Message Digests for DNS Zones
Including Plans for the Root Zone
ICANN DNS Symposium
May 25-27, 2021
What is a DNS Zone Digest?

- A cryptographic digest, or hash, of the data in a DNS zone
- Embedded in the zone data itself
- Computed by zone publishers
- Verified by zone recipients
Analogous to Checksum Files for Software

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Last Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>File: FOOTER.html</td>
<td>1 KB</td>
<td>4/18/21 1:21:00 AM PDT</td>
</tr>
<tr>
<td>File: HEADER.html</td>
<td>4 KB</td>
<td>4/18/21 1:21:00 AM PDT</td>
</tr>
<tr>
<td>File: SHA256SUMS</td>
<td>1 KB</td>
<td>4/18/21 1:21:00 AM PDT</td>
</tr>
<tr>
<td>File: SHA256SUMS.gpg</td>
<td>1 KB</td>
<td>4/18/21 1:21:00 AM PDT</td>
</tr>
<tr>
<td>File: hirsute-desktop-amd64.iso</td>
<td>2752696 KB</td>
<td>4/18/21 1:19:00 AM PDT</td>
</tr>
<tr>
<td>File: hirsute-desktop-amd64.iso.zsync</td>
<td>5377 KB</td>
<td>4/18/21 1:20:00 AM PDT</td>
</tr>
<tr>
<td>File: hirsute-desktop-amd64.list</td>
<td>16 KB</td>
<td>4/18/21 1:19:00 AM PDT</td>
</tr>
<tr>
<td>File: hirsute-desktop-amd64.manifest</td>
<td>56 KB</td>
<td>4/18/21 1:02:00 AM PDT</td>
</tr>
<tr>
<td>File: hirsute-desktop-arm64.iso</td>
<td>2226348 KB</td>
<td>4/18/21 1:20:00 AM PDT</td>
</tr>
<tr>
<td>File: hirsute-desktop-arm64.iso.zsync</td>
<td>4349 KB</td>
<td>4/18/21 1:21:00 AM PDT</td>
</tr>
<tr>
<td>File: hirsute-desktop-arm64.list</td>
<td>1 KB</td>
<td>4/18/21 1:20:00 AM PDT</td>
</tr>
<tr>
<td>File: hirsute-desktop-arm64.manifest</td>
<td>56 KB</td>
<td>4/18/21 1:16:00 AM PDT</td>
</tr>
</tbody>
</table>
How Does it Work?

- Specified in RFC 8976
- Zone data is given as input to a digest function
  - Using a well-defined and consistent ordering
  - And in a well-defined and consistent format
  - Excluding the ZONEMD record itself (and its signatures)
- Digest is included in the zone itself, and (ideally) signed with DNSSEC
Why is this Useful?

• Protects zone data “at rest”
  • e.g., data security vs channel security
• Useful in distributing zone data between primary and secondary name servers, especially in modern, complex environments
• Increased interest in serving root zone data locally (e.g., RFC 8806)
• CZDS – Centralized Zone Data Service
• RPZ – Response Policy Zones
Additional Technical Details

example. 86400 IN ZONEMD 2018031900 1 1
8ee54f64ce0d57fd70e1a4811a9ca9e849e2e50cb598edf3ba9c2a58625335c1f966835f0d4338d9f78f557227d63bf6

• Serial field
• Must match SOA record serial
Additional Technical Details

example. 86400 IN ZONEMD 2018031900 1 1
8ee54f64ce0d57fd70e1a4811a9ca9e849e2e50cb598edf3ba9c2a58
625335c1f966835f0d4338d9f78f557227d63bf6

• Scheme field

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Mnemonic</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Simple ZONEMD collation</td>
<td>SIMPLE</td>
</tr>
<tr>
<td>240-254</td>
<td>Private Use</td>
<td></td>
</tr>
<tr>
<td>255</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>
Additional Technical Details

example. 86400 IN ZONEMD 2018031900 1 1 8ee54f64ce0d57fd70e1a4811a9ca9e849e2e50cb598edf3ba9c2a58625335c1f966835f0d4338d9f78f557227d63bf6

• Hash Algorithm field

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Mnemonic</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SHA-384</td>
<td>SHA384</td>
</tr>
<tr>
<td>2</td>
<td>SHA-512</td>
<td>SHA512</td>
</tr>
<tr>
<td>240-254</td>
<td>Private Use</td>
<td></td>
</tr>
<tr>
<td>255</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>
Additional Technical Details

example. 86400 IN ZONEMD 2018031900 1 1
8ee54f64ce0d57fd70e1a4811a9ca9e849e2e50cb598edf3ba9c2a58
625335c1f966835f0d4338d9f78f557227d63bf6

- Digest field
- Length depends on chosen Hash Algorithm
  - Always 48 octets for SHA-384
  - Always 64 octets for SHA-512
  - Never less than 12 octets for any hash algorithm, including private use
# Implementations

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Publish</th>
<th>Verify</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ldns-zone-digest</td>
<td>yes</td>
<td>yes</td>
<td>RFC reference implementation</td>
</tr>
<tr>
<td>Unbound</td>
<td>no</td>
<td>yes</td>
<td><code>auth-zone</code> stanza</td>
</tr>
<tr>
<td>ldns</td>
<td>yes</td>
<td>yes</td>
<td><code>ldns-signzone</code> and <code>ldns-verifyzone</code></td>
</tr>
<tr>
<td>dns-tools from NIC Chile Labs</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>PowerDNS Resolver</td>
<td></td>
<td></td>
<td>work in progress</td>
</tr>
<tr>
<td>Knot Resolver</td>
<td></td>
<td></td>
<td>work in progress</td>
</tr>
<tr>
<td>BIND9</td>
<td></td>
<td></td>
<td>parse only</td>
</tr>
<tr>
<td>Perl Net::DNS</td>
<td></td>
<td></td>
<td>parse only</td>
</tr>
</tbody>
</table>
Benchmarks with ldns-zone-digest

Time to Calculate Zone Digest for TLD/Root Zones (SHA384)
Example Using Unbound (unreleased version)

server:
  verbosity: 3
  interface: 127.0.0.1

auth-zone:
  name: "example"
  zonefile: "example.zone"

[1619565823] unbound[73900:0] debug: module config: "validator iterator"
[1619565823] unbound[73900:0] notice: init module 0: validator
[1619565823] unbound[73900:0] debug: validator nsec3cfg keysz 1024 mxiter 150
[1619565823] unbound[73900:0] debug: validator nsec3cfg keysz 2048 mxiter 500
[1619565823] unbound[73900:0] debug: validator nsec3cfg keysz 4096 mxiter 2500
[1619565823] unbound[73900:0] notice: init module 1: iterator
[1619565823] unbound[73900:0] debug: target fetch policy for level 0 is 3
[1619565823] unbound[73900:0] debug: target fetch policy for level 1 is 2
[1619565823] unbound[73900:0] debug: target fetch policy for level 2 is 1
[1619565823] unbound[73900:0] debug: target fetch policy for level 3 is 0
[1619565823] unbound[73900:0] debug: target fetch policy for level 4 is 0
[1619565823] unbound[73900:0] debug: donotq: 127.0.0.0/8
[1619565823] unbound[73900:0] debug: donotq: ::1
[1619565823] unbound[73900:0] debug: read zonefile example.zone for example.
[1619565823] unbound[73900:0] debug: auth-zone example. ZONEMD hash is correct
[1619565823] unbound[73900:0] debug: auth zone example. ZONEMD verification successful
ZONEMD for the Root Zone
RZERC Recommendations

Earlier this year, ICANN’s Root Zone Evolution Review Committee (RZERC) made the following recommendations to the ICANN Board regarding ZONEMD in the root zone:

1. The root zone maintainer and root server operators should verify and confirm that the addition of a ZONEMD resource record will in no way negatively impact the distribution of root zone data within the RSS.

2. The DNS and Internet community should be made aware of plans to use ZONEMD in the root zone, and be given an opportunity to offer feedback.

3. Developers of name server software are encouraged to implement ZONEMD and consider enabling it by default when the software is configured to locally serve root zone data.

4. Public Technical Identifiers (PTI) and the RZM should jointly develop a plan for deploying ZONEMD in the root zone, and make this plan available for review by RZERC.

Source: https://www.icann.org/iana_rzerc_docs/449-rzerc003-adding-zone-data-protections-to-the-root-zone-v-final
Tentative Root ZONEMD Deployment Plans

• A single ZONEMD record
• Initially with a private-use algorithm?
  • Everyone remembers the “DURZ”?
• Then with SHA-512
• Possibly starting as early as December 2021
• ZONEMD RR format or generic / unknown RR format?