

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: March 29, 2021

D. Wessels
P. Barber
Verisign
M. Weinberg
Amazon
W. Kumari
Google
W. Hardaker
USC/ISI
September 25, 2020

Message Digest for DNS Zones
draft-ietf-dnsop-dns-zone-digest-11

Abstract

This document describes a protocol and new DNS Resource Record that provides a cryptographic message digest over DNS zone data. The ZONEMD Resource Record conveys the digest data in the zone itself. When a zone publisher includes a ZONEMD record, recipients can verify the zone contents for accuracy and completeness. This provides assurance that received zone data matches published data, regardless of how the zone data has been transmitted and received.

ZONEMD does not replace DNSSEC. Whereas DNSSEC protects individual RRsets (DNS data with fine granularity), 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 Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 29, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
1.1.	Motivation	4
1.2.	Alternative Approaches	4
1.3.	Design Overview	6
1.4.	Use Cases	6
1.4.1.	Root Zone	6
1.4.2.	Providers, Secondaries, and Anycast	6
1.4.3.	Response Policy Zones	7
1.4.4.	Centralized Zone Data Service	7
1.4.5.	General Purpose Comparison Check	7
1.5.	Terminology	7
2.	The ZONEMD Resource Record	7
2.1.	Non-apex ZONEMD Records	8
2.2.	ZONEMD RDATA Wire Format	8
2.2.1.	The Serial Field	8
2.2.2.	The Scheme Field	9
2.2.3.	The Hash Algorithm Field	9
2.2.4.	The Digest Field	9
2.3.	ZONEMD Presentation Format	10
2.4.	ZONEMD Example	10
3.	Calculating the Digest	10
3.1.	Add ZONEMD Placeholder	10
3.2.	Optionally Sign the Zone	11
3.3.	Scheme-Specific Processing	11
3.3.1.	The SIMPLE Scheme	11
3.3.1.1.	SIMPLE Scheme Inclusion/Exclusion Rules	11

3.3.1.2.	SIMPLE Scheme Digest Calculation	12
3.4.	Update ZONEMD RR	12
4.	Verifying Zone Digest	12
5.	IANA Considerations	14
5.1.	ZONEMD RRtype	14
5.2.	ZONEMD Scheme	14
5.3.	ZONEMD Hash Algorithm	15
6.	Security Considerations	16
6.1.	Attacks Against the Zone Digest	16
6.2.	DNSSEC Timing Considerations	16
6.3.	Attacks Utilizing ZONEMD Queries	16
6.4.	Resilience and Fragility	17
7.	Performance Considerations	17
7.1.	SIMPLE SHA384	17
8.	Privacy Considerations	18
9.	Acknowledgments	18
10.	Change Log	18
11.	References	24
11.1.	Normative References	24
11.2.	Informative References	25
Appendix A.	Example Zones With Digests	27
A.1.	Simple EXAMPLE Zone	27
A.2.	Complex EXAMPLE Zone	28
A.3.	EXAMPLE Zone with multiple digests	29
A.4.	The URI.ARPA Zone	29
A.5.	The ROOT-SERVERS.NET Zone	32
Appendix B.	Implementation Status	34
B.1.	Authors' Implementation	34
B.2.	Shane Kerr's Implementation	34
B.3.	NIC Chile Labs Implementation	35
Authors' Addresses	35

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 AXFR (zone transfer [RFC5936]), and IXFR (incremental zone transfer [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 verify the authenticity of 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 when used in combination with DNSSEC, its authenticity. 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.

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.

1.1. Motivation

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

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 (TSIG [[RFC2845](#)]). 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. One can easily imagine the distribution of zones over HTTPS-enabled web servers, as well as DNS-over-HTTPS [[RFC8484](#)], and perhaps even a future version of DNS-over-TLS ([[RFC7858](#)]).

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 incompletely saved file [[disk-full-failure](#)]. For these reasons, it is preferable to secure 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 are susceptible to the removal or addition of names between the signed nodes. Whereas DNSSEC is 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 [[RFC5751](#)]. In fact, the `internic.net` site publishes 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. Non-detached 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.

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 re-calculated over the entire zone content each time. This specification does not provide an efficient mechanism for updating the digest on incremental updates of zone data. It is, however, extensible so future schemes to support incremental zone digest algorithms (e.g. using Merkle trees) can be accommodated.

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 which 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 zone on the Internet, served by more than 1000 separate instances [[RootServers](#)] 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 which includes multiple third-party providers and hundreds of anycast instances. 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

DNS Response Policy Zones is "a method of expressing DNS response policy information inside specially constructed DNS zones..." [[RPZ](#)]. A number of companies provide RPZ feeds, which are consumed by name server and firewall products. While RPZ zones 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. ZONEMD could be added to CZDS zone data independently of the zone served by production name servers.

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.

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.

A zone MAY contain multiple ZONEMD RRs to support algorithm agility [RFC7696] and rollovers. 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 publisher is in the process of transitioning to a new Scheme or Hash Algorithm.

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:

```

          1 1 1 1 1 1 1 1 1 1 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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Serial                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Scheme   |Hash Algorithm |                                     |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               Digest                               |
/                                                           /
/                                                           /
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

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 in order to make DNS response messages of type ZONEMD meaningful. Without the serial number, a stand-alone ZONEMD digest has no 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 value 1, is the only standardized Scheme defined for ZONEMD records and it MUST be implemented. The Scheme 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 value 1, is the only standardized Hash Algorithm defined for ZONEMD records that MUST be implemented. 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 implemented. 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 Hash Algorithm 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 )
```

[3.](#) Calculating the Digest

[3.1.](#) Add ZONEMD Placeholder

In preparation for calculating the zone digest, 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 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 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 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 RRSset 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 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:

- o All records in the zone, including glue records, MUST be included.
- o Occluded data ([\[RFC5936\]](#) [Section 3.5](#)) MUST be included.

- o 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.
- o The placeholder ZONEMD RR(s) MUST NOT be included.
- o If the zone is signed, DNSSEC RRs MUST be included, except:
- o The RRSIG covering ZONEMD 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. 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. This is done by examining locally configured trust anchors, or querying for (and validating) DS RRs in the parent zone. For zones that are provably insecure, or if DNSSEC validation is not performed, digest verification continues at step 4 below.
2. For zones that are provably secure, the existence of the apex ZONEMD record MUST be verified. If the ZONEMD record provably does not exist, digest verification cannot occur. If the ZONEMD record does provably exist, but is not found in the zone, digest verification MUST NOT be considered successful.
3. For zones that are provably secure, the SOA and ZONEMD RRSets MUST have valid signatures, chaining up to a trust anchor. If DNSSEC validation of the SOA or ZONEMD records 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 RRSSet 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 and it SHOULD report that the RR's digest could not be verified due to an unsupported scheme.
 - 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 and it SHOULD report that the RR's digest could not be verified due to an unsupported algorithm.
 - 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 and the verifier SHOULD report that the

RR's digest could not be verified to to an incorrect digest length.

- 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.

[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: [this document]

[5.2.](#) ZONEMD Scheme

IANA is requested to create a new registry on the "Domain Name System (DNS) Parameters" web page as follows:

Registry Name: ZONEMD Schemes

Registration Procedure: Specification Required

Reference: [this document]

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

Table 1: ZONEMD Scheme Registry

5.3. ZONEMD Hash Algorithm

IANA is requested to create a new registry on the "Domain Name System (DNS) Parameters" web page as follows:

Registry Name: ZONEMD Hash Algorithms

Registration Procedure: Specification Required

Reference: [this document]

Value	Description	Mnemonic	Implementation Requirement	Reference
0	Reserved			
1	SHA-384	SHA384	MUST	[this document]
2	SHA-512	SHA512	SHOULD	[this document]
3-239	Unassigned			
240-254	Private Use	N/A	N/A	[his document]
255	Reserved			

Table 2: ZONEMD Hash Algorithm Registry

The IANA policy for assigning new values to the ZONEMD Hash Algorithm registry shall be Specification Required.

6. Security Considerations

6.1. Attacks Against the Zone Digest

The zone digest allows the recipient of a zone to verify its integrity. In conjunction with DNSSEC, the recipient can authenticate that it is as published by the zone originator.

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.

6.2. 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 RRSset. 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 KSK rollover.

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

6.3. 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 40 to 65

octets to a DNS response, for digest types defined herein. Other RR types, such as DNSKEY, can result in larger amplification effects.

6.4. 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 tradeoffs. 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 algorithms 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 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
+-----+	

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. 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, 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 DNS working group for their input.

10. Change Log

RFC Editor: Please remove this section before publication.

This section lists substantial changes to the document as it is being worked on.

From -00 to -01:

- o Removed requirement to sort by RR CLASS.

- o Added Kumari and Hardaker as coauthors.
- o Added Change Log section.
- o Minor clarifications and grammatical edits.

From -01 to -02:

- o Emphasize desire for data security over channel security.
- o Expanded motivation into its own subsection.
- o Removed discussion topic whether or not to include serial in ZONEMD.
- o Clarified that a zone's NS records always sort before the SOA record.
- o Clarified that all records in the zone must be digested, except as specified in the exclusion rules.
- o Added for discussion out-of-zone and occluded records.
- o Clarified that update of ZONEMD signature must not cause a serial number change.
- o Added persons to acknowledgments.

From -02 to -03:

- o Added recommendation to set ZONEMD TTL to SOA TTL.
- o Clarified that digest input uses uncompressed names.
- o Updated Implementations section.
- o Changed intended status from Standards Track to Experimental and added Scope of Experiment section.
- o Updated Motivation, Introduction, and Design Overview sections in response to working group discussion.
- o Gave ZONEMD digest types their own status, separate from DS digest types. Request IANA to create a registry.
- o Added Reserved field for future work supporting dynamic updates.
- o Be more rigorous about having just ONE ZONEMD record in the zone.

- o Expanded use cases.

From -03 to -04:

- o Added an appendix with example zones and digests.
- o Clarified that only apex ZONEMD RRs shall be processed.

From -04 to -05:

- o Made SHA384 the only supported ZONEMD digest type.
- o Disassociated ZONEMD digest types from DS digest types.
- o Updates to Introduction based on list feedback.
- o Changed "zone file" to "zone" everywhere.
- o Restored text about why ZONEMD has a Serial field.
- o Clarified ordering of RRsets having same owner to be numerically ascending.
- o Clarified that all duplicate RRs (not just SOA) must be suppressed in digest calculation.
- o Clarified that the Reserved field must be set to zero and checked for zero in verification.
- o Clarified that occluded data must be included.
- o Clarified procedure for verification, using temporary location for received digest.
- o Explained why Reserved field is 8-bits.
- o IANA Considerations section now more specific.
- o Added complex zone to examples.
- o

From -05 to -06:

- o RR type code 63 was assigned to ZONEMD by IANA.

From -06 to -07:

- o Fixed mistakes in ZONEMD examples.
- o Added private use Digest Type values 240-254.
- o Clarified that Digest field must not be empty.

From -07 to [draft-ietf-dnsop-dns-zone-digest-00](#):

- o Adopted by dnsop.
- o Clarified further that non-apex ZONEMD RRs have no meaning.
- o Changed "provably [un]signed" to "provably [in]secure".
- o Allow multiple ZONEMD RRs to support algorithm agility/rollovers.
- o Describe verification when there are multiple ZONEMD RRs.

From -00 to -01:

- o Simplified requirements around verifying multiple digests. Any one match is sufficient.
- o Updated implementation notes.
- o Both implementations produce expected results on examples given in this document.

From -01 to -02:

- o Changed the name of the Reserved field to Parameter.
- o Changed the name of Digest Type 1 from SHA384 to SHA384-STABLE.
- o The meaning of the Parameter field now depends on Digest Type.
- o No longer require Parameter field to be zero in verification.
- o Updated a rule from earlier versions that said multiple ZONEMD RRs were not allowed.

From -02 to -03:

- o Changed the name of Digest Type 1 from SHA384-STABLE to SHA384-SIMPLE.
- o Changed document status from Experimental to Standards Track.

- o Removed Scope of Experimentation section.

From -03 to -04:

- o Addressing WGLC feedback.
- o Changed from "Digest Type + Paramter" to "Scheme + Hash Algorithm". This should make it more obvious how ZONEMD can be expanded in the future with new schemes and hash algorithms, while sacrificing some of the flexibility that the Parameter was intended to provide.
- o Note: old RDATA fields: Serial, Digest Type, Parameter, Digest.
- o Note: new RDATA fields: Serial, Scheme, Hash Algorithm, Digest.
- o Add new IANA requirement for a Scheme registry.
- o Rearranged some sections and separated scheme-specific aspects from general aspects of digest calculation.
- o When discussing multiple ZONEMD RRs, allow for Scheme, as well as Hash Algorithm, transition.
- o Added Performance Considerations section with some benchmarks.
- o Further clarifications about non-apex ZONEMD RRs.
- o Clarified inclusion rule for duplicate RRs.
- o Removed or lowercased some inappropriately used [RFC 2119](#) key words.
- o Clarified that all ZONEMD RRs, even for unsupported hash algorithms, must be zeroized during digest calculation.
- o Added Resilience and Fragility to security considerations.
- o Updated examples since changes in this version result in different hash values.

From -04 to -05:

- o Clarifications about non-apex and multiple ZONEMD RRs.
- o Clarifications about benchmark results.
- o Don't compute ZONEMD on-the-fly.

- o Specification Required for updates to ZONEMD protocol registries.
- o Other rewording based on WGLC feedback.
- o Updated RFC numbers for some references.
- o Use documentation IP addresses instead of loopback.
- o Updated examples in the appendix.

From -05 to -06:

- o Per WG suggestion, no longer include any apex ZONEMD record in digest calculation.
- o Updated examples in the appendix.
- o Clarified verification procedure by describing a loop over all ZONEMD RRs.

From -06 to -07:

- o Added NIC Chile Labs implementation.

From -07 to -08:

- o Update an author's affiliation.
- o Clarified why placeholder RRs are still important (for NSEC/NSEC3).
- o Moved subsection ("Order of RRsets Having the Same Owner Name") with single sentence paragraph up into parent section.

From -08 to -09:

- o Moved format, ordering, inclusion/exclusion into a sub section specific to the SIMPLE scheme.
- o Further clarified rules about multiple ZONEMD RRs (AD comments).
- o Reworded rules about processing of duplicate zone RRs (AD comments).
- o Removed sentence about optional zeroing of digest prior to calculation (AD comments).
- o Other minor changes (AD comments).

From -09 to -10:

- o Add clarification and reference to on-disk modification / corruption of zone files.
- o Added concerns that timing of KSK rollovers could affect validation of ZONEMD record.
- o Addressed SECDIR review and accepted most proposed edits.
- o From SECDIR review, require minimum digest length of 12 octets.
- o From SECDIR review, add SHA512 has hash algorithm 2.
- o From SECDIR review, say that ZONEMD RRs MAY be ignored by local policy.
- o Moved Implementation Status to an appendix with the intention to retain it in RFC.
- o In registry tables, changed Status column to Implementation Requirement.

From -10 to -11:

- o Fixed people's names in the acknowledgments section (blush)
- o Say "has not been modified between origination and retrieval."
- o Say that ZONEMD TTL doesn't matter during verification.
- o Further clarification that the SHA-384 and SHA-512 hashes are not truncated. Future algs might be truncated, but never below 96 bits.

11. References

11.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, [RFC 1034](https://www.rfc-editor.org/info/rfc1034), 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](https://www.rfc-editor.org/info/rfc1035), 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>>.

[11.2](#). Informative References

- [CZDS] Internet Corporation for Assigned Names and Numbers, "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>>.
- [DnsTools] NIC Chile Labs, "DNS tools for zone signature (file, pkcs11-hsm) and validation, and zone digest (ZONEMD)", April 2020, <<https://github.com/niclabs/dns-tools>>.
- [InterNIC] ICANN, "InterNIC FTP site", May 2018, <<ftp://ftp.internic.net/domain/>>.
- [ldns-zone-digest] Verisign, "Implementation of Message Digests for DNS Zones using the ldns library", July 2018, <<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>>.
- [RFC2845] Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", [RFC 2845](#), DOI 10.17487/RFC2845, May 2000, <<https://www.rfc-editor.org/info/rfc2845>>.
- [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>>.
- [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>>.
- [RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", [RFC 5751](#), DOI 10.17487/RFC5751, January 2010, <<https://www.rfc-editor.org/info/rfc5751>>.
- [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>>.
- [RFC7696] Housley, R., "Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms", [BCP 201](#), [RFC 7696](#), DOI 10.17487/RFC7696, November 2015, <<https://www.rfc-editor.org/info/rfc7696>>.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", [RFC 7858](#), DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.

- [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>>.
- [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>>.
- [RootServers]
Root Server Operators, "Root Server Technical Operations", July 2018, <<https://www.root-servers.org/>>.
- [RPZ] Vixie, P. and V. Schryver, "DNS Response Policy Zones (RPZ)", [draft-vixie-dnsop-dns-rpz-00](#) (work in progress), June 2018, <<https://tools.ietf.org/html/draft-vixie-dnsop-dns-rpz-00>>.
- [ZoneDigestHackathon]
Kerr, S., "Prototype implementation of ZONEMD for the IETF 102 hackathon in Python", July 2018, <<https://github.com/shane-kerr/ZoneDigestHackathon>>.

[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.

[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, and an occluded 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 (
                                31cefb03814f5062
                                ad12fa951ba0ef5f
                                8da6ae354a415767
                                246f7dc932ceb1e7
                                42a2108f529db6a3
                                3a11c01493de358d )
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"
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
                                96c5a8f44607bbec )
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 URI.ARPA zone retrieved 2018-10-21. Note this sample zone has (expired) signatures, but no signature for the ZONEMD RR.

```
; <<>> DiG 9.9.4 <<>> @lax.xfr.dns.icann.org uri.arpa axfr
; (2 servers found)
;; global options: +cmd
uri.arpa.      3600     IN      SOA      sns.dns.icann.org. (
```



```

    noc.dns.icann.org. 2018100702 10800 3600 1209600 3600 )
uri.arpa. 3600 IN RRSIG NSEC 8 2 3600 (
    20181028142623 20181007205525 47155 uri.arpa.
    eEC4w/oXLR1Epwgv4MBiDtSBsXhqrJVvJWUpbX8XpetAvD35bxwNCUTi
    /pAJVUXefegWeiriD2rkTgCBCMmn7YQIm3gdR+HjY/+o3BXNQnz97f+e
    HAE9EDDzoNVfL1PyV/2fde9tDeUuAGVvWmD399NGq9jWYMRpyri2kysr q/g= )
uri.arpa. 86400 IN RRSIG NS 8 2 86400 (
    20181028172020 20181007175821 47155 uri.arpa.
    ATyV2A2A8ZoggC+68u4GuP5M0UuR+2rr3eW0kEU55zAHld/7FiBx14ln
    4byJYy7NudUw1M0EXajqFZE7DV18PpcvrP3HeeGaVzKqawj+aus0jbKF
    Bsvs2b1qDZemBfkz/IfAhUTJKnto0vSuiCJkFItu0GjyYNJCz2CqEuGD Wxc= )
uri.arpa. 600 IN RRSIG MX 8 2 600 (
    20181028170556 20181007175821 47155 uri.arpa.
    e7/r3KXDohX1lyVavetFF0bp8fB8aXT76HnN9KCQDxSnSghNM83UQV0t
    lTtD8JVeN1mCvcNFZpagwIgB7XhTtm6Beur/m5ES+4uSnVeS6Q66HBZK
    A3mR95IpevuVIZvvJ+GcCAQpBo6KR0DYvJ/c/ZG6sfYwkZ7qg/Em5/+3 4UI= )
uri.arpa. 3600 IN RRSIG DNSKEY 8 2 3600 (
    20181028152832 20181007175821 15796 uri.arpa.
    nzpbnh00qsgBBP8St28pLvPEQ3wZAUdEBuUwil+rtjjWLYYiqjPxZ286
    XF4Rq1usfV5x71jZz5Iqsw0aQgia91ylodFpLuXD6FTGs2nXGhNKKg1V
    chHgtwj70mXU72GefVgo8TxFYzXuEFP5ZTP92t97FVWVvyF86sbbR
    6DZj3uA2wEvqBVLECGJLrMQ9Yy7MueJl3UA4h4E6z02JY9Yp0W9woq0B
    dqkkwYTwzogYffPmGAJG91RJ2h6cHtFjEZe2MnaY2glqniZ0WT9vXXd
    uFPm0KD9U77Ac+ZtctAF9tsZwSdAoL365E2L1usZbA+K0BnPPqGFJRJk
    5R0A1w== )
uri.arpa. 3600 IN RRSIG DNSKEY 8 2 3600 (
    20181028152832 20181007175821 55480 uri.arpa.
    lwtQV/5szQjKXmbcD47/+r0W8kJPkSRFH1zxxmzt906+DBYyfrH6uq5X
    nHvrUlQ06M12uhqDeL+bDFVgqSpNy+42/OaZvaK3J8EzPZVBHPJykKmv
    63T83aAiJrAyHz0aEdmzLCpalqcEE2ImzlLHSafManRfJL8Yuv+JDZFj
    2WDWfEcUuwkmIZWX11zxp+DxwzyUlRl7x4+ok5iKZWig5UnBAf6B8T75
    WnXzlhCw3F2pXI0a5LYg71L3Tp/xhjN6Yy9jG1IRf5BjB59X2zra3a2R
    PkI09SSnuEwHyF1mDaV5BmQrLGRnCjvwXA7ho2m+vv4SP5dUdXf+GTeA
    1HeBfw== )
uri.arpa. 3600 IN RRSIG SOA 8 2 3600 (
    20181029114753 20181008222815 47155 uri.arpa.
    qn8yBNOHdjGdT79U2Wu9IIahoS0YPOgYP8lG+qwPcrZ1BwGiHywuoUa2
    Mx6BWZlg+HDyaxj2i0mox+IIqoUHHXUb07IUkJFlgr0KCgAR2twDHRXu
    9BUQHy9Sov16wYm3kBTepyXW5FFm8vcdnKAF7sxSY8BbaYNpRIEjDx4A Juc= )
uri.arpa. 3600 IN NSEC ftp.uri.arpa. NS SOA (
    MX RRSIG NSEC DNSKEY )
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. 600 IN MX 10 pechora.icann.org.
uri.arpa. 3600 IN DNSKEY 256 3 8 (

```



```

AwEAAcBi7tSart2J599zbYWspMNGN70IBWb4ziqyQYH9MTB/VCz6WyUK
uXunwiJJbbQ3bcLqTLWEw134B6cTMHrZpjTAb5WAwg4XcWUu8mdcPTiL
Bl6qVRlRD0WiFCTzuYUfkwsh1Rbr7rvrxSQhF5rh71zSpwV5jjjp65Wx
SdJjlH0B )
uri.arpa.          3600      IN          DNSKEY  257 3 8 (
AwEAAbNVv6ulgRd031MtAehz7j3ALRjwZglWesnzvllQl/+hBRZr9QoY
c02I+Dk04Q1NKxox4DUixj8SxP03GwDu0FR9q2/CFi200mZjafbdYtWc
3zSdBbi3q0cwCIx7GuG9eq1L+pg7mdk9dgdNZfHwB0LnqTD8ebLPsr0/
Id7kBaIQY0fMlZnh2fp+2h600JZHtY0DK1U1ssyB5PKsE0tVzo5s6zo9
iXKe5u+8WTMaGDY49vG80JPAKE7ezMiH/NZcUMiE0PRZ8D3foq2dYuS5
ym+vA83Z7v8A+Rwh4UGnjxKB8zmr803V0ASAmHz/gwH5Vb0nH+L0bwFt
l3wpbp+Wpm8= )
uri.arpa.          3600      IN          DNSKEY  257 3 8 (
AwEAAbwnFTakCvaUKsXji4mgmxZUJi1IygbnGahbkmFEa0L16J+TchKR
wczgVfsxUGa2MmeA4hgkAooC3uy+tTmoMsgy8uq/JAaj24DjiHzd46LfD
FK/qMidVqFpYSHeq2Vv5ojkuIsx4oe4KsafGWYN0czKZgh5loGjN2aJG
mrIm++XCphOskgCsQYl65MIzuXffzJyx1Aut+ecAIiVeqRaqQfr8LRU
7wIsLxinXirprtQrbor+Etv1Hp9qXE6ARTZDzf4jvsNpKvLFZtmxzFf3
e/UJz5eHjpwDSiZL7xE8aE1o1nGfPtJx9ZnB3bap1taJ5wY+5XOCKgY0
xmJVvNQlwdE= )
ftp.uri.arpa.      3600      IN          RRSIG   NSEC 8 3 3600 (
20181028080856 20181007175821 47155 uri.arpa.
HCLGAqPxzkYkAT7Q/QNTqEb6YrkP6EP0ef+9Qo5/2zngwAewXEAQiyF9
jD1USJiroM11QqBS3v3aIdW/LX0Rs4Ez3hLcKN01cKhsOuWAqzmE+BPP
Arfh8N95jqh/q6vpaB9UtMkQ53tM2fYU1Gsz0LN0knxbHgDHAh2axMGH lqM= )
ftp.uri.arpa.      604800   IN          RRSIG   NAPTR 8 3 604800 (
20181028103644 20181007205525 47155 uri.arpa.
WoLi+vZzkxaoLr2IGZnwkrVcDf6KxiWQd1WZP/U+AWnV+7MiqsWPZaf0
9toRErerGoF0iOASNxZjBGJrRgjmvOM9U+LZSconP9zrNFd4dIu6kp5
YxlQJ0uHOvx1ZHFCj6lAt1ACUIw04ZhMydTmi27c8MzEOMepvn7iH7r7 k7k= )
ftp.uri.arpa.      3600      IN          NSEC     http.uri.arpa. NAPTR (
RRSIG NSEC )
ftp.uri.arpa.      604800   IN          NAPTR    0 0 "" "" (
"!^ftp://([^:/?#]*).*!$!\\1!i" . )
http.uri.arpa.     3600      IN          RRSIG   NSEC 8 3 3600 (
20181029010647 20181007175821 47155 uri.arpa.
U03NntQ73LHWpfLmUK8nMsqkwVs0Gw2KdsyuHYAjqQSZvKbtmbv7HBmE
H1+Ii3Z+wtfdMZBy5aC/6sHdx69BfZJs16xumycMLAy6325DKTQbIMN+
ift9GrKBC7cgCd2msF/uzSrYxxg4MJQzBPv1kwXnY3b7eJSLIXisBIN7 3b8= )
http.uri.arpa.     604800   IN          RRSIG   NAPTR 8 3 604800 (
20181029011815 20181007205525 47155 uri.arpa.
T7mRrdag+WSmG+n22mtBSQ/0Y3v+rdDnfQV90LN5Fq32N5K2iYFajF7F
Tp56o0znytfcL4fHrq0E0wRc9NW0CCUec9C7Wa1gJQc1lEvgoAM+L6f0
RsEjWq6+9jv1LKMXQv0xQuMX17338uoD/xiAFQSnDbiQKxwWMqVAimv5 7Zs= )
http.uri.arpa.     3600      IN          NSEC     mailto.uri.arpa. NAPTR (
RRSIG NSEC )
http.uri.arpa.     604800   IN          NAPTR    0 0 "" "" (
"!^http://([^:/?#]*).*!$!\\1!i" . )

```



```

mailto.uri.arpa. 3600 IN RRSIG NSEC 8 3 3600 (
    20181028110727 20181007175821 47155 uri.arpa.
    GvxzVL85rEukwGqtuLxek9ipwjBMfTOFIEyJ7afC8HxVMs6mfFa/nEM/
    IdFvvFg+lcYoJSQYuSAVYF13xPbgrxVSLK125QutCFMdC/YjuZEnq5c1
    fQciMRD7R3+znZfm8d8u/snLV9w4D+lTBZrJJUBe1Efc8vum5vvV7819 ZoY= )
mailto.uri.arpa. 604800 IN RRSIG NAPTR 8 3 604800 (
    20181028141825 20181007205525 47155 uri.arpa.
    MaADUgc3fc5v++M0YmqjGk3jBdfIA5RuP62hUSlPsFZ04k37erjIGCfF
    j+g84yc+QgbSde0PQHszl9fE/+SU5ZXiS9YdcbzSZxp2erFpZ0Tchrpg
    916T4vx6i59scodjb0l6bDyZ+mtIPrc1w6b4hUy0UTsDQoAJYxdfEuMg Vy4= )
mailto.uri.arpa. 3600 IN NSEC urn.uri.arpa. NAPTR (
    RRSIG NSEC )
mailto.uri.arpa. 604800 IN NAPTR 0 0 "" "" (
    "!^mailto:(.*)@(.*)$!\2!i" . )
urn.uri.arpa. 3600 IN RRSIG NSEC 8 3 3600 (
    20181028123243 20181007175821 47155 uri.arpa.
    Hgsw4Deops108uWyELGe6hpR/OEqCnTHvahlwiQkHh05CSEQRbhmFAWe
    U0kmGAdTEYrSz+skLRQuITRMwzyFf4oUkZihGyhZyzHbcxWfuDc/Pd/9
    DSl56gdeBwy1evn5wBTms8yWQVknTphbJH395gRqZuaJs3LD/qTyJ5Dp LvA= )
urn.uri.arpa. 604800 IN RRSIG NAPTR 8 3 604800 (
    20181029071816 20181007205525 47155 uri.arpa.
    ALIZD0vBqAQQt40GQ0Efaj80CyE9xSRJRdyvyn/H/wZVXFRFKrQYrLAS
    D/K7q6CMT0xTRCu2J8yes63WJiaJEdnh+dscXzZkm0g4n5PsgZbkvUSW
    BiGtxvz5jNncM0xVbkjbtByrvJQA01cU1mn1DKe1FmVB1uLpVdA9Ib4J hMU= )
urn.uri.arpa. 3600 IN NSEC uri.arpa. NAPTR RRSIG (
    NSEC )
urn.uri.arpa. 604800 IN NAPTR 0 0 "" "" (
    "/urn:([^:]+)/\1/i" . )
uri.arpa. 3600 IN SOA sns.dns.icann.org. (
    noc.dns.icann.org. 2018100702 10800 3600 1209600 3600 )
;; Query time: 66 msec
;; SERVER: 192.0.32.132#53(192.0.32.132)
;; WHEN: Sun Oct 21 20:39:28 UTC 2018
;; XFR size: 34 records (messages 1, bytes 3941)
uri.arpa. 3600 IN ZONEMD 2018100702 1 1 (
    1291b78ddf7669b1a39d014d87626b709b55774c5d7d58fa
    dc556439889a10eaf6f11d615900a4f996bd46279514e473 )

```

[A.5. The ROOT-SERVERS.NET Zone](#)

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

RFC Editor: Please retain this section upon publication.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC 7942](#). The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. 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:

- o Read an input zone and output a zone with the ZONEMD placeholder.
- o Compute zone digest over signed zone and update the ZONEMD record.
- o Re-compute DNSSEC signature over the ZONEMD record.
- o Verify the zone digest from an input zone.

This implementation does not:

- o 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 [[ZoneDigestHackathon](#)]. This implementation is in Python and is able to perform the following functions:

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

This implementation does not:

- o Re-compute DNSSEC signature over the ZONEMD record.
- o Perform DNSSEC validation of the ZONEMD record.

B.3. NIC Chile Labs Implementation

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

- o Compute zone digest over signed zone and update the ZONEMD record.
- o Verify the zone digest from an input zone.
- o Perform DNSSEC validation of the ZONEMD record during verification.
- o Re-compute DNSSEC signature over the ZONEMD record.

Authors' Addresses

Duane Wessels
Verisign
12061 Bluemont Way
Reston, VA 20190

Phone: +1 703 948-3200
Email: dwessels@verisign.com
URI: <https://verisign.com>

Piet Barber
Verisign
12061 Bluemont Way
Reston, VA 20190

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

Email: warren@kumari.net

Wes Hardaker
USC/ISI
P.O. Box 382
Davis, CA 95617

Email: ietf@hardakers.net

