Stream: Internet Engineering Task Force (IETF)
RFC: 8976
Category: Standards Track
Published: January 2021
ISSN: 2070-1721
Authors: D. Wessels, *Verisign*; P. Barber, *Verisign*; M. Weinberg, *Amazon*; W. Kumari, *Google*; W. Hardaker, *USC/ISI*

RFC 8976
Message Digest for DNS Zones

Abstract

This document describes a protocol and new DNS Resource Record that provides a cryptographic message digest over DNS zone data at rest. The ZONEMD Resource Record conveys the digest data in the zone itself. When used in combination with DNSSEC, ZONEMD allows recipients to verify the zone contents for data integrity and origin authenticity. This provides assurance that received zone data matches published data, regardless of how the zone data has been transmitted and received. When used without DNSSEC, ZONEMD functions as a checksum, guarding only against unintentional changes.

ZONEMD does not replace DNSSEC: DNSSEC protects individual RRsets (DNS data with fine granularity), whereas ZONEMD protects a zone's data as a whole, whether consumed by authoritative name servers, recursive name servers, or any other applications.

As specified herein, ZONEMD is impractical for large, dynamic zones due to the time and resources required for digest calculation. However, the ZONEMD record is extensible so that new digest schemes may be added in the future to support large, dynamic zones.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8976.

Copyright Notice

Table of Contents

1. Introduction

In the DNS, a zone is the collection of authoritative resource records (RRs) sharing a common origin ([RFC8499]). Zones are often stored as files in the so-called "master file format" ([RFC1034]). Zones are generally distributed among name servers using the zone transfer (AXFR) ([RFC5936]) and incremental zone transfer (IXFR) ([RFC1995]) protocols. They can also be distributed outside of the DNS with any file transfer protocol such as FTP, HTTP, and rsync, or even as email attachments. Currently, there is no standard way to compute a hash or message digest for a stand-alone zone.

This document specifies an RR type that provides a cryptographic message digest of the data in a zone. It allows a receiver of the zone to verify the zone's integrity and authenticity when used in combination with DNSSEC. The digest RR is a part of the zone itself, allowing verification of the zone, no matter how it is transmitted. The digest uses the wire format of zone data in a canonical ordering. Thus, it is independent of presentation format such as whitespace, capitalization, and comments.

This specification is **OPTIONAL** to implement by both publishers and consumers of zone data.

1.1. Motivation

The primary motivation for this protocol enhancement is the desire to verify the data integrity and origin authenticity of a stand-alone zone, regardless of how it is transmitted. A consumer of zone data should be able to verify that it is as published by the zone operator.

Note, however, that integrity and authenticity can only be assured when the zone is signed. DNSSEC provides three strong security guarantees relevant to this protocol:

1. whether or not to expect DNSSEC records in the zone,
2. whether or not to expect a ZONEMD record in a signed zone, and

3. whether or not the ZONEMD record has been altered since it was signed.

A secondary motivation is to provide the equivalent of a checksum, allowing a zone recipient to check for unintended changes and operational errors such as accidental truncation.

1.2. Alternative Approaches

One approach to preventing data tampering and corruption is to secure the distribution channel. The DNS has a number of features that are already used for channel security. Perhaps the most widely used is DNS transaction signatures (TSIGs) ([RFC8945]). A TSIG uses shared secret keys and a message digest to protect individual query and response messages. It is generally used to authenticate and validate UPDATE ([RFC2136]), AXFR ([RFC5936]), and IXFR ([RFC1995]) messages.

DNS Request and Transaction Signatures (SIG(0)) ([RFC2931]) is another protocol extension that authenticates individual DNS transactions. Whereas SIG records normally cover specific RR types, SIG(0) is used to sign an entire DNS message. Unlike TSIG, SIG(0) uses public key cryptography rather than shared secrets.

The Transport Layer Security protocol suite also provides channel security. The DPRIVE Working Group is in the process of specifying DNS Zone Transfer-over-TLS ([DPRIVE-XFR-OVER-TLS]). One can also easily imagine the distribution of zones over HTTPS-enabled web servers as well as DNS-over-HTTPS ([RFC8484]).

Unfortunately, the protections provided by these channel security techniques are (in practice) ephemeral and are not retained after the data transfer is complete. They ensure that the client receives the data from the expected server and that the data sent by the server is not modified during transmission. However, they do not guarantee that the server transmits the data as originally published and do not provide any methods to verify data that is read after transmission is complete. For example, a name server loading saved zone data upon restart cannot guarantee that the on-disk data has not been modified. Such modification could be the result of an accidental corruption of the file or perhaps an incomplete saving of the file ([DISK-FULL-FAILURE]). For these reasons, it is preferable to protect the integrity of the data itself.

Why not simply rely on DNSSEC, which provides certain data security guarantees? For zones that are signed, a recipient could validate all of the signed RRsets. Additionally, denial-of-existence records prove that RRsets have not been added or removed. However, delegations (non-apex NS records) are not signed by DNSSEC and neither are any glue records. ZONEMD protects the integrity of delegation, glue, and other records that are not otherwise covered by DNSSEC. Furthermore, zones that employ NSEC3 with Opt-Out ([RFC5155]) are susceptible to the removal or addition of names between the signed nodes. Whereas DNSSEC primarily protects consumers of DNS response messages, this protocol protects consumers of zones.

There are existing tools and protocols that provide data security, such as OpenPGP ([RFC4880]) and S/MIME ([RFC8551]). In fact, the internic.net site publishes Pretty Good Privacy (PGP) signatures alongside the root zone and other files available there. However, this is a detached signature with no strong association to the corresponding zone file other than its timestamp.

Attached signatures are of course possible, but these necessarily change the format of the file being distributed; a zone signed with OpenPGP or S/MIME no longer looks like a DNS zone and could not directly be loaded into a name server. Once loaded, the signature data is lost, so it cannot be further propagated.

It seems the desire for data security in DNS zones was envisioned as far back as 1997. [RFC2065] is an obsoleted specification of the first generation DNSSEC Security Extensions. It describes a zone transfer signature, identified as the AXFR SIG, which is similar to the technique proposed by this document. That is, it proposes ordering all (signed) RRsets in a zone, hashing their contents, and then signing the zone hash. The AXFR SIG is described only for use during zone transfers. It did not postulate the need to validate zone data distributed outside of the DNS. Furthermore, its successor, [RFC2535], omits the AXFR SIG while at the same time introducing an IXFR SIG. (Note: RFC 2535 was obsoleted by RFCs 4033, 4034, and 4035.)

1.3. Design Overview

This document specifies a new Resource Record type to convey a message digest of the content of a zone. The digest is calculated at the time of zone publication. If the zone is signed with DNSSEC, any modifications of the digest can be detected. The procedures for digest calculation and DNSSEC signing are similar. Both require data to be processed in a well-defined order and format. It may be possible to perform DNSSEC signing and digest calculation in parallel.

The zone digest is designed to be used on zones that have infrequent updates. As specified herein, the digest is recalculated over the entire zone content each time the zone is updated. This specification does not provide an efficient mechanism for updating the digest on incremental updates of zone data. It is, however, extensible so that future schemes may be defined to support efficient incremental digest updates.

It is expected that verification of a zone digest will be implemented in name server software. That is, a name server can verify the zone data it was given and refuse to serve a zone that fails verification. For signed zones, the name server needs a trust anchor to perform DNSSEC validation. For signed non-root zones, the name server may need to send queries to validate a chain of trust. Digest verification could also be performed externally.

1.4. Use Cases

1.4.1. Root Zone

The root zone ([InterNIC]) is one of the most widely distributed DNS zones on the Internet, served by more than 1000 separate instances ([ROOT-SERVERS]) at the time of this writing. Additionally, many organizations configure their own name servers to serve the root zone locally. Reasons for doing so include privacy and reduced access time. [RFC8806] describes one way to do this. As the root zone spreads beyond its traditional deployment boundaries, the verification of the completeness of the zone contents becomes more important.

1.4.2. Providers, Secondaries, and Anycast

Since its very early days, the developers of the DNS recognized the importance of secondary name servers and service diversity. However, modern DNS service has complex provisioning that includes multiple third-party providers ([RFC8901]) and hundreds of anycast instances ([RFC3258]). Instead of a simple primary-to-secondary zone distribution system, today it is possible to have multiple levels, multiple parties, and multiple protocols involved in the distribution of zone data. This complexity introduces new places for problems to arise. The zone digest protects the integrity of data that flows through such systems.

1.4.3. Response Policy Zones

A Response Policy Zone (RPZ) is "a mechanism to introduce a customized policy in Domain Name System servers, so that recursive resolvers return possibly modified results" ([RPZ]). The policy information is carried inside specially constructed DNS zones. A number of companies provide RPZ feeds, which are consumed by name server and firewall products. While RPZs can be signed with DNSSEC, the data is not queried directly and would not be subject to DNSSEC validation.

1.4.4. Centralized Zone Data Service

ICANN operates the Centralized Zone Data Service ([CZDS]), which is a repository of top-level domain zone files. Users that have been granted access are then able to download zone data. Adding a zone digest to these would provide CZDS users with assurances that the data has not been modified between origination and retrieval. Note that ZONEMD could be added to zone data supplied to CZDS without requiring it to be present in the zone data served by production name servers, since the digest is inherently attached to the specific copy of the zone.

1.4.5. General Purpose Comparison Check

Since the zone digest calculation does not depend on presentation format, it could be used to compare multiple copies of a zone received from different sources, or copies generated by different processes. In this case, it serves as a checksum and can be useful even for unsigned zones.

1.5. Terminology

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The terms Private Use, Reserved, Unassigned, and Specification Required are to be interpreted as defined in [RFC8126].

2. The ZONEMD Resource Record

This section describes the ZONEMD Resource Record, including its fields, wire format, and presentation format. The Type value for the ZONEMD RR is 63. The ZONEMD RR is class independent. The RDATA of the resource record consists of four fields: Serial, Scheme, Hash Algorithm, and Digest.

2.1. Non-apex ZONEMD Records

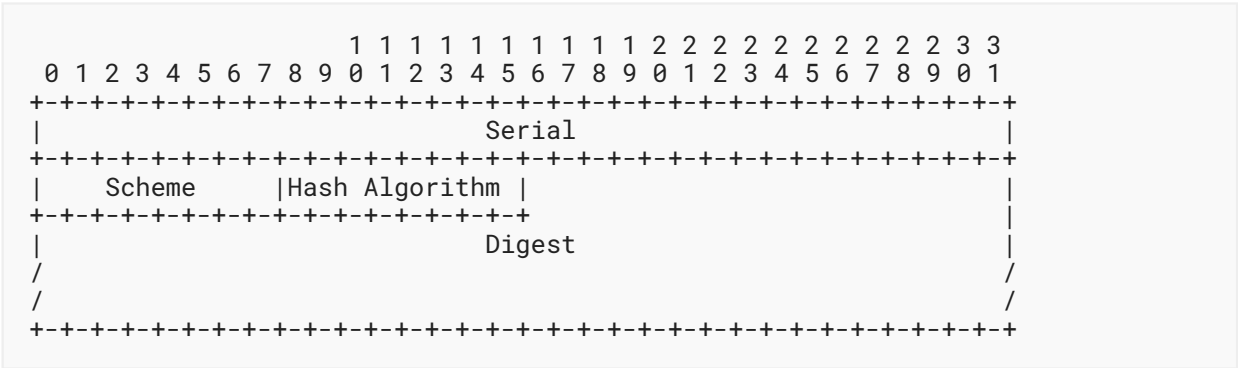
This document specifies ZONEMD RRs located at the zone apex. Non-apex ZONEMD RRs are not forbidden, but have no meaning in this specification. Non-apex ZONEMD RRs **MUST NOT** be used for verification.

During digest calculation, non-apex ZONEMD RRs are treated as ordinary RRs. They are digested as is, and the RR is not replaced by a placeholder RR.

Unless explicitly stated otherwise, "ZONEMD" always refers to apex records throughout this document.

2.2. ZONEMD RDATA Wire Format

The ZONEMD RDATA wire format is encoded as follows:



2.2.1. The Serial Field

The Serial field is a 32-bit unsigned integer in network byte order. It is the serial number from the zone's SOA record ([RFC1035], Section 3.3.13) for which the zone digest was generated.

It is included here to clearly bind the ZONEMD RR to a particular version of the zone's content. Without the serial number, a stand-alone ZONEMD digest has no obvious association to any particular instance of a zone.

2.2.2. The Scheme Field

The Scheme field is an 8-bit unsigned integer that identifies the methods by which data is collated and presented as input to the hashing function.

Herein, SIMPLE, with Scheme value 1, is the only standardized Scheme defined for ZONEMD records and it **MUST** be supported by implementations. The "ZONEMD Schemes" registry is further described in Section 5.

Scheme values 240-254 are allocated for Private Use.

2.2.3. The Hash Algorithm Field

The Hash Algorithm field is an 8-bit unsigned integer that identifies the cryptographic hash algorithm used to construct the digest.

Herein, SHA384 ([RFC6234]), with Hash Algorithm value 1, is the only standardized Hash Algorithm defined for ZONEMD records that **MUST** be supported by implementations. When SHA384 is used, the size of the Digest field is 48 octets. The result of the SHA384 digest algorithm **MUST NOT** be truncated, and the entire 48-octet digest is published in the ZONEMD record.

SHA512 ([RFC6234]), with Hash Algorithm value 2, is also defined for ZONEMD records and **SHOULD** be supported by implementations. When SHA512 is used, the size of the Digest field is 64 octets. The result of the SHA512 digest algorithm **MUST NOT** be truncated, and the entire 64-octet digest is published in the ZONEMD record.

Hash Algorithm values 240-254 are allocated for Private Use.

The "ZONEMD Hash Algorithms" registry is further described in Section 5.

2.2.4. The Digest Field

The Digest field is a variable-length sequence of octets containing the output of the hash algorithm. The length of the Digest field is determined by deducting the fixed size of the Serial, Scheme, and Hash Algorithm fields from the RDATA size in the ZONEMD RR header.

The Digest field **MUST NOT** be shorter than 12 octets. Digests for the SHA384 and SHA512 hash algorithms specified herein are never truncated. Digests for future hash algorithms **MAY** be truncated but **MUST NOT** be truncated to a length that results in less than 96 bits (12 octets) of equivalent strength.

Section 3 describes how to calculate the digest for a zone. Section 4 describes how to use the digest to verify the contents of a zone.

2.3. ZONEMD Presentation Format

The presentation format of the RDATA portion is as follows:

- The Serial field is represented as an unsigned decimal integer.
- The Scheme field is represented as an unsigned decimal integer.
- The Hash Algorithm field is represented as an unsigned decimal integer.
- The Digest is represented as a sequence of case-insensitive hexadecimal digits. Whitespace is allowed within the hexadecimal text.

2.4. ZONEMD Example

The following example shows a ZONEMD RR in presentation format:

```
example.com. 86400 IN ZONEMD 2018031500 1 1 (
    FEBE3D4CE2EC2FFA4BA99D46CD69D6D29711E55217057BEE
    7EB1A7B641A47BA7FED2DD5B97AE499FAFA4F22C6BD647DE )
```

2.5. Including ZONEMD RRs in a Zone

The zone operator chooses an appropriate hash algorithm and scheme and includes the calculated zone digest in the apex ZONEMD RRset. The zone operator **MAY** choose any of the defined hash algorithms and schemes, including the Private Use code points.

The ZONEMD RRset **MAY** contain multiple records to support algorithm agility ([BCP201]). When multiple ZONEMD RRs are present, each **MUST** specify a unique Scheme and Hash Algorithm tuple. It is **RECOMMENDED** that a zone include only one ZONEMD RR, unless the zone operator is in the process of transitioning to a new scheme or hash algorithm.

3. Calculating the Digest

The algorithm described in this section is designed for the common case of offline DNSSEC signing. Slight deviations may be permitted or necessary in other situations, such as with unsigned zones or online DNSSEC signing. Implementations that deviate from the described algorithm are advised to ensure that it produces ZONEMD RRs, signatures, and denial-of-existence records that are identical to the ones generated by this procedure.

3.1. Add ZONEMD Placeholder

In preparation for calculating the zone digest(s), any existing ZONEMD records (and covering RRSIGs) at the zone apex are first deleted.

Prior to calculation of the digest, and prior to signing with DNSSEC, one or more placeholder ZONEMD records are added to the zone apex. This ensures that denial-of-existence (NSEC, NSEC3) records are created correctly if the zone is signed with DNSSEC. If placeholders were not added prior to signing, the later addition of ZONEMD records would also require updating the Type Bit Maps field of any apex NSEC/NSEC3 RRs, which then invalidates the calculated digest value.

When multiple ZONEMD RRs are published in the zone, e.g., during an algorithm rollover, each **MUST** specify a unique Scheme and Hash Algorithm tuple.

It is **RECOMMENDED** that the TTL of the ZONEMD record match the TTL of the Start of Authority (SOA). However, the TTL of the ZONEMD record may be safely ignored during verification in all cases.

In the placeholder record, the Serial field is set to the current SOA Serial. The Scheme field is set to the value for the chosen collation scheme. The Hash Algorithm field is set to the value for the chosen hash algorithm. Since apex ZONEMD records are excluded from digest calculation, the value of the Digest field does not matter at this point in the process.

3.2. Optionally, Sign the Zone

Following the addition of placeholder records, the zone may be signed with DNSSEC. When the digest calculation is complete, and the ZONEMD record is updated, the signature(s) for the ZONEMD RRset **MUST** be recalculated and updated as well. Therefore, the signer is not required to calculate a signature over the placeholder record at this step in the process, but it is harmless to do so.

3.3. Scheme-Specific Processing

Herein, only the SIMPLE collation scheme is defined. Additional schemes may be defined in future updates to this document.

3.3.1. The SIMPLE Scheme

For the SIMPLE scheme, the digest is calculated over the zone as a whole. This means that a change to a single RR in the zone requires iterating over all RRs in the zone to recalculate the digest. SIMPLE is a good choice for zones that are small and/or stable, but it is probably not good for zones that are large and/or dynamic.

Calculation of a zone digest requires RRs to be processed in a consistent format and ordering. This specification uses DNSSEC's canonical on-the-wire RR format (without name compression) and ordering as specified in Sections 6.1, 6.2, and 6.3 of [RFC4034] with the additional provision that RRsets having the same owner name **MUST** be numerically ordered, in ascending order, by their numeric RR TYPE.

3.3.1.1. SIMPLE Scheme Inclusion/Exclusion Rules

When iterating over records in the zone, the following inclusion/exclusion rules apply:

- All records in the zone, including glue records, **MUST** be included unless excluded by a subsequent rule.
- Occluded data ([RFC5936], Section 3.5) **MUST** be included.
- If there are duplicate RRs with equal owner, class, type, and RDATA, only one instance is included ([RFC4034], Section 6.3) and the duplicates **MUST** be omitted.
- The placeholder apex ZONEMD RR(s) **MUST NOT** be included.
- If the zone is signed, DNSSEC RRs **MUST** be included, except:
- The RRSIG covering the apex ZONEMD RRset **MUST NOT** be included because the RRSIG will be updated after all digests have been calculated.

3.3.1.2. SIMPLE Scheme Digest Calculation

A zone digest using the SIMPLE scheme is calculated by concatenating all RRs in the zone, in the format and order described in Section 3.3.1 subject to the inclusion/exclusion rules described in Section 3.3.1.1, and then applying the chosen hash algorithm:

```
digest = hash( RR(1) | RR(2) | RR(3) | ... )

where "|" denotes concatenation.
```

3.4. Update ZONEMD RR

The calculated zone digest is inserted into the placeholder ZONEMD RR. Repeat for each digest if multiple digests are to be published.

If the zone is signed with DNSSEC, the RRSIG record(s) covering the ZONEMD RRset **MUST** then be added or updated. Because the ZONEMD placeholder was added prior to signing, the zone will already have the appropriate denial-of-existence (NSEC, NSEC3) records.

Some DNSSEC implementations (especially "online signing") might update the SOA serial number whenever a new signature is made. To preserve the calculated digest, generation of a ZONEMD signature **MUST NOT** also result in a change to the SOA serial number. The ZONEMD RR and the matching SOA **MUST** be published at the same time.

4. Verifying Zone Digest

The recipient of a zone that has a ZONEMD RR verifies the zone by calculating the digest as follows:

> Note: If multiple ZONEMD RRs are present in the zone, e.g., during an algorithm rollover, a match using any one of the recipient's supported Schemes and Hash Algorithms is sufficient to verify the zone. The verifier **MAY** ignore a ZONEMD RR if its Scheme and Hash Algorithm violates local policy.

1. The verifier **MUST** first determine whether or not to expect DNSSEC records in the zone. By examining locally configured trust anchors and, if necessary, querying for and validating Delegation Signer (DS) RRs in the parent zone, the verifier knows whether or not the zone to be verified should include DNSSEC keys and signatures. For zones where signatures are not expected, or if DNSSEC validation is not performed, digest verification continues at step 4 below.
2. For zones where signatures are expected, the existence of the apex ZONEMD record **MUST** be validated. If the DNSSEC data proves the ZONEMD RRset does not exist, digest verification cannot occur. If the DNSSEC data proves the ZONEMD does exist, but is not found in the zone, digest verification **MUST NOT** be considered successful.

3. For zones where signatures are expected, the SOA and ZONEMD RRsets **MUST** have valid signatures, chaining up to a trust anchor. If DNSSEC validation of the SOA or ZONEMD RRsets fails, digest verification **MUST NOT** be considered successful.
4. When multiple ZONEMD RRs are present, each **MUST** specify a unique Scheme and Hash Algorithm tuple. If the ZONEMD RRset contains more than one RR with the same Scheme and Hash Algorithm, digest verification for those ZONEMD RRs **MUST NOT** be considered successful.
5. Loop over all apex ZONEMD RRs and perform the following steps:

    a. The SOA Serial field **MUST** exactly match the ZONEMD Serial field. If the fields do not match, digest verification **MUST NOT** be considered successful with this ZONEMD RR.

    b. The Scheme field **MUST** be checked. If the verifier does not support the given scheme, verification **MUST NOT** be considered successful with this ZONEMD RR.

    c. The Hash Algorithm field **MUST** be checked. If the verifier does not support the given hash algorithm, verification **MUST NOT** be considered successful with this ZONEMD RR.

    d. The Digest field size **MUST** be checked. If the size of the given Digest field is smaller than 12 octets, or if the size is not equal to the size expected for the corresponding Hash Algorithm, verification **MUST NOT** be considered successful with this ZONEMD RR.

    e. The zone digest is computed over the zone data as described in Section 3.3 using the Scheme and Hash Algorithm for the current ZONEMD RR.

    f. The computed digest is compared to the received digest. If the two digest values match, verification is considered successful. Otherwise, verification **MUST NOT** be considered successful for this ZONEMD RR.

Each time zone verification is performed, the verifier **SHOULD** report the status as either successful or unsuccessful. When unsuccessful, the verifier **SHOULD** report the reason(s) that verification did not succeed.

5. IANA Considerations

5.1. ZONEMD RRtype

This document defines a new DNS RR type, ZONEMD, whose value 63 has been allocated by IANA from the "Resource Record (RR) TYPEs" subregistry of the "Domain Name System (DNS) Parameters" registry:

Type: ZONEMD
Value: 63
Meaning: Message Digest Over Zone Data
Reference: [RFC8976]

5.2. ZONEMD Scheme

IANA has created a new subregistry in the "Domain Name System (DNS) Parameters" registry as follows:

Registry Name: ZONEMD Schemes
Registration Procedure: Specification Required
Reference: [RFC8976]

Value	Description	Mnemonic	Reference
0	Reserved		[RFC8976]
1	Simple ZONEMD collation	SIMPLE	[RFC8976]
2-239	Unassigned		
240-254	Private Use	N/A	[RFC8976]
255	Reserved		[RFC8976]

Table 1: ZONEMD Scheme Registry

5.3. ZONEMD Hash Algorithms

IANA has created a new subregistry in the "Domain Name System (DNS) Parameters" registry as follows:

Registry Name: ZONEMD Hash Algorithms
Registration Procedure: Specification Required
Reference: [RFC8976]

Value	Description	Mnemonic	Reference
0	Reserved		[RFC8976]
1	SHA-384	SHA384	[RFC8976]
2	SHA-512	SHA512	[RFC8976]
3-239	Unassigned		
240-254	Private Use	N/A	[RFC8976]
255	Reserved		[RFC8976]

Table 2: ZONEMD Hash Algorithms Registry

6. Security Considerations

6.1. Using Zone Digest without DNSSEC

Users of ZONEMD with unsigned zones are advised that it provides no real protection against attacks. While zone digests can be used in the absence of DNSSEC, this only provides protection against accidental zone corruption such as transmission errors and truncation. When used in this manner, it effectively serves only as a checksum. For zones not signed with DNSSEC, an attacker can make any zone modifications appear to be valid by recomputing the Digest field of a ZONEMD RR.

6.2. Attacks against the Zone Digest

An attacker, whose goal is to modify zone content before it is used by the victim, may consider a number of different approaches.

The attacker might perform a downgrade attack to an unsigned zone. This is why Section 4 talks about determining whether or not to expect DNSSEC signatures for the zone in step 1.

The attacker might perform a downgrade attack by removing one or more ZONEMD records. Such a removal is detectable only with DNSSEC validation and is why Section 4 talks about checking denial-of-existence proofs in step 2 and signature validation in step 3.

The attacker might alter the Scheme, Hash Algorithm, or Digest fields of the ZONEMD record. Such modifications are detectable only with DNSSEC validation.

As stated in [BCP201], cryptographic algorithms age and become weaker as cryptanalysis techniques and computing resources improve with time. Implementors and publishers of zone digests should anticipate the need for algorithm agility on long timescales.

6.3. Use of Multiple ZONEMD Hash Algorithms

When a zone publishes multiple ZONEMD RRs, the overall security is only as good as the weakest hash algorithm in use. For this reason, Section 2 recommends only publishing multiple ZONEMD RRs when transitioning to a new scheme or hash algorithm. Once the transition is complete, the old scheme or hash algorithm should be removed from the ZONEMD RRset.

6.4. DNSSEC Timing Considerations

As with all DNSSEC signatures, the ability to perform signature validation of a ZONEMD record is limited in time. If the DS record(s) or trust anchors for the zone to be verified are no longer available, the recipient cannot validate the ZONEMD RRset. This could happen even if the ZONEMD signature is still current (not expired), since the zone's DS record(s) may have been withdrawn following a Key Signing Key (KSK) rollover.

For zones where it may be important to validate a ZONEMD RRset through its entire signature validity period, the zone operator should ensure that KSK rollover timing takes this into consideration.

6.5. Attacks Utilizing ZONEMD Queries

Nothing in this specification prevents clients from making, and servers from responding to, ZONEMD queries. Servers **SHOULD NOT** calculate zone digests dynamically (for each query) as this can be used as a CPU resource exhaustion attack.

ZONEMD responses could be used in a distributed denial-of-service amplification attack. The ZONEMD RR is moderately sized, much like the DS RR. A single ZONEMD RR contributes approximately 65 to 95 octets to a DNS response for digest types defined herein. Other RR types, such as DNS Public Key (DNSKEY), can result in larger amplification effects.

6.6. Resilience and Fragility

ZONEMD is used to detect incomplete or corrupted zone data prior to its use, thereby increasing resilience by not using corrupt data, but also introduces some denial-of-service fragility by making good data in a zone unavailable if some other data is missing or corrupt. Publishers and consumers of zones containing ZONEMD records should be aware of these trade-offs. While the intention is to secure the zone data, misconfigurations or implementation bugs are generally indistinguishable from intentional tampering and could lead to service failures when verification is performed automatically.

Zone publishers may want to deploy ZONEMD gradually perhaps by utilizing one of the Private Use hash algorithm code points listed in Section 5.3. Similarly, recipients may want to initially configure verification failures only as a warning, and later as an error after gaining experience and confidence with the feature.

7. Performance Considerations

This section is provided to make zone publishers aware of the performance requirements and implications of including ZONEMD RRs in a zone.

7.1. SIMPLE SHA384

As mentioned previously, the SIMPLE scheme may be impractical for use in zones that are either large or highly dynamic. Zone publishers should carefully consider the use of ZONEMD in such zones since it might cause consumers of zone data (e.g., secondary name servers) to expend resources on digest calculation. For such use cases, it is recommended that ZONEMD only be used when digest calculation time is significantly less than propagation times and update intervals.

The authors' implementation (Appendix B.1) includes an option to record and report CPU usage of its operation. The software was used to generate digests for more than 800 Top-Level Domain (TLD) zones available from [CZDS]. The table below summarizes the results for the SIMPLE scheme and SHA384 hash algorithm grouped by zone size. The Rate column is the mean amount of time per RR to calculate the digest, running on commodity hardware in early 2020.

Zone Size (RRs)	Rate (msec/RR)
10 - 99	0.00683
100 - 999	0.00551
1000 - 9999	0.00505
10000 - 99999	0.00602
100000 - 999999	0.00845
1000000 - 9999999	0.0108
10000000 - 99999999	0.0148

Table 3

For example, based on the above table, it takes approximately 0.13 seconds to calculate a SIMPLE SHA384 digest for a zone with 22,000 RRs, and about 2.5 seconds for a zone with 300,000 RRs.

These benchmarks attempt to emulate a worst-case scenario and take into account the time required to canonicalize the zone for processing. Each of the 800+ zones were measured three times and then averaged, with a different random sorting of the input data prior to each measurement.

8. Privacy Considerations

This specification has no impact on user privacy.

9. References

9.1. Normative References

[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>.

[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>.

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

9.2. Informative References

[BCP201] Housley, R., "Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms", BCP 201, RFC 7696, November 2015.

<https://www.rfc-editor.org/info/bcp201>

[CZDS] Internet Corporation for Assigned Names and Numbers (ICANN), "Centralized Zone Data Service", October 2018, <https://czds.icann.org/>.

[DISK-FULL-FAILURE] DENIC, "Background of the Partial Failure of the Name Service for .de Domains", May 2010, <https://web.archive.org/web/20100618032705/https://www.denic.de/en/denic-in-dialogue/news/2733.html>.

[DNS-TOOLS] "DNS tools for zone signature (file, pkcs11-hsm) and validation, and zone digest (ZONEMD)", commit 489de21, December 2020, <https://github.com/niclabs/dns-tools>.

[DPRIVE-XFR-OVER-TLS] Toorop, W., Dickinson, S., Sahib, S., Aras, P., and A. Mankin, "DNS Zone Transfer-over-TLS", Work in Progress, Internet-Draft, draft-ietf-dprive-xfr-over-tls-05, 20 January 2021, <https://tools.ietf.org/html/draft-ietf-dprive-xfr-over-tls-05>.

[InterNIC] InterNIC, "Index of ftp://rs.internic.net/", May 2018, <ftp://ftp.internic.net/domain/>.

[LDNS-ZONE-DIGEST] "Implementation of Message Digests for DNS Zones using the ldns library", commit 71c0cd1, January 2021, <https://github.com/verisign/ldns-zone-digest>.

[RFC1995] Ohta, M., "Incremental Zone Transfer in DNS", RFC 1995, DOI 10.17487/RFC1995, August 1996, <https://www.rfc-editor.org/info/rfc1995>.

[RFC2065] Eastlake 3rd, D. and C. Kaufman, "Domain Name System Security Extensions", RFC 2065, DOI 10.17487/RFC2065, January 1997, <https://www.rfc-editor.org/info/rfc2065>.

[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>.

[RFC2535] Eastlake 3rd, D., "Domain Name System Security Extensions", RFC 2535, DOI 10.17487/RFC2535, March 1999, <https://www.rfc-editor.org/info/rfc2535>.

[RFC2931] Eastlake 3rd, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, DOI 10.17487/RFC2931, September 2000, <https://www.rfc-editor.org/info/rfc2931>.

[RFC3258] Hardie, T., "Distributing Authoritative Name Servers via Shared Unicast Addresses", RFC 3258, DOI 10.17487/RFC3258, April 2002, <https://www.rfc-editor.org/info/rfc3258>.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, <https://www.rfc-editor.org/info/rfc4880>.

[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>.

[RFC5936] Lewis, E. and A. Hoenes, Ed., "DNS Zone Transfer Protocol (AXFR)", RFC 5936, DOI 10.17487/RFC5936, June 2010, <https://www.rfc-editor.org/info/rfc5936>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://www.rfc-editor.org/info/rfc8484>.

[RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <https://www.rfc-editor.org/info/rfc8499>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/info/rfc8551>.

[RFC8806] Kumari, W. and P. Hoffman, "Running a Root Server Local to a Resolver", RFC 8806, DOI 10.17487/RFC8806, June 2020, <https://www.rfc-editor.org/info/rfc8806>.

[RFC8901] Huque, S., Aras, P., Dickinson, J., Vcelak, J., and D. Blacka, "Multi-Signer DNSSEC Models", RFC 8901, DOI 10.17487/RFC8901, September 2020, <https://www.rfc-editor.org/info/rfc8901>.

[RFC8945] Dupont, F., Morris, S., Vixie, P., Eastlake 3rd, D., Gudmundsson, O., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", STD 93, RFC 8945, DOI 10.17487/RFC8945, November 2020, <https://www.rfc-editor.org/info/rfc8945>.

[ROOT-SERVERS] Root Server Operators, "root-servers.org", July 2018, <https://www.root-servers.org/>.

[RPZ] Wikipedia, "Response policy zone", May 2020, <https://en.wikipedia.org/w/index.php?title=Response_policy_zone&oldid=960043728>.

[ZONE-DIGEST-HACKATHON] "Prototype implementation of ZONEMD for the IETF 102 hackathon", commit 76ad7a7, August 2019, <https://github.com/shane-kerr/ZoneDigestHackathon>.

[ZONE-DIGEST-TESTS] IETF, "RFC 8976 ZONEMD Test Cases", January 2021, <https://trac.ietf.org/trac/dnsop/wiki/RFC8976ZONEMDTestCases>.

Appendix A. Example Zones with Digests

This appendix contains example zones with accurate ZONEMD records. These can be used to verify an implementation of the zone digest protocol. Additional and more extensive test cases can be found via the ZONEMD Tests Wiki ([ZONE-DIGEST-TESTS]) maintained by the IETF DNSOP Working Group.

A.1. Simple EXAMPLE Zone

Here, the EXAMPLE zone contains an SOA record, NS and glue records, and a ZONEMD record.

```
example.      86400  IN  SOA     ns1 admin 2018031900 (
                                 1800 900 604800 86400 )
              86400  IN  NS      ns1
              86400  IN  NS      ns2
              86400  IN  ZONEMD  2018031900 1 1 (
                                 c68090d90a7aed71
                                 6bc459f9340e3d7c
                                 1370d4d24b7e2fc3
                                 a1ddc0b9a87153b9
                                 a9713b3c9ae5cc27
                                 777f98b8e730044c )
ns1           3600   IN  A       203.0.113.63
ns2           3600   IN  AAAA    2001:db8::63
```

A.2. Complex EXAMPLE Zone

Here, the EXAMPLE zone contains duplicate RRs, an occluded RR, uppercase names, a wildcard, a multi-record RRset, a non-apex ZONEMD RR, and one out-of-zone RR.

```
example.       86400  IN  SOA     ns1 admin 2018031900 (
                                   1800 900 604800 86400 )
               86400  IN  NS      ns1
               86400  IN  NS      ns2
               86400  IN  ZONEMD  2018031900 1 1 (
                                   a3b69bad980a3504
                                   e1cffcb0fd6397f9
                                   3848071c93151f55
                                   2ae2f6b1711d4bd2
                                   d8b39808226d7b9d
                                   b71e34b72077f8fe )
ns1            3600   IN  A       203.0.113.63
NS2            3600   IN  AAAA    2001:db8::63
occluded.sub   7200   IN  TXT     "I'm occluded but must be digested"
sub            7200   IN  NS      ns1
duplicate      300    IN  TXT     "I must be digested just once"
duplicate      300    IN  TXT     "I must be digested just once"
foo.test.      555    IN  TXT     "out-of-zone data must be excluded"
UPPERCASE      3600   IN  TXT     "canonicalize uppercase owner names"
*              777    IN  PTR     dont-forget-about-wildcards
mail           3600   IN  MX      20 MAIL1
mail           3600   IN  MX      10 Mail2.Example.
sortme         3600   IN  AAAA    2001:db8::5:61
sortme         3600   IN  AAAA    2001:db8::3:62
sortme         3600   IN  AAAA    2001:db8::4:63
sortme         3600   IN  AAAA    2001:db8::1:65
sortme         3600   IN  AAAA    2001:db8::2:64
non-apex       900    IN  ZONEMD  2018031900 1 1 (
                                   616c6c6f77656420
                                   6275742069676e6f
                                   7265642e20616c6c
                                   6f77656420627574
                                   2069676e6f726564
                                   2e20616c6c6f7765 )
```

A.3. EXAMPLE Zone with Multiple Digests

Here, the EXAMPLE zone contains multiple ZONEMD records. It has both SHA384 and SHA512 digests using the SIMPLE scheme. It also includes ZONEMD records with Scheme and Hash Algorithm values in the private range (240-254). These additional private-range digests are not verifiable.

```
example.      86400  IN  SOA     ns1 admin 2018031900 (
                                 1800 900 604800 86400 )
example.      86400  IN  NS      ns1.example.
example.      86400  IN  NS      ns2.example.
example.      86400  IN  ZONEMD  2018031900 1 1 (
                                 62e6cf51b02e54b9
                                 b5f967d547ce4313
                                 6792901f9f88e637
                                 493daaf401c92c27
                                 9dd10f0edb1c56f8
                                 080211f8480ee306 )
example.      86400  IN  ZONEMD  2018031900 1 2 (
                                 08cfa1115c7b948c
                                 4163a901270395ea
                                 226a930cd2cbcf2f
                                 a9a5e6eb85f37c8a
                                 4e114d884e66f176
                                 eab121cb02db7d65
                                 2e0cc4827e7a3204
                                 f166b47e5613fd27 )
example.      86400  IN  ZONEMD  2018031900 1 240 (
                                 e2d523f654b9422a
                                 96c5a8f44607bbee )
example.      86400  IN  ZONEMD  2018031900 241 1 (
                                 e1846540e33a9e41
                                 89792d18d5d131f6
                                 05fc283e )
ns1.example.  3600   IN  A       203.0.113.63
ns2.example.  86400  IN  TXT     "This example has multiple digests"
NS2.EXAMPLE.  3600   IN  AAAA    2001:db8::63
```

A.4. The URI.ARPA Zone

The following sample zone is the URI.ARPA zone retrieved 2021-01-21. Note this sample zone has been re-signed with unpublished keys, so that the added ZONEMD RR also has a signature.

```
uri.arpa.       3600    IN      SOA     sns.dns.icann.org. (
   noc.dns.icann.org. 2018100702 10800 3600 1209600 3600 )
uri.arpa.       3600    IN      RRSIG   SOA 8 2 3600 (
    20210217232440 20210120232440 37444 uri.arpa.
    GzQw+QzwLDJr13REPGVmpEChjD1D2XlX0ie1DnWHpgaEw1E/dhs3lCN3+B
    mHd4Kx3tffTRgiyq65HxR6feQ5v7VmAifjyXUYB1DZur1eP5q0Ms2ygCB3
    byoeMgCNsFS1oKZ2LdzNBRpy3oace8xQn1SpmHGfyrsgg+WbHKCT1dY= )
uri.arpa.       86400   IN      NS      a.iana-servers.net.
uri.arpa.       86400   IN      NS      b.iana-servers.net.
uri.arpa.       86400   IN      NS      c.iana-servers.net.
uri.arpa.       86400   IN      NS      ns2.lacnic.net.
uri.arpa.       86400   IN      NS      sec3.apnic.net.
uri.arpa.       86400   IN      RRSIG   NS 8 2 86400 (
    20210217232440 20210120232440 37444 uri.arpa.
    M+Iei2lcewWGaMtkPlrhM9FpUAHXFkCHTVpeyrjxjEONeNgKtHZor5e4V4
    qJBOzNqo8go/qJpWlFBm+T5Hn3asaBZVstFIYky38/C8UeRLPKq1hTTHAR
    YUlFrexr5fMtSUAVOgOQPSBfH3xBq/BgSccTdRb9clD+HE7djpqrLS4= )
uri.arpa.       600     IN      MX      10 pechora.icann.org.
uri.arpa.       600     IN      RRSIG   MX 8 2 600 (
    20210217232440 20210120232440 37444 uri.arpa.
    kQAJQivmv6A5hqYBK8h6Z13ESY69gmosXwKI6WE09I8RFetfrxr24ecdnY
    d0lpnDtgNNSoHkYRSOoB+C4+zuJsoyAAzGo9uoWMWj97/2xeGhf3PTC9me
    Q9Ohi6hul9By7OR76XYmGhdWX8PBi60RUmZ1guslFBfQ8izwPqzuphs= )
uri.arpa.       3600    IN      DNSKEY  256 3 8 (
    AwEAAbMxuFuLeVDuOwIMzYOTD/bTREjLflo7wOi6ieIJhqltEzgjNzmWJf
    9kGwwDmzxU7kbthMEhBNBZNn84zmcyRSCMzuStWveL7xmqqUlE3swL8kLO
    vdZvc75XnmpHrk3ndTyEb6eZM7slh2C63Oh6K8VR5VkiZAkEGg0uZIT3Nj
    sF )
uri.arpa.       3600    IN      DNSKEY  257 3 8 (
    AwEAAdkTaWkZtZuRh7/OobBUFxM+ytTst+bCu0r9w+rEwXD7GbDs0pIMhM
    enrZzoAvmv1fQxw2MGs6Ri6yPKfNULcFOSt9l8i6BVBLI+SKTY6XXeDUQp
    SEmSaxohHeRPMQFzpysfjxINp/L2rGtZ7yPmxY/XRiFPSO0myqwGJa9r06
    Zw9CHM5UDHKWV/E+zxPFq/I7CfPbrrzbUotBX7Z6Vh3Sarllbe8cGUB2UF
    NaTRgwB0TwDBPRD5ER3w2Dzbry9NhbElTr7vVfhaGWeOGuqAUXwlXEg6Cr
    NkmJXJ2F1Rzr9WHUzhp7uWxhAbmJREGfi2dEyPAbUAyCjBqhFaqglknvc= )
uri.arpa.       3600    IN      DNSKEY  257 3 8 (
    AwEAAenQaBoFmDmvRT+/H5oNbm0Tr5FmNRNDEun0Jpj/ELkzeUrTWhNpQm
    ZeIMC8I0kZ185tEvOnRvn8OvV39B17QIdrvvKGIh2HlgeDRCLolhaojfn2
    QM0DStjF/WWHpxJOmE6CIuvhqYEU37yoJscGAPpPVPzNvnL1HhYTaao1VR
    YWQ/maMrJ+bfHg+YX1N6M/8MnRjIKBif1FWjbCKvsn6dnuGGL9oCWYUFJ3
    DwofXuhgPyZMkzPc88YkJj5EMvbMH4wtelbCwC+ivx732l0w/rXJn0ciQS
    OgoeVvDio8dIJmWQITWQAuP+q/ZHFEFHPlrP3gvQh5mcVS48eLX71Bq7c= )
uri.arpa.       3600    IN      RRSIG   DNSKEY 8 2 3600 (
    20210217232440 20210120232440 12670 uri.arpa.
    DBE2gkKAoxJCfz47KKxzoImN/0AKArhIVHE7TyTwy0DdRPo44V5R+vL6th
    UxlQ1CJi2Rw0jwAXymx5Y3Q873pOEllH+4bJoIT4dmoBmPXfYWW7Clvw9U
    PKHRP0igKHmCVwIeBYDTU3gfLcMTbR4nEWPDN0GxlL1Mf7ITaC2Ioabo79
    Ip3M/MR8I3Vx/xZ4ZKKPHtLn3xUuJluPNanqJrED2gTslL2xWZ1tqjsAjJ
    v7JnJo2HJ8XVRB5zBto0IaJ2oBlqcjdcQ/0VlyoM8uOy1pDwHQ2BJl7322
    gNMHBP9HSiUPIOaIDNUCwW8eUcW6DIUk+s9u3GN1uTqwWzsYB/rA== )
uri.arpa.       3600    IN      RRSIG   DNSKEY 8 2 3600 (
    20210217232440 20210120232440 30577 uri.arpa.
    Kx6HwP4UlkGc1UZ7SERXtQjPajOF4iUvkwDj7MEG1xbQFB1KoJiEb/eiW0
    qmSWdIhMDv8myhgauejRLyJxwxz8HDRV4xOeHWnRGfWBk4XGYwkejVzOHz
    oIArVdUVRbr2JKigcTOoyFN+uu52cNB7hRYu7dH5y1hlc6UbOnzRpMtGxc
    gVyKQ+/ARbIqGG3pegdEOvV49wTPWEiyY65P2urqhvnRg5ok/jzwAdMx4X
    Gshiib7Ojq0sRVl2ZIzj4rFgY/qsSO8SEXEhMo2VuSkoJNiofVzYoqpxEe
```

```
    GnANkIT7Tx2xJL1BWyJxyc7E8Wr2QSgCcc+rYL6IkHDtJGHy7TaQ== )
uri.arpa.       3600    IN      ZONEMD  2018100702 1 1 (
    0dbc3c4dbfd75777c12ca19c337854b1577799901307c482e9d91d5d15
    cd934d16319d98e30c4201cf25a1d5a0254960 )
uri.arpa.       3600    IN      RRSIG   ZONEMD 8 2 3600 (
    20210217232440 20210120232440 37444 uri.arpa.
    QDo4XZcL3HMyn8aAHyCUsu/Tqj4Gkth8xY1EqByOb8XOTwVtA4ZNQORE1s
    iqNqjtJUbeJPtJSbLNqCL7rCq0CzNNnBscv6IIf4gnqJZjlGtHO30ohXtK
    vEc4z7SU3IASsi6bB3nLmEAyERdYSeU6UBfx8vatQDIRhkgEnnWUTh4= )
uri.arpa.       3600    IN      NSEC    ftp.uri.arpa. (
    NS SOA MX RRSIG NSEC DNSKEY ZONEMD )
uri.arpa.       3600    IN      RRSIG   NSEC 8 2 3600 (
    20210217232440 20210120232440 37444 uri.arpa.
    dU/rXLM/naWd1+1PiWiYVaNJyCkiuyZJSccr91pJI673T8r3685B4ODMYF
    afZRboVgwnl3ZrXddY6xOhZL3n9V9nxXZwjLJ2HJUojFoKcXTlpnUyYUYv
    VQ2kj4GHAo6fcGCEp5QFJ2KbCpeJoS+PhKGRRx28icCiNT4/uXQvO2E= )
ftp.uri.arpa.   604800  IN      NAPTR   0 0 "" "" (
    "!^ftp://([^:/?#]*).*$!\\1!i" . )
ftp.uri.arpa.   604800  IN      RRSIG   NAPTR 8 3 604800 (
    20210217232440 20210120232440 37444 uri.arpa.
    EygekDgl+Lyyq4NMSEpPyOrOywYf9Y3FAB4v1DT44J3R5QGidaH8l7ZFjH
    oYFI8sY64iYOCV4sBnX/dh6C1L5NgpY+8l5065Xu3vvjyzbtuJ2k6YYwJr
    rCbvl5DDn53zAhhO2hL9uLgyLraZGi9i7TFGd0sm3zNyUF/EVL0CcxU= )
ftp.uri.arpa.   3600    IN      NSEC    http.uri.arpa. (
    NAPTR RRSIG NSEC )
ftp.uri.arpa.   3600    IN      RRSIG   NSEC 8 3 3600 (
    20210217232440 20210120232440 37444 uri.arpa.
    pbP4KxevPXCu/bDqcvXiuBppXyFEmtHyiy0eAN5gS7mi6mp9Z9bWFjx/Ld
    H9+6oFGYa5vGmJ5itu/4EDMe8iQeZbI8yrpM4TquB7RR/MGfBnTd8S+sjy
    QtlRYG7yqEu77Vd78Fme22BKPJ+MVqjS0JHMUE/YUGomPkAjLJJwwGw= )
http.uri.arpa.  604800  IN      NAPTR   0 0 "" "" (
    "!^http://([^:/?#]*).*$!\\1!i" . )
http.uri.arpa.  604800  IN      RRSIG   NAPTR 8 3 604800 (
    20210217232440 20210120232440 37444 uri.arpa.
    eTqbWvt1GvTeXozuvm4ebaAfkXFQKrtdu0cEiExto80sHIiCbO0WL8UDa/
    J3cDivtQca7LgUbOb6c17NESsrsVkc6zNPx5RK2tG7ZQYmhYmtqtfg1oU5
    BRdHZ5TyqIXcHlw9Blo2pir1Y9IQgshhD7UOGkbkEmvB1Lrd0aHhAAg= )
http.uri.arpa.  3600    IN      NSEC    mailto.uri.arpa. (
    NAPTR RRSIG NSEC )
http.uri.arpa.  3600    IN      RRSIG   NSEC 8 3 3600 (
    20210217232440 20210120232440 37444 uri.arpa.
    R9rlNzw1CVz2N08q6DhULzcsuUm0UKcPaGAWEU40tr81jEDHsFHNM+khCd
    OI8nDstzA42aee4rwCEgijxJpRCcY9hrO1Ysrrr2fdqNz60JikMdarvU5O
    0p0VXeaaJDfJQT44+o+YXaBwI7Qod3FTMx7aRib8i7istvPm1Rr7ixA= )
mailto.uri.arpa.        604800  IN      NAPTR   0 0 "" "" (
    "!^mailto:(.*)@(.*)$!\\2!i" . )
mailto.uri.arpa.        604800  IN      RRSIG   NAPTR 8 3 604800 (
    20210217232440 20210120232440 37444 uri.arpa.
    Ch2zTG2F1plEvQPyIH4Yd80XXLjXOPvMbiqDjpJBcnCJsV8QF7kr0wTLnU
    T3dB+asQudOjPyzaHGwFlMzmrrAsszN4XAMJ6htDtFJdsgTMP/NkHhYRSm
    Vv6rLeAhd+mVfObY12M//b/GGVTjeUI/gJaLW0fLVZxr1Fp5U5CRjyw= )
mailto.uri.arpa.        3600    IN      NSEC    urn.uri.arpa. (
    NAPTR RRSIG NSEC )
mailto.uri.arpa.        3600    IN      RRSIG   NSEC 8 3 3600 (
    20210217232440 20210120232440 37444 uri.arpa.
    fQUbSIE6E7JDi2rosah4SpCOTrKufeszFyj5YEavbQuYlQ5cNFvtm8KuE2
    xXMRgRI4RGvM2leVqcoDw5hS3m2pOJLxH8l2WE72YjYvWhvnwc5Rofe/8y
    B/vaSK9WCnqN8y2q6Vmy73AGP0fuiwmuBra7LlkOiqmyx3amSFizwms= )
```

```
urn.uri.arpa.   604800  IN      NAPTR   0 0 "" "" (
    "/urn:([^:]+)/\\1/i" . )
urn.uri.arpa.   604800  IN      RRSIG   NAPTR 8 3 604800 (
    20210217232440 20210120232440 37444 uri.arpa.
    CVt2Tgz0e5ZmaSXqRfNys/8OtVCk9nfP0zhezhN8Bo6MDt6yyKZ2kEEWJP
    jkN7PCYHjO8fGjnUn0AHZI2qBNv7PKHcpR42VY03q927q85a65weOO1YE0
    vPYMzACpua9TOtfNnynM2Ws0uN9URxUyvYkXBdqOC81N3sx1dVELcwc= )
urn.uri.arpa.   3600    IN      NSEC    uri.arpa. NAPTR RRSIG NSEC
urn.uri.arpa.   3600    IN      RRSIG   NSEC 8 3 3600 (
    20210217232440 20210120232440 37444 uri.arpa.
    JuKkMiC3/j9iM3V8/izcouXWAVGnSZjkOgEgFPhutMqoylQNRcSkbEZQzF
    K8B/PIVdzZF0Y5xkO6zaKQjOzz6OkSaNPIo1a7Vyyl3wDY/uLCRRAHRJfp
    knuY7O+AUNXvVVIEYJqZggd4kl/Rjh1GTzPYZTRrVi5eQidI1LqCOeg= )
```

A.5. The ROOT-SERVERS.NET Zone

The following sample zone is the ROOT-SERVERS.NET zone retrieved 2018-10-21.

```
root-servers.net.     3600000 IN  SOA     a.root-servers.net. (
    nstld.verisign-grs.com. 2018091100 14400 7200 1209600 3600000 )
root-servers.net.     3600000 IN  NS      a.root-servers.net.
root-servers.net.     3600000 IN  NS      b.root-servers.net.
root-servers.net.     3600000 IN  NS      c.root-servers.net.
root-servers.net.     3600000 IN  NS      d.root-servers.net.
root-servers.net.     3600000 IN  NS      e.root-servers.net.
root-servers.net.     3600000 IN  NS      f.root-servers.net.
root-servers.net.     3600000 IN  NS      g.root-servers.net.
root-servers.net.     3600000 IN  NS      h.root-servers.net.
root-servers.net.     3600000 IN  NS      i.root-servers.net.
root-servers.net.     3600000 IN  NS      j.root-servers.net.
root-servers.net.     3600000 IN  NS      k.root-servers.net.
root-servers.net.     3600000 IN  NS      l.root-servers.net.
root-servers.net.     3600000 IN  NS      m.root-servers.net.
a.root-servers.net.   3600000 IN  AAAA    2001:503:ba3e::2:30
a.root-servers.net.   3600000 IN  A       198.41.0.4
b.root-servers.net.   3600000 IN  MX      20 mail.isi.edu.
b.root-servers.net.   3600000 IN  AAAA    2001:500:200::b
b.root-servers.net.   3600000 IN  A       199.9.14.201
c.root-servers.net.   3600000 IN  AAAA    2001:500:2::c
c.root-servers.net.   3600000 IN  A       192.33.4.12
d.root-servers.net.   3600000 IN  AAAA    2001:500:2d::d
d.root-servers.net.   3600000 IN  A       199.7.91.13
e.root-servers.net.   3600000 IN  AAAA    2001:500:a8::e
e.root-servers.net.   3600000 IN  A       192.203.230.10
f.root-servers.net.   3600000 IN  AAAA    2001:500:2f::f
f.root-servers.net.   3600000 IN  A       192.5.5.241
g.root-servers.net.   3600000 IN  AAAA    2001:500:12::d0d
g.root-servers.net.   3600000 IN  A       192.112.36.4
h.root-servers.net.   3600000 IN  AAAA    2001:500:1::53
h.root-servers.net.   3600000 IN  A       198.97.190.53
i.root-servers.net.   3600000 IN  MX      10 mx.i.root-servers.org.
i.root-servers.net.   3600000 IN  AAAA    2001:7fe::53
i.root-servers.net.   3600000 IN  A       192.36.148.17
j.root-servers.net.   3600000 IN  AAAA    2001:503:c27::2:30
j.root-servers.net.   3600000 IN  A       192.58.128.30
k.root-servers.net.   3600000 IN  AAAA    2001:7fd::1
k.root-servers.net.   3600000 IN  A       193.0.14.129
l.root-servers.net.   3600000 IN  AAAA    2001:500:9f::42
l.root-servers.net.   3600000 IN  A       199.7.83.42
m.root-servers.net.   3600000 IN  AAAA    2001:dc3::35
m.root-servers.net.   3600000 IN  A       202.12.27.33
root-servers.net.     3600000 IN  SOA     a.root-servers.net. (
    nstld.verisign-grs.com. 2018091100 14400 7200 1209600 3600000 )
root-servers.net.     3600000 IN  ZONEMD  2018091100 1 1 (
    f1ca0ccd91bd5573d9f431c00ee0101b2545c97602be0a97
    8a3b11dbfc1c776d5b3e86ae3d973d6b5349ba7f04340f79 )
```

Appendix B. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of publication, and is inspired by the concepts described in RFC 7942.

Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

B.1. Authors' Implementation

The authors have an open-source implementation in C, using the ldns library ([LDNS-ZONE-DIGEST]). This implementation is able to perform the following functions:

- Read an input zone and output a zone with the ZONEMD placeholder.
- Compute the zone digest over the signed zone and update the ZONEMD record.
- Recompute DNSSEC signatures over the ZONEMD record.
- Verify the zone digest from an input zone.

This implementation does not:

- Perform DNSSEC validation of the ZONEMD record during verification.

B.2. Shane Kerr's Implementation

Shane Kerr wrote an implementation of this specification during the IETF 102 hackathon ([ZONE-DIGEST-HACKATHON]). This implementation is in Python and is able to perform the following functions:

- Read an input zone and output a zone with ZONEMD record.
- Verify the zone digest from an input zone.
- Output the ZONEMD record in its defined presentation format.

This implementation does not:

- Recompute DNSSEC signatures over the ZONEMD record.
- Perform DNSSEC validation of the ZONEMD record.

B.3. NIC Chile Lab's Implementation

NIC Chile Labs wrote an implementation of this specification as part of "dns-tools" suite ([DNS-TOOLS]), which besides digesting, can also sign and verify zones. This implementation is in Go and is able to perform the following functions:

- Compute zone digest over signed zone and update the ZONEMD record.
- Verify the zone digest from an input zone.
- Perform DNSSEC validation of the ZONEMD record during verification.
- Recompute DNSSEC signatures over the ZONEMD record.

Acknowledgments

The authors wish to thank David Blacka, Scott Hollenbeck, and Rick Wilhelm for providing feedback on early drafts of this document. Additionally, they thank Joe Abley, Mark Andrews, Ralph Dolmans, Donald Eastlake 3rd, Richard Gibson, Olafur Gudmundsson, Bob Harold, Paul Hoffman, Evan Hunt, Shumon Huque, Tatuya Jinmei, Mike St. Johns, Burt Kaliski, Shane Kerr, Matt Larson, Barry Leiba, John Levine, Ed Lewis, Matt Pounsett, Mukund Sivaraman, Petr Spacek, Ondrej Sury, Willem Toorop, Florian Weimer, Tim Wicinski, Wouter Wijngaards, Paul Wouters, and other members of the DNSOP Working Group for their input.

Authors' Addresses

Duane Wessels
Verisign
12061 Bluemont Way
Reston, VA 20190
United States of America
Phone: +1 703 948-3200
Email: dwessels@verisign.com
URI: https://verisign.com

Piet Barber
Verisign
12061 Bluemont Way
Reston, VA 20190
United States of America
Phone: +1 703 948-3200
Email: pbarber@verisign.com
URI: https://verisign.com

Matt Weinberg
Amazon
Email: matweinb@amazon.com
URI: https://amazon.com

Warren Kumari
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
United States of America
Email: warren@kumari.net

Wes Hardaker
USC/ISI
P.O. Box 382
Davis, CA 95617
United States of America
Email: ietf@hardakers.net