. If the NSEC5 proof is known, anyone can compute its corresponding NSEC5 hash.

NSEC5 algorithm: A triple of VRF algorithms that compute an NSEC5 proof (VRF_prove), verify an NSEC5 proof (VRF_verify), and process an NSEC5 proof to obtain its NSEC5 hash (VRF_proof2hash).

2. Backward Compatibility

The specification describes a protocol change that is not backward compatible with [RFC4035] and [RFC5155]. An NSEC5-unaware resolver will fail to validate responses introduced by this document.

To prevent NSEC5-unaware resolvers from attempting to validate the responses, new DNSSEC algorithms identifiers are introduced in Section 16 which alias existing algorithm numbers. The zones signed according to this specification MUST use only these algorithm identifiers, thus NSEC5-unaware resolvers will treat the zone as insecure.

3. How NSEC5 Works

With NSEC5, the original domain name is hashed using a VRF [I-D.irtf-cfrg-vrf] using the following steps:

1. The domain name is processed using a VRF keyed with the private NSEC5 key to obtain the NSEC5 proof. Anyone who knows the public NSEC5 key, normally acquired via an NSEC5KEY RR, can verify that a given NSEC5 proof corresponds to a given domain name.

2. The NSEC5 proof is then processed using a publicly-computable VRF proof2hash function to obtain the NSEC5 hash. The NSEC5 hash can be computed by anyone who knows the input NSEC5 proof.

The NSEC5 hash determines the position of a domain name in an NSEC5 chain.

To sign a zone, the private NSEC5 key is used to compute the NSEC5 hashes for each name in the zone. These NSEC5 hashes are sorted in canonical order [RFC4034], and each consecutive pair forms an NSEC5 RR. Each NSEC5 RR is signed offline using the private zone-signing key. The resulting signed chain of NSEC5 RRs is provided to all authoritative servers for the zone, along with the private NSEC5 key.

To prove non-existence of a particular domain name in response to a query, the server uses the private NSEC5 key to compute the NSEC5 proof and NSEC5 hash corresponding to the queried name. The server then identifies the NSEC5 RR that covers the NSEC5 hash, and responds with this NSEC5 RR and its corresponding RRSIG signature RRset, as well as a synthesized NSEC5PROOF RR that contains the NSEC5 proof corresponding to the queried name.

To validate the response, the client verifies the following items:

- The client uses the public NSEC5 key, normally acquired from the NSEC5KEY RR, to verify that the NSEC5 proof in the NSEC5PROOF RR corresponds to the queried name.

- The client uses the VRF proof2hash function to compute the NSEC5 hash from the NSEC5 proof in the NSEC5PROOF RR. The client verifies that the NSEC5 hash is covered by the NSEC5 RR.

- o The client verifies that the NSEC5 RR is validly signed by the RRSIG RRset.

4. NSEC5 Algorithms

The algorithms used for NSEC5 authenticated denial are independent of the algorithms used for DNSSEC signing. An NSEC5 algorithm defines how the NSEC5 proof and the NSEC5 hash are computed and validated.

The NSEC5 proof corresponding to a name is computed using ECVRF_prove(), as specified in [I-D.irtf-cfrg-vrf]. The input to ECVRF_prove() is a public NSEC5 key followed by a private NSEC5 key followed by an RR owner name in [RFC4034] canonical wire format. The output NSEC5 proof is an octet string.

An NSEC5 hash corresponding to a name is computed from its NSEC5 proof using ECVRF_proof2hash(), as specified in [I-D.irtf-cfrg-vrf]. The input to VRF_proof2hash() is an NSEC5 proof as an octet string. The output NSEC5 hash is either an octet string, or INVALID.

An NSEC5 proof for a name is verified using ECVRF_verify(), as specified in [I-D.irtf-cfrg-vrf]. The input is the NSEC5 public key, followed by an NSEC5 proof as an octet string, followed by an RR owner name in [RFC4034] canonical wire format. The output is either VALID or INVALID.

This document defines the EC-P256-SHA256 NSEC5 algorithm as follows:

- o The VRF is the ECVRF algorithm using the ECVRF-P256-SHA256 ciphersuite specified in [I-D.irtf-cfrg-vrf].

- o The public key format to be used in the NSEC5KEY RR is defined in Section 4 of [RFC6605] and thus is the same as the format used to store ECDSA public keys in DNSKEY RRs.
  [NOTE: This specification does not compress the elliptic curve point used for the public key, but we do compress curve points in every other place we use them. The NSEC5KEY record can be shrunk by 31 additional octets by encoding the public key with point compression.]

This document defines the EC-ED25519-SHA512 NSEC5 algorithm as follows:

- o The VRF is the EC-VRF algorithm using the ECVRF-ED25519-SHA512 ciphersuite specified in [I-D.irtf-cfrg-vrf].

o The public key format to be used in the NSEC5KEY RR is defined in Section 3 of [RFC8080] and thus is the same as the format used to store Ed25519 public keys in DNSKEY RRs.

[NOTE: Could alternatively have the EC-ED25519-SHA512 NSEC5 ciphersuite use the EC-VRF-ED25519-SHA512-ELLIGATOR2 ciphersuite specified in [I-D.irtf-cfrg-vrf].]

5. The NSEC5KEY Resource Record

The NSEC5KEY RR stores a public NSEC5 key. The key allows clients to validate an NSEC5 proof sent by a server.

5.1. NSEC5KEY RDATA Wire Format

The RDATA for the NSEC5KEY 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Algorithm   |                  Public Key                   /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Algorithm is a single octet identifying the NSEC5 algorithm.

Public Key is a variable-sized field holding public key material for NSEC5 proof verification.

5.2. NSEC5KEY RDATA Presentation Format

The presentation format of the NSEC5KEY RDATA is as follows:

The Algorithm field is represented as an unsigned decimal integer.

The Public Key field is represented in Base64 encoding. Whitespace is allowed within the Base64 text.

6. The NSEC5 Resource Record

The NSEC5 RR provides authenticated denial of existence for an RRset or domain name. One NSEC5 RR represents one piece of an NSEC5 chain, proving existence of the owner name and non-existence of other domain names in the part of the hashed domain space that is covered until the next owner name hashed in the RDATA.

6.1. NSEC5 RDATA Wire Format

The RDATA for the NSEC5 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Key Tag            |     Flags     |  Next Length  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Next Hashed Owner Name                    /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                         Type Bit Maps                         /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Key Tag field contains the key tag value of the NSEC5KEY RR that validates the NSEC5 RR, in network byte order. The value is computed from the NSEC5KEY RDATA using the same algorithm used to compute key tag values for DNSKEY RRs. This algorithm is defined in [RFC4034].

The Flags field is a single octet. The meaning of individual bits of the field is defined in Section 6.2.

The Next Length field is an unsigned single octet specifying the length of the Next Hashed Owner Name field in octets.

The Next Hashed Owner Name field is a sequence of binary octets. It contains an NSEC5 hash of the next domain name in the NSEC5 chain.

Type Bit Maps is a variable-sized field encoding RR types present at the original owner name matching the NSEC5 RR. The format of the field is equivalent to the format used in the NSEC3 RR, described in [RFC5155].

6.2. NSEC5 Flags Field

The following one-bit NSEC5 flags are defined:

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

O - Opt-Out flag

W - Wildcard flag

All the other flags are reserved for future use and MUST be zero.

The Opt-Out flag has the same semantics as in NSEC3. The definition and considerations in [RFC5155] are valid, except that NSEC3 is replaced by NSEC5.

The Wildcard flag indicates that a wildcard synthesis is possible at the original domain name level (i.e., there is a wildcard node immediately descending from the immediate ancestor of the original domain name). The purpose of the Wildcard flag is to reduce the maximum number of RRs required for an authenticated denial of existence proof from (at most) three to (at most) two, as originally described in [I-D.gieben-nsec4] Section 7.2.1.

6.3. NSEC5 RDATA Presentation Format

The presentation format of the NSEC5 RDATA is as follows:

The Key Tag field is represented as an unsigned decimal integer.

The Flags field is represented as an unsigned decimal integer.

The Next Length field is not represented.

The Next Hashed Owner Name field is represented as a sequence of case-insensitive Base32hex digits without any whitespace and without padding.

The Type Bit Maps representation is equivalent to the representation used in NSEC3 RR, described in [RFC5155].

7. The NSEC5PROOF Resource Record

The NSEC5PROOF record is not to be included in the zone file. The NSEC5PROOF record contains the NSEC5 proof, proving the position of the owner name in an NSEC5 chain.

7.1. NSEC5PROOF RDATA Wire Format

The RDATA for the NSEC5PROOF RR is 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Key Tag           |        Owner Name Hash        /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Key Tag field contains the key tag value of the NSEC5KEY RR that validates the NSEC5PROOF RR, in network byte order.

Owner Name Hash is a variable-sized sequence of binary octets encoding the NSEC5 proof of the owner name of the RR.

7.2. NSEC5PROOF RDATA Presentation Format

The presentation format of the NSEC5PROOF RDATA is as follows:

The Key Tag field is represented as an unsigned decimal integer.

The Owner Name Hash is represented in Base64 encoding. Whitespace is allowed within the Base64 text.

8. Types of Authenticated Denial of Existence with NSEC5

This section summarizes all possible types of authenticated denial of existence. For each type the following lists are included:

1. Facts to prove: the minimum amount of information that an authoritative server must provide to a client to assure the client that the response content is valid.

2. Authoritative server proofs: the names for which the NSEC5PROOF RRs are synthesized and added into the response along with the NSEC5 RRs matching or covering each such name. These records together prove the listed facts.

3. Validator checks: the individual checks that a validating server is required to perform on a response. The response content is considered valid only if all of the checks pass.

If NSEC5 is said to match a domain name, the owner name of the NSEC5 RR has to be equivalent to an NSEC5 hash of that domain name. If an NSEC5 RR is said to cover a domain name, the NSEC5 hash of the domain name must sort in canonical order between that NSEC5 RR's Owner Name and Next Hashed Owner Name.

8.1. Name Error Responses

Facts to prove:

Non-existence of the domain name that explictly matches the QNAME.

Non-existence of the wildcard that matches the QNAME.

Authoritative server proofs:

NSEC5PROOF for closest encloser and matching NSEC5 RR.

NSEC5PROOF for next closer name and covering NSEC5 RR.

Validator checks:

Closest encloser is in the zone.

The NSEC5 RR matching the closest encloser has its Wildcard flag cleared.

The NSEC5 RR matching the closest encloser does not have NS without SOA in the Type Bit Map.

The NSEC5 RR matching the closest encloser does not have DNAME in the Type Bit Map.

Next closer name is not in the zone.

8.2. No Data Responses

The processing of a No Data response for DS QTYPE differs if the Opt-Out is in effect. For DS QTYPE queries, the validator has two possible checking paths. The correct path can be simply decided by inspecting if the NSEC5 RR in the response matches the QNAME.

Note that the Opt-Out is valid only for DS QTYPE queries.

8.2.1. No Data Response, Opt-Out Not In Effect

Facts to prove:

Existence of an RRset explicitly matching the QNAME.

Non-existence of QTYPE RRset matching the QNAME.

Non-existence of CNAME RRset matching the QNAME.

Authoritative server proofs:

NSEC5PROOF for the QNAME and matching NSEC5 RR.

Validator checks:

QNAME is in the zone.

NSEC5 RR matching the QNAME does not have QTYPE in Type Bit Map.

NSEC5 RR matching the QNAME does not have CNAME in Type Bit Map.

8.2.2. No Data Response, Opt-Out In Effect

Facts to prove:

The delegation is not covered by the NSEC5 chain.

Authoritative server proofs:

NSEC5PROOF for closest provable encloser and matching NSEC5 RR.

Validator checks:

Closest provable encloser is in zone.

Closest provable encloser covers (not matches) the QNAME.

NSEC5 RR matching the closest provable encloser has Opt-Out flag set.

8.3. Wildcard Responses

Facts to prove:

A signed positive response to the QNAME demonstrating the existence of the wildcard (label count in RRSIG is less than in QNAME), and also providing closest encloser name.

Non-existence of the domain name matching the QNAME.

Authoritative server proofs:

A signed positive response for the wildcard expansion of the QNAME.

NSEC5PROOF for next closer name and covering NSEC5 RR.

Validator checks:

Next closer name is not in the zone.

8.4. Wildcard No Data Responses

Facts to prove:

The existence of the wildcard at the closest encloser to the QNAME.

Non-existence of both the QTYPE and of the CNAME type that matches QNAME via wildcard expansion.

Authoritative server proofs:

NSEC5PROOF for source of synthesis (i.e., wildcard at closest encloser) and matching NSEC5 RR.

NSEC5PROOF for next closer name and covering NSEC5 RR.

Validator checks:

Closest encloser to the QNAME exists.

NSEC5 RR matching the wildcard label prepended to the closest encloser, and which does not have the bits corresponding to the QTYPE and CNAME types set it the type bitmap.

9. Authoritative Server Considerations

9.1. Zone Signing

Zones using NSEC5 MUST satisfy the same properties as described in Section 7.1 of [RFC5155], with NSEC3 replaced by NSEC5. In addition, the following conditions MUST be satisfied as well:

o If the original owner name has a wildcard label immediately descending from the original owner name, the corresponding NSEC5 RR MUST have the Wildcard flag set in the Flags field. Otherwise, the flag MUST be cleared.

o The zone apex MUST include an NSEC5KEY RRset containing a NSEC5 public key allowing verification of the current NSEC5 chain.

The following steps describe one possible method to properly add required NSEC5 related records into a zone. This is not the only such existing method.

1. Select an algorithm for NSEC5 and generate the public and private NSEC5 keys.

2. Add an NSEC5KEY RR into the zone apex containing the public NSEC5 key.

3. For each unique original domain name in the zone and each empty non-terminal, add an NSEC5 RR. If Opt-Out is used, owner names of unsigned delegations MAY be excluded.

A. The owner name of the NSEC5 RR is the NSEC5 hash of the original owner name encoded in Base32hex without padding, prepended as a single label to the zone name.

B. Set the Key Tag field to be the key tag corresponding to the public NSEC5 key.

C. Clear the Flags field. If Opt-Out is being used, set the Opt-Out flag. If there is a wildcard label directly descending from the original domain name, set the Wildcard flag. Note that the wildcard can be an empty non-terminal (i.e., the wildcard synthesis does not take effect and therefore the flag is not to be set).

D. Set the Next Length field to a value determined by the used NSEC5 algorithm. Leave the Next Hashed Owner Name field blank.

E. Set the Type Bit Maps field based on the RRsets present at the original owner name.

4. Sort the set of NSEC5 RRs into canonical order.

5. For each NSEC5 RR, set the Next Hashed Owner Name field by using the owner name of the next NSEC5 RR in the canonical order. If the updated NSEC5 is the last NSEC5 RR in the chain, the owner name of the first NSEC5 RR in the chain is used instead.

The NSEC5KEY and NSEC5 RRs MUST have the same class as the zone SOA RR. Also the NSEC5 RRs SHOULD have the same TTL value as the SOA minimum TTL field.

Notice that a use of Opt-Out is not indicated in the zone. This does not affect the ability of a server to prove insecure delegations. The Opt-Out MAY be part of the zone-signing tool configuration.

9.1.1. Precomputing Closest Provable Encloser Proofs

Per Section 8, the worst-case scenario when answering a negative query with NSEC5 requires the authoritative server to respond with two NSEC5PROOF RRs and two NSEC5 RRs. One pair of NSEC5PROOF and NSEC5 RRs corresponds to the closest provable encloser, and the other pair corresponds to the next closer name. The NSEC5PROOF corresponding to the next closer name MUST be computed on the fly by the authoritative server when responding to the query. However, the NSEC5PROOF corresponding to the closest provable encloser MAY be precomputed and stored as part of zone signing.

Precomputing NSEC5PROOF RRs can halve the number of online cryptographic computations required when responding to a negative query. NSEC5PROOF RRs MAY be precomputed as part of zone signing as follows: For each NSEC5 RR, compute an NSEC5PROOF RR corresponding to the original owner name of the NSEC5 RR. The content of the precomputed NSEC5PROOF record MUST be the same as if the record was computed on the fly when serving the zone. NSEC5PROOF records are not part of the zone and SHOULD be stored separately from the zone file.

9.2. Zone Serving

This specification modifies DNSSEC-enabled DNS responses generated by authoritative servers. In particular, it replaces use of NSEC or NSEC3 RRs in such responses with NSEC5 RRs and adds NSEC5PROOF RRs.

According to the type of a response, an authoritative server MUST include NSEC5 RRs in the response, as defined in Section 8. For each NSEC5 RR in the response, a corresponding RRSIG RRset and an NSEC5PROOF MUST be added as well. The NSEC5PROOF RR has its owner name set to the domain name required according to the description in Section 8. The class and TTL of the NSEC5PROOF RR MUST be the same as the class and TTL value of the corresponding NSEC5 RR. The RDATA payload of the NSEC5PROOF is set according to the description in Section 7.1.

Notice that the NSEC5PROOF owner name can be a wildcard (e.g., source of synthesis proof in wildcard No Data responses). The name also always matches the domain name required for the proof while the NSEC5 RR may only cover (not match) the name in the proof (e.g., closest encloser in Name Error responses).

If NSEC5 is used, an answering server MUST use exactly one NSEC5 chain for one signed zone.

NSEC5 MUST NOT be used in parallel with NSEC, NSEC3, or any other authenticated denial of existence mechanism that allows for enumeration of zone contents, as this would defeat a principal security goal of NSEC5.

Similarly to NSEC3, the owner names of NSEC5 RRs are not represented in the NSEC5 chain and therefore NSEC5 records deny their own existence. The desired behavior caused by this paradox is the same as described in Section 7.2.8 of [RFC5155].

9.3. NSEC5KEY Rollover Mechanism

Replacement of the NSEC5 key implies generating a new NSEC5 chain. The NSEC5KEY rollover mechanism is similar to "Pre-Publish Zone Signing Key Rollover" as specified in [RFC6781]. The NSEC5KEY rollover MUST be performed as a sequence of the following steps:

1. A new public NSEC5 key is added into the NSEC5KEY RRset in the zone apex.

2. The old NSEC5 chain is replaced by a new NSEC5 chain constructed using the new key. This replacement MUST happen as a single atomic operation; the server MUST NOT be responding with RRs from both the new and old chain at the same time.

3. The old public key is removed from the NSEC5KEY RRset in the zone apex.

The minimum delay between steps 1 and 2 MUST be the time it takes for the data to propagate to the authoritative servers, plus the TTL value of the old NSEC5KEY RRset.

The minimum delay between steps 2 and 3 MUST be the time it takes for the data to propagate to the authoritative servers, plus the maximum zone TTL value of any of the data in the previous version of the zone.

9.4. Secondary Servers

This document does not define mechanism to distribute private NSEC5 keys. See Section 15.2 for security considerations for private NSEC5 keys.

9.5. Zones Using Unknown NSEC5 Algorithms

Zones that are signed with an unknown NSEC5 algorithm or with an unavailable private NSEC5 key cannot be effectively served. Such zones SHOULD be rejected when loading and servers SHOULD respond with RCODE=2 (Server failure) when handling queries that would fall under such zones.

9.6. Dynamic Updates

A zone signed using NSEC5 MAY accept dynamic updates [RFC2136]. The changes to the zone MUST be performed in a way that ensures that the zone satisfies the properties specified in Section 9.1 at any time. The process described in [RFC5155] Section 7.5 describes how to

handle the issues surrounding the handling of empty non-terminals as well as Opt-Out.

It is RECOMMENDED that the server rejects all updates containing changes to the NSEC5 chain and its related RRSIG RRs, and performs itself any required alternations of the NSEC5 chain induced by the update. Alternatively, the server MUST verify that all the properties are satisfied prior to performing the update atomically.

10. Resolver Considerations

The same considerations as described in Section 9 of [RFC5155] for NSEC3 apply to NSEC5. In addition, as NSEC5 RRs can be validated only with appropriate NSEC5PROOF RRs, the NSEC5PROOF RRs MUST be all together cached and included in responses with NSEC5 RRs.

11. Validator Considerations

11.1. Validating Responses

The validator MUST ignore NSEC5 RRs with Flags field values other than the ones defined in Section 6.2.

The validator MAY treat responses as bogus if the response contains NSEC5 RRs that refer to a different NSEC5KEY.

According to a type of a response, the validator MUST verify all conditions defined in Section 8. Prior to making decision based on the content of NSEC5 RRs in a response, the NSEC5 RRs MUST be validated.

To validate a denial of existence, public NSEC5 keys for the zone are required in addition to DNSSEC public keys. Similarly to DNSKEY RRs, the NSEC5KEY RRs are present at the zone apex.

The NSEC5 RR is validated as follows:

1. Select a correct public NSEC5 key to validate the NSEC5 proof. The Key Tag value of the NSEC5PROOF RR must match with the key tag value computed from the NSEC5KEY RDATA.

2. Validate the NSEC5 proof present in the NSEC5PROOF Owner Name Hash field using the public NSEC5 key. If there are multiple NSEC5KEY RRs matching the key tag, at least one of the keys must validate the NSEC5 proof.

3. Compute the NSEC5 hash value from the NSEC5 proof and check if the response contains NSEC5 RR matching or covering the computed

NSEC5 hash. The TTL values of the NSEC5 and NSEC5PROOF RRs must be the same.

4. Validate the signature on the NSEC5 RR.

If the NSEC5 RR fails to validate, it MUST be ignored. If some of the conditions required for an NSEC5 proof are not satisfied, the response MUST be treated as bogus.

Notice that determining the closest encloser and next closer name in NSEC5 is easier than in NSEC3. NSEC5 and NSEC5PROOF RRs are always present in pairs in responses and the original owner name of the NSEC5 RR matches the owner name of the NSEC5PROOF RR.

11.2. Validating Referrals to Unsigned Subzones

The same considerations as defined in Section 8.9 of [RFC5155] for NSEC3 apply to NSEC5.

11.3. Responses With Unknown NSEC5 Algorithms

A validator MUST ignore NSEC5KEY RRs with unknown NSEC5 algorithms. The practical result of this is that zones signed with unknown algorithms will be considered bogus.

12. Special Considerations

12.1. Transition Mechanism

[TODO: The following information will be covered.]

o Transition to NSEC5 from NSEC/NSEC3

o Transition from NSEC5 to NSEC/NSEC3

o Transition to new NSEC5 algorithms

12.2. Private NSEC5 keys

This document does not define a format to store private NSEC5 keys. Use of a standardized and adopted format is RECOMMENDED.

The private NSEC5 key MAY be shared between multiple zones, however a separate key is RECOMMENDED for each zone.

12.3. Domain Name Length Restrictions

NSEC5 creates additional restrictions on domain name lengths. 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 depends on the length of the zone name and the NSEC5 algorithm used.

All NSEC5 algorithms defined in this document use 256-bit NSEC5 hash values. Such a value can be encoded in 52 characters in Base32hex without padding. When constructing the NSEC5 RR owner name, the encoded hash is prepended to the name of the zone as a single label which includes the length field of a single octet. The maximum length of the zone name in wire format using the 256-bit hash is therefore 202 octets (255 - 53).

13. Implementation Status

NSEC5 has been implemented for the Knot DNS authoritative server (version 1.6.4) and the Unbound recursive server (version 1.5.9). The implementations did not introduce additional library dependencies; all cryptographic primitives are already present in OpenSSL v1.0.2j, which is used by both implementations. The implementations support the full spectrum of negative responses, (i.e., NXDOMAIN, NODATA, Wildcard, Wildcard NODATA, and unsigned delegation) using the EC-P256-SHA256 algorithm. The code is deliberately modular, so that the EC-ED25519-SHA256 algorithm could be implemented by using the Ed25519 elliptic curve [RFC8080] as a drop-in replacement for the P256 elliptic curve. The authoritative server implements the optimization from Section 9.1.1 to precompute the NSEC5PROOF RRs matching each NSEC5 record.

14. Performance Considerations

The performance of NSEC5 has been evaluated in [nsec5ecc].

15. Security Considerations

15.1. Zone Enumeration Attacks

NSEC5 is robust to zone enumeration via offline dictionary attacks by any attacker that does not know the private NSEC5 key. Without the private NSEC5 key, that attacker cannot compute the NSEC5 proof that corresponds to a given domain name. The only way it can learn the NSEC5 proof value for a domain name is by querying the authoritative server for that name. Without the NSEC5 proof value, the attacker

cannot learn the NSEC5 hash value. Thus, even an attacker that collects the entire chain of NSEC5 RR for a zone cannot use offline attacks to "reverse" that NSEC5 hash values in these NSEC5 RR and thus learn which names are present in the zone. A formal cryptographic proof of this property is in [nsec5] and [nsec5ecc].

15.2. Compromise of the Private NSEC5 Key

NSEC5 requires authoritative servers to hold the private NSEC5 key, but not the private zone-signing keys or the private key-signing keys for the zone.

The private NSEC5 key cannot be used to modify zone contents, because zone contents are signed using the private zone-signing key. As such, a compromise of the private NSEC5 key does not compromise the integrity of the zone. An adversary that learns the private NSEC5 key can, however, perform offline zone-enumeration attacks. For this reason, the private NSEC5 key need only be as secure as the DNSSEC records whose privacy (against zone enumeration) is being protected by NSEC5. A formal cryptographic proof of this property is in [nsec5] and [nsec5ecc].

To preserve this property of NSEC5, the private NSEC5 key MUST be different from the private zone-signing keys or key-signing keys for the zone.

15.3. Key Length Considerations

The NSEC5 key must be long enough to withstand attacks for as long as the privacy of the zone contents is important. Even if the NSEC5 key is rolled frequently, its length cannot be too short, because zone privacy may be important for a period of time longer than the lifetime of the key. For example, an attacker might collect the entire chain of NSEC5 RR for the zone over one short period, and then, later (even after the NSEC5 key expires) perform an offline dictionary attack that attempts to reverse the NSEC5 hash values present in the NSEC5 RRs. This is in contrast to zone-signing and key-signing keys used in DNSSEC; these keys, which ensure the authenticity and integrity of the zone contents, need to remain secure only during their lifetime.

15.4. NSEC5 Hash Collisions

If the NSEC5 hash of a QNAME collides with the NSEC5 hash of the owner name of an NSEC5 RR, it will be impossible to prove the non-existence of the colliding QNAME. However, the NSEC5 VRFs ensure that obtaining such a collision is as difficult as obtaining a collision in the SHA-256 hash function, requiring approximately 2^128

effort. Note that DNSSEC already relies on the assumption that a cryptographic hash function is collision-resistant, since these hash functions are used for generating and validating signatures and DS RRs. See also the discussion on key lengths in [nsec5].

16. IANA Considerations

This document updates the IANA registry "Domain Name System (DNS) Parameters" in subregistry "Resource Record (RR) TYPEs", by defining the following new RR types:

NSEC5KEY value TBD.

NSEC5 value TBD.

NSEC5PROOF value TBD.

This document creates a new IANA registry for NSEC5 algorithms. This registry is named "DNSSEC NSEC5 Algorithms". The initial content of the registry is:

0 is Reserved.

1 is EC-P256-SHA256.

2 is EC-ED25519-SHA256.

3-255 is Available for assignment.

This document updates the IANA registry "DNS Security Algorithm Numbers" by defining following aliases:

TBD is NSEC5-ECDSAP256SHA256 alias for ECDSAP256SHA256 (13).

TBD is NSEC5-ED25519, alias for ED25519 (15).

17. Contributors

This document would not be possible without help of Moni Naor (Weizmann Institute), Sachin Vasant (Cisco Systems), Leonid Reyzin (Boston University), and Asaf Ziv (Weizmann Institute) who contributed to the design of NSEC5. Ondrej Sury (CZ.NIC Labs), and Duane Wessels (Verisign Labs) provided advice on the implementation of NSEC5, and assisted with evaluating its performance.

18. References

18.1. Normative References

[FIPS-186-3]
National Institute for Standards and Technology, "Digital Signature Standard (DSS)", FIPS PUB 186-3, June 2009.

[I-D.irtf-cfrg-vrf]
Goldberg, S., Reyzin, L., Papadopoulos, D., and J. Vcelak, "Verifiable Random Functions (VRFs)", draft-irtf-cfrg-vrf-03 (work in progress), September 2018.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <https://www.rfc-editor.org/info/rfc2136>.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <https://www.rfc-editor.org/info/rfc2181>.

[RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <https://www.rfc-editor.org/info/rfc2308>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <https://www.rfc-editor.org/info/rfc4033>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <https://www.rfc-editor.org/info/rfc4034>.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <https://www.rfc-editor.org/info/rfc4035>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC5114] Lepinski, M. and S. Kent, "Additional Diffie-Hellman Groups for Use with IETF Standards", RFC 5114, DOI 10.17487/RFC5114, January 2008, <https://www.rfc-editor.org/info/rfc5114>.

[RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, DOI 10.17487/RFC5155, March 2008, <https://www.rfc-editor.org/info/rfc5155>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[RFC6605] Hoffman, P. and W. Wijngaards, "Elliptic Curve Digital Signature Algorithm (DSA) for DNSSEC", RFC 6605, DOI 10.17487/RFC6605, April 2012, <https://www.rfc-editor.org/info/rfc6605>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

[RFC8080] Sury, O. and R. Edmonds, "Edwards-Curve Digital Security Algorithm (EdDSA) for DNSSEC", RFC 8080, DOI 10.17487/RFC8080, February 2017, <https://www.rfc-editor.org/info/rfc8080>.

18.2. Informative References

[I-D.gieben-nsec4] Gieben, R. and M. Mekking, "DNS Security (DNSSEC) Authenticated Denial of Existence", draft-gieben-nsec4-01 (work in progress), July 2012.

[ldns-walk] NLNetLabs, "ldns", 2015, <http://git.nlnetlabs.nl/ldns/tree/examples/ldns-walk.c>.

[MRV99] Michali, S., Rabin, M., and S. Vadhan, "Verifiable Random Functions", in FOCS, 1999.

[nmap-nsec-enum]
Bond, J., "nmap: dns-nsec-enum", 2011, <https://nmap.org/nsedoc/scripts/dns-nsec-enum.html>.

[nmap-nsec3-enum]
Nikolic, A. and J. Bond, "nmap: dns-nsec3-enum", 2011, <https://nmap.org/nsedoc/scripts/dns-nsec3-enum.html>.

[nsec3gpu]
Wander, M., Schwittmann, L., Boelmann, C., and T. Weis, "GPU-Based NSEC3 Hash Breaking", in IEEE Symp. Network Computing and Applications (NCA), 2014.

[nsec3map]
anonion0, "nsec3map with John the Ripper plugin", 2015, <https://github.com/anonion0/nsec3map.>.

[nsec3walker]
Bernstein, D., "Nsec3 walker", 2011, <http://dnscurve.org/nsec3walker.html>.

[nsec5] Goldberg, S., Naor, M., Papadopoulos, D., Reyzin, L., Vasant, S., and A. Ziv, "NSEC5: Provably Preventing DNSSEC Zone Enumeration", in NDSS'15, July 2014, <https://eprint.iacr.org/2014/582.pdf>.

[nsec5ecc]
Papadopoulos, D., Wessels, D., Huque, S., Vcelak, J., Naor, M., Reyzin, L., and S. Goldberg, "Can NSEC5 be Practical for DNSSEC Deployments?", in ePrint Cryptology Archive 2017/099, February 2017, <https://eprint.iacr.org/2017/099.pdf>.

[RFC6781] Kolkman, O., Mekking, W., and R. Gieben, "DNSSEC Operational Practices, Version 2", RFC 6781, DOI 10.17487/RFC6781, December 2012, <https://www.rfc-editor.org/info/rfc6781>.

[RFC7129] Gieben, R. and W. Mekking, "Authenticated Denial of Existence in the DNS", RFC 7129, DOI 10.17487/RFC7129, February 2014, <https://www.rfc-editor.org/info/rfc7129>.

[RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 7719, DOI 10.17487/RFC7719, December 2015, <https://www.rfc-editor.org/info/rfc7719>.

Appendix A. Examples

We use a small DNS zone to illustrate how negative responses are handled with NSEC5. For brevity, the class is not shown (defaults to IN) and the SOA record is shortened, resulting in the following zone file:

```
example.org.        SOA ( ... )
example.org.        NS  a.example.org

a.example.org.      A 192.0.2.1

c.example.org.      A 192.0.2.2
c.example.org.      TXT "c record"

d.example.org.      NS ns1.d.example.org

ns1.d.example.org.  A 192.0.2.4

g.example.org.      A 192.0.2.1
g.example.org.      TXT "g record"

*.a.example.org.    TXT "wildcard record"
```

Notice the delegation to an unsigned zone d.example.org served by ns1.d.example.org. (Note: if the d.example.org zone was signed, then the example.org zone have a DS record for d.example.org.)

Next we present example responses. All cryptographic values are shortened as indicated by "..." and ADDITIONAL sections have been removed.

A.1. Name Error Example

Consider a query for a type A record for a.b.c.example.org.

The server must prove the following facts:

- o Existence of closest encloser c.example.org.

- o Non-existence of wildcard at closest encloser *.c.example.org.

- o Non-existence of next closer b.c.example.org.

To do this, the server returns:

```
;; ->>HEADER<<- opcode: QUERY; status: NXDOMAIN; id: 5937

;; QUESTION SECTION:
;; a.b.c.example.org.           IN      A

;; AUTHORITY SECTION:
example.org.         3600 IN SOA a.example.org. hostmaster.example.org. (
            2010111214 21600 3600 604800 86400 )

example.org.         3600 IN RRSIG  SOA 16 2 3600 (
            20170412024301 20170313024301 5137 example.org. rT231b1rH... )
```

This is an NSEC5PROOF RR for c.example.com. It's RDATA is the NSEC5 proof corresponding to c.example.com. (NSEC5 proofs are randomized values, because NSEC5 proof values are computed uses the EC-VRF from [I-D.irtf-cfrg-vrf].) Per Section 9.1.1, this NSEC5PROOF RR may be precomputed.

```
   c.example.org.      86400 IN NSEC5PROOF 48566 Amgn22zUiZ9JVyaT...
```

This is a signed NSEC5 RR "matching" c.example.org, which proves the existence of closest encloser c.example.org. The NSEC5 RR has its owner name equal to the NSEC5 hash of c.example.org, which is O4K89V. (NSEC5 hash values are deterministic given the public NSEC5 key.) The NSEC5 RR also has its Wildcard flag cleared (see the "0" after the key ID 48566). This proves the non-existence of the wildcard at the closest encloser *.c.example.com. NSEC5 RRs are precomputed.

```
o4k89v.example.org. 86400 IN NSEC5   48566 0 0O49PI A TXT RRSIG
o4k89v.example.org. 86400 IN RRSIG   NSEC5 16 3 86400 (
            20170412024301 20170313024301 5137 example.org. zDNTSMQNlz... )
```

This is an NSEC5PROOF RR for b.c.example.org. It's RDATA is the NSEC5 proof corresponding to b.c.example.com. This NSEC5PROOF RR must be computed on the fly.

```
   b.c.example.org.    86400 IN NSEC5PROOF 48566 AuvvJqbUcEs8sCpY...
```

This is a signed NSEC5 RR "covering" b.c.example.org, which proves the non-existence of the next closer name b.c.example.org The NSEC5 hash of b.c.example.org, which is AO5OF, sorts in canonical order between the "covering" NSEC5 RR's Owner Name (which is 0O49PI) and Next Hashed Owner Name (which is BAPROH).

```
0o49pi.example.org. 86400 IN NSEC5      48566 0 BAPROH (
            NS SOA RRSIG DNSKEY NSEC5KEY )

0o49pi.example.org. 86400 IN RRSIG   NSEC5 16 3 86400 (
            20170412024301 20170313024301 5137 example.org. 4HT1uj1YlMzO)
```

[TODO: Add discussion of CNAME and DNAME to the example?]

A.2. No Data Example

Consider a query for a type MX record for c.example.org.

The server must prove the following facts:

- Existence of c.example.org. for any type other than MX or CNAME

To do this, the server returns:

```
;; ->>HEADER<<- opcode: QUERY; status: NOERROR; id: 38781

;; QUESTION SECTION:
;; c.example.org.    IN MX

;; AUTHORITY SECTION:
example.org.    3600 IN SOA     a.example.org. hostmaster.example.org. (
            2010111214 21600 3600 604800 86400 )

example.org.    3600 IN RRSIG   SOA 16 2 3600 20170412024301 20170313024301
5137 example.org. /rT231b1rH/p
```

This is an NSEC5PROOF RR for c.example.com. Its RDATA corresponds to the NSEC5 proof for c.example.com. which is a randomized value. Per Section 9.1.1, this NSEC5PROOF RR may be precomputed.

```
   c.example.org. 86400 IN NSEC5PROOF 48566 Amgn22zUiZ9JVyaT
```

This is a signed NSEC5 RR "matching" c.example.org. with CNAME and MX Type Bits cleared and its TXT Type Bit set. This NSEC5 RR has its owner name equal to the NSEC5 hash of c.example.org. This proves the existence of c.example.org. for a type other than MX and CNAME. NSEC5 RR are precomputed.

```
o4k89v.example.org. 86400 IN NSEC5   48566 0 0O49PI A TXT RRSIG

o4k89v.example.org. 86400 IN RRSIG   NSEC5 16 3 86400 (
            20170412024301 20170313024301 5137 example.org. zDNTSMQNlz/J)
```

A.3. Delegation to an Unsigned Zone in an Opt-Out span Example

Consider a query for a type A record for foo.d.example.org.

Here, d.example.org is a delegation to an unsigned zone, which lies within an Opt-Out span.

The server must prove the following facts:

- Non-existence of signature on next closer name d.example.org.
- Opt-out bit is set in NSEC5 record covering next closer name d.example.org.
- Existence of closest provable encloser example.org

To do this, the server returns:

```
;; ->>HEADER<<- opcode: QUERY; status: NOERROR; id: 45866

;; QUESTION SECTION:
;; foo.d.example.org.         IN A

;; AUTHORITY SECTION:
d.example.org.       3600  IN NS      ns1.d.example.org.
```

This is an NSEC5PROOF RR for d.example.org. Its RDATA is the NSEC5 proof corresponding to d.example.org. This NSEC5PROOF RR is computed on the fly.

```
d.example.org.      86400   IN      NSEC5PROOF      48566 A9FpmeH79q7g6VNW
```

This is a signed NSEC5 RR "covering" d.example.org with its Opt-out bit set (see the "1" after the key ID 48566). The NSEC5 hash of d.example.org (which is BLE8LR) sorts in canonical order between the "covering" NSEC5 RR's Owner Name (BAPROH) and Next Hashed Owner Name (JQBMG4). This proves that no signed RR exists for d.example.org, but that the zone might contain a unsigned RR for a name whose NSEC5 hash sorts in canonical order between BAPROH and JQBMG4.

```
baproh.example.org. 86400 IN NSEC5   48566 1 JQBMG4 A TXT RRSIG

baproh.example.org. 86400 IN RRSIG   NSEC5 16 3 86400 (
            20170412024301 20170313024301 5137 example.org. fjTcoRKgdML1)
```

This is an NSEC5PROOF RR for example.com. It's RDATA is the NSEC5 proof corresponding to example.com. Per Section 9.1.1, this NSEC5PROOF RR may be precomputed.

```
   example.org.        86400 IN NSEC5PROOF      48566 AjwsPCJZ8zH/D0Tr
```

This is a signed NSEC5 RR "matching" example.org which proves the existence of a signed RRs for example.org. This NSEC5 RR has its owner name equal to the NSEC5 hash of example.org which is 0O49PI. NSEC5 RR are precomputed.

```
0o49pi.example.org. 86400 IN NSEC5   48566 0 BAPROH (
            NS SOA RRSIG DNSKEY NSEC5KEY)

0o49pi.example.org. 86400 IN RRSIG   NSEC5 16 3 86400 (
            20170412034216 20170313034216 5137 example.org. 4HT1uj1YlMzO)
```

A.4. Wildcard Example

Consider a query for a type TXT record for foo.a.example.org.

The server must prove the following facts:

- o Existence of the TXT record for the wildcard *.a.example.org
- o Non-existence of the next closer name foo.a.example.org.

To do this, the server returns:

```
   ;; ->>HEADER<<- opcode: QUERY; status: NOERROR; id: 53731

   ;; QUESTION SECTION:
   ;; foo.a.example.org.        IN TXT
```

This is a signed TXT record for the wildcard at a.example.org (number of labels is set to 3 in the RRSIG record).

```
;; ANSWER SECTION:
foo.a.example.org.      3600 IN TXT     "wildcard record"

foo.a.example.org.      3600 IN RRSIG   TXT 16 3 3600 (
            20170412024301 20170313024301 5137 example.org. aeaLgZ8sk+98)

;; AUTHORITY SECTION:
example.org.            3600 IN NS      a.example.org.

example.org.            3600 IN RRSIG   NS 16 2 3600 (
            20170412024301 20170313024301 5137 example.org. 8zuN0h2x5WyF)
```

This is an NSEC5PROOF RR for foo.a.example.org. This NSEC5PROOF RR must be computed on-the-fly.

```
  foo.a.example.org.     86400 IN NSEC5PROOF      48566 AjqF5FGGVso40Lda
```

This is a signed NSEC5 RR "covering" foo.a.example.org. The NSEC5 hash of foo.a.example.org is FORDMO and sorts in canonical order between the NSEC5 RR's Owner Name (which is BAPROH) and Next Hashed Owner Name (which is JQBMG4). This proves the non-existence of the next closer name foo.a.example.com. NSEC5 RRs are precomputed.

```
    baproh.example.org. 86400 IN NSEC5   48566 1 JQBMG4 A TXT RRSIG
    baproh.example.org. 86400 IN RRSIG   NSEC5 16 3 86400 (
        20170412024301 20170313024301 5137 example.org. fjTcoRKgdML1
```

A.5. Wildcard No Data Example

Consider a query for a type MX record for foo.a.example.org.

The server must prove the following facts:

- o Existence of wildcard at closest encloser *.a.example.org. for any type other than MX or CNAME.

- o Non-existence of the next closer name foo.a.example.org.

To do this, the server returns:

```
;; ->>HEADER<<- opcode: QUERY; status: NOERROR; id: 17332

;; QUESTION SECTION:
;; foo.a.example.org.           IN      MX

;; AUTHORITY SECTION:
example.org.       3600 IN SOA     a.example.org. hostmaster.example.org. (
            2010111214 21600 3600 604800 86400 )

example.org.       3600 IN RRSIG   SOA 16 2 3600 (
            20170412024301 20170313024301 5137 example.org. /rT231b1rH/p )
```

This is an NSEC5PROOF RR for *.a.example.com, with RDATA equal to the NSEC5 proof for *.a.example.com. Per Section 9.1.1, this NSEC5PROOF RR may be precomputed.

```
   *.a.example.org.  86400 IN NSEC5PROOF      48566 Aq38RWWPhbs/vtih
```

This is a signed NSEC5 RR "matching" *.a.example.org with its CNAME and MX Type Bits cleared and its TXT Type Bit set. This NSEC5 RR has its owner name equal to the NSEC5 hash of *.a.example.org. NSEC5 RRs are precomputed.

```
mpu6c4.example.org. 86400 IN NSEC5   48566 0 O4K89V TXT RRSIG

mpu6c4.example.org. 86400 IN RRSIG   NSEC5 16 3 86400 (
            20170412024301 20170313024301 5137 example.org. m3I75ttcWwVC )
```

This is an NSEC5PROOF RR for foo.a.example.com. This NSEC5PROOF RR must be computed on-the-fly.

```
   foo.a.example.org.  86400 IN NSEC5PROOF      48566 AjqF5FGGVso40Lda
```

This is a signed NSEC5 RR "covering" foo.a.example.org. The NSEC5 hash of foo.a.example.org is FORDMO, and sorts in canonical order between this covering NSEC5 RR's Owner Name (which is BAPROH) and Next Hashed Owner Name (which is JQBMG4). This proves the existence of the wildcard at closest encloser *.a.example.org. for any type other than MX or CNAME. NSEC5 RRs are precomputed.

```
baproh.example.org. 86400 IN NSEC5   48566 1 JQBMG4 A TXT RRSIG

baproh.example.org. 86400 IN RRSIG   NSEC5 16 3 86400 (
            20170412024301 20170313024301 5137 example.org. fjTcoRKgdML1 )
```

Appendix B. Change Log

Note to RFC Editor: if this document does not obsolete an existing RFC, please remove this appendix before publication as an RFC.

pre 00 - initial version of the document submitted to mailing list only

00 - fix NSEC5KEY rollover mechanism, clarify NSEC5PROOF RDATA, clarify inputs and outputs for NSEC5 proof and NSEC5 hash computation.

01 - Add Performance Considerations section.

02 - Add elliptic curve based VRF. Add measurement of response sizes based on empirical data.

03 - Mention precomputed NSEC5PROOF Values in Performance Considerations section.

04 - Edit Rationale, How NSEC5 Works, and Security Consideration sections for clarity. Edit Zone Signing section, adding precomputation of NSEC5PROOFs. Remove RSA-based NSEC5 specification. Rewrite Performance Considerations and Implementation Status sections.

05 - Remove appendix specifying VRFs and add reference to draft-goldbe-vrf. Add Appendix A.

06 - Editorial changes. Minor updates to Section 8.1.

07 - Updated reference to [I-D.irtf-cfrg-vrf], updated VRF ciphersuites.

Authors' Addresses

Jan Vcelak
CZ.NIC
Milesovska 1136/5
Praha 130 00
CZ

EMail: jan.vcelak@nic.cz

Sharon Goldberg
Boston University
111 Cummington St, MCS135
Boston, MA 02215
USA

EMail: goldbe@cs.bu.edu

Dimitrios Papadopoulos
HKUST
Clearwater Bay
Hong Kong

EMail: dipapado@ust.hk

Shumon Huque
Salesforce
2550 Wasser Terr
Herndon, VA 20171
USA

EMail: shuque@gmail.com

David C Lawrence
Dyn
150 Dow Street, Tower Two
Manchester, NH 03101
USA

EMail: tale@dd.org