| | |
|---|---|
| Stream: | Independent Submission |
| RFC: | 9102 |
| Category: | Experimental |
| Published: | July 2021 |
| ISSN: | 2070-1721 |


| Authors: | V. Dukhovni | S. Huque | W. Toorop | P. Wouters | M. Shore |
|---|---|---|---|---|---|
| | *Two Sigma* | *Salesforce* | *NLnet Labs* | *Aiven* | *Fastly* |


# RFC 9102
# TLS DNSSEC Chain Extension

## Abstract


This document describes an experimental TLS extension for the in-band transport of the complete set of records that can be validated by DNSSEC and that are needed to perform DNS-Based Authentication of Named Entities (DANE) of a TLS server. This extension obviates the need to perform separate, out-of-band DNS lookups. When the requisite DNS records do not exist, the extension conveys a denial-of-existence proof that can be validated.

This experimental extension is developed outside the IETF and is published here to guide implementation of the extension and to ensure interoperability among implementations.


## Status of This Memo



## Copyright Notice

## Table of Contents

## 1. Introduction

This document describes an experimental TLS [RFC5246] [RFC8446] extension for in-band transport of the complete set of resource records (RRs) validated by DNSSEC [RFC4033] [RFC4034] [RFC4035]. This extension enables a TLS client to perform DANE authentication [RFC6698] [RFC7671] of a TLS server without the need to perform out-of-band DNS lookups. Retrieval of the required DNS records may be unavailable to the client [DISCOVERY] or may incur undesirable additional latency.

The extension described here allows a TLS client to request that the TLS server return the DNSSEC authentication chain corresponding to its DNSSEC-validated DANE TLSA resource record set (RRset) or authenticated denial of existence of such an RRset (as described in Section 2.3.1). If the server supports this extension, it performs the appropriate DNS queries, builds the authentication chain, and returns it to the client. The server will typically use a previously cached authentication chain, but it will need to rebuild it periodically as described in Section 4. The client then authenticates the chain using a preconfigured DNSSEC trust anchor.

In the absence of TLSA records, this extension conveys the required authenticated denial of existence. Such proofs are needed to securely signal that specified TLSA records are not available so that TLS clients can safely fall back to authentication based on Public Key Infrastructure X.509 (PKIX, sometimes called WebPKI) if allowed by local policy. These proofs are also needed to avoid downgrade from opportunistic authenticated TLS (when DANE TLSA records are present) to unauthenticated opportunistic TLS (in the absence of DANE). Denial-of-existence records are also used by the TLS client to clear extension pins that are no longer relevant, as described in Section 7.

This extension supports DANE authentication of either X.509 certificates or raw public keys, as described in the DANE specification [RFC6698] [RFC7671] [RFC7250].

This extension also mitigates against an unknown key share (UKS) attack [DANE-UKS] when using raw public keys since the server commits to its DNS name (normally found in its certificate) via the content of the returned TLSA RRset.

This experimental extension is developed outside the IETF and is published here to guide implementation of the extension and to ensure interoperability among implementations.

### 1.1. Scope of the Experiment

The mechanism described in this document is intended to be used with applications on the wider internet. One application of TLS well suited for the TLS DNSSEC Chain extension is DNS over TLS [RFC7858]. In fact, one of the authentication methods for DNS over TLS *is* the mechanism described in this document, as specified in Section 8.2.2 of [RFC8310].

The need for this mechanism when using DANE to authenticate a DNS-over-TLS resolver is obvious, since DNS may not be available to perform the required DNS lookups. Other applications of TLS would benefit from using this mechanism as well. The client sides of those applications would not be required to be used on endpoints with a working DNSSEC resolver in order for them to use the DANE authentication of the TLS service. Therefore, we invite other TLS services to try out this mechanism as well.

In the TLS Working Group, concerns have been raised that the pinning technique as described in Section 7 would complicate deployability of the TLS DNSSEC chain extension. The goal of the experiment is to study these complications in real-world deployments. This experiment hopefully will give the TLS Working Group some insight into whether or not this is a problem.

If the experiment is successful, it is expected that the findings of the experiment will result in an updated document for Standards Track approval.

### 1.2. Requirements Notation

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.

## 2. DNSSEC Authentication Chain Extension

### 2.1. Protocol, TLS 1.2

A client **MAY** include an extension of type `dnssec_chain` in the (extended) ClientHello. The `extension_data` field of this extension consists of the server's 16-bit TCP port number in network (big-endian) byte order. Clients sending this extension **MUST** also send the Server Name Identification (SNI) extension [RFC6066]. Together, these make it possible for the TLS server to determine which authenticated TLSA RRset chain needs to be used for the `dnssec_chain` extension.

When a server that implements (and is configured to enable the use of) this extension receives a `dnssec_chain` extension in the ClientHello, it **MUST** first check whether the requested TLSA RRset (based on the port number in this extension and hostname in the SNI extension) is associated with the server. If the extension, the SNI hostname, or the port number is unsupported, the server's extended ServerHello message **MUST NOT** include the `dnssec_chain` extension.

Otherwise, the server's extended ServerHello message **MUST** contain a serialized authentication chain using the format described below. If the server does not have access to the requested DNS chain -- for example, due to a misconfiguration or expired chain -- the server **MUST** omit the extension rather than send an incomplete chain. Clients that are expecting this extension **MUST** interpret this as a downgrade attack and **MUST** abort the TLS connection. Therefore, servers **MUST** send denial-of-existence proofs unless, for the particular application protocol or service, clients are expected to continue even in the absence of such a proof. As with all TLS extensions, if the server does not support this extension, it will not return any authentication chain.

The set of supported combinations of a port number and SNI name may be configured explicitly by server administrators or could be inferred from the available certificates combined with a list of supported ports. It is important to note that the client's notional port number may be different from the actual port on which the server is receiving connections.

Differences between the client's notional port number and the actual port at the server could be a result of intermediate systems performing network address translation or a result of a redirect via HTTPS or SVCB records (both defined in [DNSOP-SVCB-HTTPS]).

Though a DNS zone's HTTPS or SVCB records may be signed, a client using this protocol might not have direct access to a validating resolver and might not be able to check the authenticity of the target port number or hostname. In order to avoid downgrade attacks via forged DNS records, the SNI name and port number inside the client extension **MUST** be based on the original SNI name and port and **MUST NOT** be taken from the encountered HTTPS or SVCB record. The client supporting this document and HTTPS or SVCB records **MUST** still use the HTTPS or SVCB records to select the target transport endpoint. Servers supporting this extension that are targets of HTTPS or SVCB records **MUST** be provisioned to process client extensions based on the client's logical service endpoint's SNI name and port as it is prior to HTTPS or SVCB indirection.

## 2.2. Protocol, TLS 1.3

In TLS 1.3 [RFC8446], when the server receives the `dnssec_chain` extension, it adds its `dnssec_chain` extension to the extension block of the Certificate message containing the end-entity certificate being validated rather than to the extended ServerHello message.

The extension protocol behavior otherwise follows that specified for TLS version 1.2 [RFC5246].

### 2.3. DNSSEC Authentication Chain Data

The `extension_data` field of the client's `dnssec_chain` extension **MUST** contain the server's 16-bit TCP port number in network (big-endian) byte order:

```
struct {
    uint16 PortNumber;
} DnssecChainExtension;
```

The `extension_data` field of the server's `dnssec_chain` extension **MUST** contain a DNSSEC authentication chain encoded in the following form:

```
struct {
    uint16 ExtSupportLifetime;
    opaque AuthenticationChain<1..2^16-1>
} DnssecChainExtension;
```

The ExtSupportLifetime value is the number of hours that the TLS server has committed itself to serving this extension. A value of zero prohibits the client from unilaterally requiring ongoing use of the extension based on prior observation of its use (extension pinning). This is further described in Section 7.

The AuthenticationChain is composed of a sequence of uncompressed wire format DNS RRs (including all requisite RRSIG RRs [RFC4034]) in no particular order. The format of the resource record is described in [RFC1035], Section 3.2.1.

```
RR = { owner, type, class, TTL, RDATA length, RDATA }
```

The order of returned RRs is unspecified, and a TLS client **MUST NOT** assume any ordering of RRs.

Use of DNS wire format records enables easier generation of the data structure on the server and easier verification of the data on the client by means of existing DNS library functions.

The returned RRsets **MUST** contain either the TLSA RRset or the associated denial-of-existence proof of the configured (and requested) SNI name and port. In either case, the chain of RRsets **MUST** be accompanied by the full set of DNS records needed to authenticate the TLSA record set or its denial of existence up the DNS hierarchy to either the root zone or another trust anchor mutually configured by the TLS server and client.

When some subtree in the chain is subject to redirection via DNAME records, the associated inferred CNAME records need not be included. They can be inferred by the DNS validation code in the client. Any applicable ordinary CNAME records that are not synthesized from DNAME records **MUST** be included along with their RRSIGs.

In case of a server-side DNS problem, servers may be unable to construct the authentication chain and would then have no choice but to omit the extension.

In the case of a denial-of-existence response, the authentication chain **MUST** include all DNSSEC-signed records, starting with those from the trust anchor zone, that chain together to reach a proof of either:

- the nonexistence of the TLSA records (possibly redirected via aliases) or
- an insecure delegation above or at the (possibly redirected) owner name of the requested TLSA RRset.

Names that are aliased via CNAME and/or DNAME records may involve multiple branches of the DNS tree. In this case, the authentication chain structure needs to include DS and DNSKEY record sets that cover all the necessary branches.

The returned chain **SHOULD** also include the DNSKEY RRsets of all relevant trust anchors (typically just the root DNS zone). Though the same trust anchors are presumably also preconfigured in the TLS client, including them in the response from the server permits TLS clients to use the automated trust anchor rollover mechanism defined in [RFC5011] to update their configured trust anchors.

Barring prior knowledge of particular trust anchors that the server shares with its clients, the chain constructed by the server **MUST** be extended as closely as possible to the root zone. Truncation of the chain at some intermediate trust anchor is generally only appropriate inside private networks where all clients and the server are expected to be configured with DNS trust anchors for one or more non-root domains.

The following is an example of the records in the AuthenticationChain structure for the HTTPS server at `www.example.com`, where there are zone cuts at `com` and `example.com` (record data are omitted here for brevity):

```
_443._tcp.www.example.com. TLSA
RRSIG(_443._tcp.www.example.com. TLSA)
example.com. DNSKEY
RRSIG(example.com. DNSKEY)
example.com. DS
RRSIG(example.com. DS)
com. DNSKEY
RRSIG(com. DNSKEY)
com. DS
RRSIG(com. DS)
. DNSKEY
RRSIG(. DNSKEY)
```

The following is an example of denial of existence for a TLSA RRset at `_443._tcp.www.example.com`. The NSEC record in this example asserts the nonexistence of both the requested RRset and any potentially relevant wildcard records.

```
www.example.com. IN NSEC example.com. A NSEC RRSIG
RRSIG(www.example.com. NSEC)
example.com. DNSKEY
RRSIG(example.com. DNSKEY)
example.com. DS
RRSIG(example.com. DS)
com. DNSKEY
RRSIG(com. DNSKEY)
com. DS
RRSIG(com. DS)
. DNSKEY
RRSIG(. DNSKEY)
```

The following is an example of (hypothetical) insecure delegation of `example.com` from the `.com` zone. This example shows NSEC3 records [RFC5155] with opt-out.

```
; covers example.com
onib9mgub9h0rml3cdf5bgrj59dkjhvj.com. NSEC3 (1 1 0 -
  onib9mgub9h0rml3cdf5bgrj59dkjhvl NS DS RRSIG)
RRSIG(onib9mgub9h0rml3cdf5bgrj59dkjhvj.com. NSEC3)
; covers *.com
3rl2r262eg0n1ap5olhae7mah2ah09hi.com. NSEC3 (1 1 0 -
  3rl2r262eg0n1ap5olhae7mah2ah09hk NS DS RRSIG)
RRSIG(3rl2r262eg0n1ap5olhae7mah2ah09hj.com. NSEC3)
; closest-encloser "com"
ck0pojmg874ljref7efn8430qvit8bsm.com. NSEC3 (1 1 0 -
  ck0pojmg874ljref7efn8430qvit8bsm.com
  NS SOA RRSIG DNSKEY NSEC3PARAM)
RRSIG(ck0pojmg874ljref7efn8430qvit8bsm.com. NSEC3)
com. DNSKEY
RRSIG(com. DNSKEY)
com. DS
RRSIG(com. DS)
. DNSKEY
RRSIG(. DNSKEY)
```

### 2.3.1. Authenticated Denial of Existence

TLS servers that support this extension and respond to a request containing this extension that do not have a signed TLSA record for the configured (and requested) SNI name and port **MUST** instead return a DNSSEC chain that provides authenticated denial of existence for the configured SNI name and port. A TLS client receiving proof of authenticated denial of existence **MUST** use an alternative method to verify the TLS server identity or close the connection. Such an alternative could be the classic PKIX model of preinstalled root certificate authorities (CAs).

Authenticated denial chains include NSEC or NSEC3 records that demonstrate one of the following facts:

- The TLSA record (after any DNSSEC-validated alias redirection) does not exist.
- There is no signed delegation to a DNS zone that is either an ancestor of or the same as the TLSA record name (after any DNSSEC-validated alias redirection).

## 3. Construction of Serialized Authentication Chains

This section describes a possible procedure for the server to use to build the serialized DNSSEC chain.

When the goal is to perform DANE authentication [RFC6698] [RFC7671] of the server, the DNS record set to be serialized is a TLSA record set corresponding to the server's domain name, protocol, and port number.

The domain name of the server **MUST** be that included in the TLS `server_name` (SNI) extension [RFC6066]. If the server does not recognize the SNI name as one of its own names but wishes to proceed with the handshake rather than abort the connection, the server **MUST NOT** send a `dnssec_chain` extension to the client.

The name in the client's SNI extension **MUST NOT** be CNAME expanded by the server. The TLSA base domain (Section 3 of [RFC6698]) **SHALL** be the hostname from the client's SNI extension, and the guidance in Section 7 of [RFC7671] does not apply. See Section 9 for further discussion.

The TLSA record to be queried is constructed by prepending underscore-prefixed port number and transport name labels to the domain name as described in [RFC6698]. The port number is taken from the client's `dnssec_chain` extension. The transport name is "tcp" for TLS servers and "udp" for DTLS servers. The port number label is the leftmost label, followed by the transport name label, followed by the server domain name (from SNI).

The components of the authentication chain are typically built by starting at the target record set and its corresponding RRSIG, then traversing the DNS tree upwards towards the trust anchor zone (normally the DNS root). For each zone cut, the DNSKEY, DS RRsets, and their signatures are added. However, see Section 2.3 for specific processing needed for aliases. If DNS response messages contain any domain names utilizing name compression [RFC1035], then they **MUST** be uncompressed prior to inclusion in the chain.

Implementations of EDNS CHAIN query requests as specified in [RFC7901] may offer an easier way to obtain all of the chain data in one transaction with an upstream DNSSEC-aware recursive server.

## 4. Caching and Regeneration of the Authentication Chain

DNS records have Time To Live (TTL) parameters, and DNSSEC signatures have validity periods (specifically signature expiration times). After the TLS server constructs the serialized authentication chain, it **SHOULD** cache and reuse it in multiple TLS connection handshakes.

However, it **SHOULD** refresh and rebuild the chain as TTL values require. A server implementation could carefully track TTL parameters and requery component records in the chain correspondingly. Alternatively, it could be configured to rebuild the entire chain at some predefined periodic interval that does not exceed the DNS TTLs of the component records in the chain. If a record in the chain has a very short TTL (e.g., 0 up to a few seconds), the server **MAY** decide to serve the authentication chain a few seconds past the minimum TTL in the chain. This allows an implementation to dedicate a process or single thread to building the authentication chain and reuse it for more than a single waiting TLS client before needing to rebuild the authentication chain.

## 5. Expired Signatures in the Authentication Chain

A server **MAY** look at the signature expiration of RRSIG records. While these should never expire before the TTL of the corresponding DNS record is reached, if this situation is nevertheless encountered, the server **MAY** lower the TTL to prevent serving expired RRSIGs if possible. If the signatures are already expired, the server **MUST** still include these records in the authentication chain. This allows the TLS client to either support a grace period for staleness or give a detailed error, either as a log message or a message to a potential interactive user of the TLS connection. The TLS client **SHOULD** handle expired RRSIGs similarly to how it handles expired PKIX certificates.

## 6. Verification

A TLS client performing DANE-based verification might not need to use this extension. For example, the TLS client could perform DNS lookups and DANE verification without this extension, or it could fetch authentication chains via another protocol. If the TLS client already possesses a valid TLSA record, it **MAY** bypass use of this extension. However, if it includes this extension, it **MUST** use the TLS server reply to update the extension pinning status of the TLS server's extension lifetime. See Section 7.

A TLS client making use of this specification that receives a valid DNSSEC authentication chain extension from a TLS server **MUST** use this information to perform DANE authentication of the TLS server. In order to perform the validation, it uses the mechanism specified by the DNSSEC protocol [RFC4035] [RFC5155]. This mechanism is sometimes implemented in a DNSSEC validation engine or library.

If the authentication chain validates, the TLS client then performs DANE authentication of the server according to the DANE TLS protocol [RFC6698] [RFC7671].

Clients **MAY** cache the server's validated TLSA RRset to amortize the cost of receiving and validating the chain over multiple connections. The period of such caching **MUST NOT** exceed the TTL associated with those records. A client that possesses a validated and unexpired TLSA RRset or the full chain in its cache does not need to send the `dnssec_chain` extension for subsequent connections to the same TLS server. It can use the cached information to perform DANE authentication.

Note that when a client and server perform TLS session resumption, the server sends no `dnssec_chain`. This is particularly clear with TLS 1.3, where the certificate message to which the chain might be attached is also not sent on resumption.

# 7. Extension Pinning

TLS applications can be designed to unconditionally mandate this extension. Such TLS clients requesting this extension would abort a connection to a TLS server that does not respond with an extension reply that can be validated.

However, in a mixed-use deployment of PKIX and DANE, there is the possibility that the security of a TLS client is downgraded from DANE to PKIX. This can happen when a TLS client connection is intercepted and redirected to a rogue TLS server presenting a TLS certificate that is considered valid from a PKIX point of view but does not match the legitimate server's TLSA records. By omitting this extension, such a rogue TLS server could downgrade the TLS client to validate the mis-issued certificate using only PKIX and not via DANE, provided the TLS client is also not able to fetch the TLSA records directly from DNS.

The ExtSupportLifetime element of the extension provides a countermeasure against such downgrade attacks. Its value represents the number of hours that the TLS server (or cluster of servers serving the same server name) commits to serving this extension in the future. This is referred to as the "pinning time" or "extension pin" of the extension. A non-zero extension pin value received **MUST** ONLY be used if the extension also contains a valid TLSA authentication chain that matches the server's certificate chain (the server passes DANE authentication based on the enclosed TLSA RRset).

Any existing extension pin for the server instance (name and port) **MUST** be cleared on receipt of a valid denial of existence for the associated TLSA RRset. The same also applies if the client obtained the denial-of-existence proof via another method, such as through direct DNS queries. Based on the TLS client's local policy, it **MAY** then terminate the connection or **MAY** continue using PKIX-based server authentication.

Extension pins **MUST** also be cleared upon the completion of a DANE-authenticated handshake with a server that returns a `dnssec_chain` extension with a zero ExtSupportLifetime.

Upon completion of a fully validated handshake with a server that returns a `dnssec_chain` extension with a non-zero ExtSupport lifetime, the client **MUST** update any existing pin lifetime for the service (name and port) to a value that is not longer than that indicated by the server. The client **MAY**, subject to local policy, create a previously nonexistent pin, again for a lifetime that is not longer than that indicated by the server.

The extension support lifetime is not constrained by any DNS TTLs or RRSIG expirations in the returned chain. The extension support lifetime is the time for which the TLS server is committing itself to serve the extension; it is not a validity time for the returned chain data. During this period, the DNSSEC chain may be updated. Therefore, the ExtSupportLifetime value is not constrained by any DNS TTLs or RRSIG expirations in the returned chain.

Clients **MAY** implement support for a subset of DANE certificate usages. For example, clients may support only DANE-EE(3) and DANE-TA(2) [RFC7218], only PKIX-EE(1) and PKIX-TA(0), or all four. Clients that implement DANE-EE(3) and DANE-TA(2) **MUST** implement the relevant updates in [RFC7671].

For a non-zero saved value ("pin") of the ExtSupportLifetime element of the extension, TLS clients that do not have a cached TLSA RRset with an unexpired TTL **MUST** use the extension for the associated name and port to obtain this information from the TLS server. This TLS client then **MUST** require that the TLS server respond with this extension, which **MUST** contain a valid TLSA RRset or proof of nonexistence of the TLSA RRset that covers the requested name and port. Note that a nonexistence proof or proof of insecure delegation will clear the pin. The TLS client **MUST** require this for as long as the time period specified by the pin value, independent of the DNS TTLs. During this process, if the TLS client fails to receive this information, it **MUST** either abort the connection or delay communication with the server via the TLS connection until it is able to obtain valid TLSA records (or proof of nonexistence) out of band, such as via direct DNS lookups. If attempts to obtain the TLSA RRset out of band fail, the client **MUST** abort the TLS connection. It **MAY** try a new TLS connection again (for example, using an exponential back-off timer) in an attempt to reach a different TLS server instance that does properly serve the extension.

A TLS client that has a cached validated TLSA RRset and a valid non-zero extension pin time **MAY** still refrain from requesting the extension as long as it uses the cached TLSA RRset to authenticate the TLS server. This RRset **MUST NOT** be used beyond its received TTL. Once the TLSA RRset's TTL has expired, the TLS client with a valid non-zero extension pin time **MUST** request the extension and **MUST** abort the TLS connection if the server responds without the extension. A TLS client **MAY** attempt to obtain the valid TLSA RRset by some other means before initiating a new TLS connection.

Note that requiring the extension is NOT the same as requiring the use of DANE TLSA records or even DNSSEC. A DNS zone operator may at any time delete the TLSA records or even remove the DS records to disable the secure delegation of the server's DNS zone. The TLS server will replace the chain with the corresponding denial-of-existence chain when it updates its cached TLSA authentication chain. The server's only obligation is continued support for this extension.

## 8. Trust Anchor Maintenance

The trust anchor may change periodically, e.g., when the operator of the trust anchor zone performs a DNSSEC key rollover. TLS clients using this specification **MUST** implement a mechanism to keep their trust anchors up to date. They could use the method defined in [RFC5011] to perform trust anchor updates in-band in TLS by tracking the introduction of new keys seen in the trust anchor DNSKEY RRset. However, alternative mechanisms external to TLS may also be utilized. Some operating systems may have a system-wide service to maintain and keep the root trust anchor up to date. In such cases, the TLS client application could simply reference that as its trust anchor, periodically checking whether it has changed. Some applications may prefer to implement trust anchor updates as part of their automated software updates.

## 9. Virtual Hosting

Delivery of application services is often provided by a third party on behalf of the domain owner (hosting customer). Since the domain owner may want to be able to move the service between providers, non-zero support lifetimes for this extension should only be enabled by mutual agreement between the provider and domain owner.

When CNAME records are employed to redirect network connections to the provider's network, as mentioned in Section 3, the server uses the client's SNI hostname as the TLSA base domain without CNAME expansion. When the certificate chain for the service is managed by the provider, it is impractical to coordinate certificate changes by the provider with updates in the hosting customer's DNS. Therefore, the TLSA RRset for the hosted domain is best configured as a CNAME from the customer's domain to a TLSA RRset that is managed by the provider as part of delivering the hosted service. For example:

```
; Customer DNS
www.example.com. IN CNAME node1.provider.example.
_443._tcp.www.example.com. IN CNAME _dane443.node1.provider.example.
; Provider DNS
node1.provider.example. IN A 192.0.2.1
_dane443.node1.provider.example. IN TLSA 1 1 1 ...
```

Clients that obtain TLSA records directly from DNS, bypassing this extension, may perform CNAME expansion as in Section 7 of [RFC7671]. If TLSA records are associated with the fully expanded name, that name may be used as the TLSA base domain and SNI name for the TLS handshake.

To avoid confusion, it is **RECOMMENDED** that server operators not publish TLSA RRs (_port._tcp. + base domain) based on the expanded CNAMEs used to locate their network addresses. Instead, the server operator **SHOULD** publish TLSA RRs at an alternative DNS node (as in the example above), to which the hosting customer will publish a CNAME alias. This results in all clients (whether they obtain TLSA records from DNS directly or employ this extension) seeing the same TLSA records and sending the same SNI name.

## 10. Operational Considerations

When DANE is being introduced incrementally into an existing PKIX environment, there may be scenarios in which DANE authentication for a server fails but PKIX succeeds, or vice versa. What happens here depends on TLS client policy. If DANE authentication fails, the client may decide to fall back to regular PKIX authentication. In order to do so efficiently within the same TLS handshake, the TLS server needs to have provided the full X.509 certificate chain. When TLS servers only support DANE-EE or DANE-TA modes, they have the option to send a much smaller certificate chain: just the EE certificate for the former and a short certificate chain from the

DANE trust anchor to the EE certificate for the latter. If the TLS server supports both DANE and regular PKIX and wants to allow efficient PKIX fallback within the same handshake, they should always provide the full X.509 certificate chain.

When a TLS server operator wishes to no longer deploy this extension, it must properly decommission its use. If a non-zero pin lifetime is presently advertised, it must first be changed to 0. The extension can be disabled once all previously advertised pin lifetimes have expired. Removal of TLSA records or even DNSSEC signing of the zone can be done at any time, but the server **MUST** still be able to return the associated denial-of-existence proofs to any clients that have unexpired pins.

TLS clients **MAY** reduce the received extension pin value to a maximum set by local policy. This can mitigate a theoretical yet unlikely attack where a compromised TLS server is modified to advertise a pin value set to the maximum of 7 years. Care should be taken not to set a local maximum that is too short as that would reduce the downgrade attack protection that the extension pin offers.

If the hosting provider intends to use end-entity TLSA records (certificate usage PKIX-EE(1) or DANE-EE(3)), then the simplest approach is to use the same key pair for all the certificates at a given hosting node and publish "1 1 1" or "3 1 1" RRs matching the common public key. Since key rollover cannot be simultaneous across multiple certificate updates, there will be times when multiple "1 1 1" (or "3 1 1") records will be required to match all the extant certificates. Multiple TLSA records are, in any case, needed a few TTLs before certificate updates as explained in Section 8 of [RFC7671].

If the hosting provider intends to use trust anchor TLSA records (certificate usage PKIX-TA(0) or DANE-TA(2)), then the same TLSA record can match all end-entity certificates issues by the certification authority in question and continues to work across end-entity certificate updates so long as the issuer certificate or public keys remain unchanged. This can be easier to implement at the cost of greater reliance on the security of the selected certification authority.

The provider can, of course, publish separate TLSA records for each customer, which increases the number of such RRsets that need to be managed but makes each one independent of the rest.

## 11. Security Considerations

The security considerations of the normatively referenced RFCs all pertain to this extension. Since the server is delivering a chain of DNS records and signatures to the client, it **MUST** rebuild the chain in accordance with TTL and signature expiration of the chain components as described in Section 4. TLS clients need roughly accurate time in order to properly authenticate these signatures. This could be achieved by running a time synchronization protocol like NTP [RFC5905] or SNTP [RFC5905], which are already widely used today. TLS clients **MUST** support a mechanism to track and roll over the trust anchor key or be able to avail themselves of a service that does this, as described in Section 8. Security considerations related to mandating the use of this extension are described in Section 7.

The received DNSSEC chain could contain DNS RRs that are not related to the TLSA verification of the intended DNS name. If such an unrelated RR is not DNSSEC signed, it **MUST** be discarded. If the unrelated RRset is DNSSEC signed, the TLS client **MAY** decide to add these RRsets and their DNSSEC signatures to its cache. It **MAY** even pass this data to the local system resolver for caching outside the application. However, care must be taken because caching these records could be used for timing and caching attacks to de-anonymize the TLS client or its user. A TLS client that wants to present the strongest anonymity protection to their users **MUST** refrain from using and caching all unrelated RRs.

## 12. IANA Considerations

IANA has made the following assignment in the "TLS ExtensionType Values" registry:

| Value | Extension Name | TLS 1.3 | Recommended | Reference |
|---|---|---|---|---|
| 59 | dnssec_chain | CH | No | RFC 9102 |

*Table 1*

## 13. References

### 13.1. Normative References

**[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>.

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

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

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

**[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>.

**[RFC5246]** Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

**[RFC6066]** Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <https://www.rfc-editor.org/info/rfc6066>.

**[RFC6698]** Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <https://www.rfc-editor.org/info/rfc6698>.

**[RFC7218]** Gudmundsson, O., "Adding Acronyms to Simplify Conversations about DNS-Based Authentication of Named Entities (DANE)", RFC 7218, DOI 10.17487/RFC7218, April 2014, <https://www.rfc-editor.org/info/rfc7218>.

**[RFC7671]** Dukhovni, V. and W. Hardaker, "The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance", RFC 7671, DOI 10.17487/RFC7671, October 2015, <https://www.rfc-editor.org/info/rfc7671>.

**[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>.

**[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>.

**[RFC8310]** Dickinson, S., Gillmor, D., and T. Reddy, "Usage Profiles for DNS over TLS and DNS over DTLS", RFC 8310, DOI 10.17487/RFC8310, March 2018, <https://www.rfc-editor.org/info/rfc8310>.

**[RFC8446]** Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

## 13.2. Informative References

**[DANE-UKS]** Barnes, R., Thomson, M., and E. Rescorla, "Unknown Key-Share Attacks on DNS-Based Authentications of Named Entities (DANE)", Work in Progress, Internet-Draft, draft-barnes-dane-uks-00, 9 October 2016, <https://datatracker.ietf.org/doc/html/draft-barnes-dane-uks-00>.

**[DISCOVERY]** Gorjon, X. and W. Toorop, "Discovery method for a DNSSEC validating stub resolver", 14 July 2015, <https://www.nlnetlabs.nl/downloads/publications/os3-2015-rp2-xavier-torrent-gorjon.pdf>.

**[DNSOP-SVCB-HTTPS]** Schwartz, B., Bishop, M., and E. Nygren, "Service Binding and Parameter Specification via the DNS (DNS SVCB and HTTPS RRs)", Work in Progress, Internet-Draft, draft-ietf-dnsop-svcb-https-06, 16 June 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-dnsop-svcb-https-06>.

**[RFC5011]** StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors", STD 74, RFC 5011, DOI 10.17487/RFC5011, September 2007, <https://www.rfc-editor.org/info/rfc5011>.

**[RFC5905]** Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

**[RFC7250]** Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <https://www.rfc-editor.org/info/rfc7250>.

**[RFC7901]** Wouters, P., "CHAIN Query Requests in DNS", RFC 7901, DOI 10.17487/RFC7901, June 2016, <https://www.rfc-editor.org/info/rfc7901>.

**[SERIALIZECHAIN]** Langley, A., "Serializing DNS Records with DNSSEC Authentication", Work in Progress, Internet-Draft, draft-agl-dane-serializechain-01, 1 July 2011, <https://datatracker.ietf.org/doc/html/draft-agl-dane-serializechain-01>.

# Appendix A. Test Vectors

The test vectors in this appendix are representations of the content of the "opaque AuthenticationChain" field in DNS presentation format and, except for the `extension_data` in Appendix A.1, do not contain the "uint16 ExtSupportLifetime" field.

For brevity and reproducibility, all DNS zones involved with the test vectors are signed using keys with algorithm 13 (ECDSA Curve P-256 with SHA-256).

To reflect operational practice, different zones in the examples are in different phases of rolling their signing keys:

- All zones use a Key Signing Key (KSK) and Zone Signing Key (ZSK), except for the `example.com` and `example.net` zones, which use a Combined Signing Key (CSK).
- The root and org zones are rolling their ZSKs.
- The com and org zones are rolling their KSKs.

The test vectors are DNSSEC valid in the same period as the certificate is valid, which is between November 28, 2018 and December 2, 2020 with the following root trust anchor:

```
.   IN   DS   ( 47005 13 2 2eb6e9f2480126691594d649a5a613de3052e37861634
           641bb568746f2ffc4d4 )
```

The test vectors will authenticate the certificate used with `https://example.com/`, `https://example.net/`, and `https://example.org/` at the time of writing:


```
-----BEGIN CERTIFICATE-----
MIIHQDCCBiigAwIBAgIQD9B43Ujxor1NDyupa2A4/jANBgkqhkiG9w0BAQsFADBN
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMScwJQYDVQQDEx5E
aWdpQ2VydCBTSEEyIFNlY3VyZSBTZXJ2ZXIgQ0EwHhcNMTgxMTI4MDAwMDAwWhcN
MjAxMjAyMTIwMDAwWjCBpTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3Ju
aWExFDASBgNVBAcTC0xvcyBBbmdlbGVzMTwwOgYDVQQKEzNJbnRlcm5ldCBDb3Jw
b3JhdGlvbiBmb3IgQXNzaWduZWQgTmFtZXMgYW5kIE51bWJlcnMxEzARBgNVBAsT
ClRlY2hub2xvZ3kxGDAWBgNVBAMTD3d3dy5leGFtcGxlLm9yZzCCASIwDQYJKoZI
hvcNAQEBBQADggEPADCCAQoCggEBANDwEnSgliByCGUZElpdStA6jGaPoCkrp9vV
rAzPpXGSFUIVsAeSdjF11yeOTVBqddF7U14nqu3rpGA68o5FGGtFM1yFEaogEv5g
rJ1MRY/d0w4+dw8JwoVlNMci+3QTuUKf9yH28JxEdG3J37Mfj2C3cREGkGNBnY80
eyRJRqzy8I0LSPTTkhr3okXuzOXXg38ugr1x3SgZWDNuEaE6oGpyYJIBWZ9jF3pJ
QnucP9vTBejMh374qvyd0QVQq3WxHrogy4nUbWw3gihMxT98wRD1oKVma1NTydvt
hcNtBfhkp8kO64/hxLHrLWgOFT/l4tz8IWQt7mkrBHjbd2XLVPkCAwEAAaOCA8Ew
ggO9MB8GA1UdIwQYMBaAFA+AYRyCMWHVLyjnjUY4tCzhxtniMB0GA1UdDgQWBBRm
mGIC4AmRp9njNvt2xrC/oW2nvjCBgQYDVR0RBHoweIIPd3d3LmV4YW1wbGUub3Jn
ggtleGFtcGxlLmNvbYILZXhhbXBsZS5lZHWCC2V4YW1wbGUubmV0ggtleGFtcGxl
Lm9yZ4IPd3d3LmV4YW1wbGUuY29tgg93d3cuZXhhbXBsZS5lZHWCD3d3dy5leGFt
cGxlLm5ldDAOBgNVHQ8BAf8EBAMCBaAwHQYDVR0lBBYwFAYIKwYBBQUHAwEGCCsG
AQUFBwMCMGsGA1UdHwRkMGIwL6AtoCuGKWh0dHA6Ly9jcmwzLmRpZ2ljZXJ0LmNv
bS9zc2NhLXNoYTItZzYuY3JsMC+gLaArhilodHRwOi8vY3JsNC5kaWdpY2VydC5j
b20vc3NjYS1zaGEyLWc2LmNybDBMBgNVHSAERTBDMDcGCWCGSAGG/WwBATAqMCgG
CCsGAQUFBwIBFhxodHRwczovL3d3dy5kaWdpY2VydC5jb20vQ1BTMAgGBmeBDAEC
AjB8BggrBgEFBQcBAQRwMG4wJAYIKwYBBQUHMAGGGGh0dHA6Ly9vY3NwLmRpZ2lj
ZXJ0LmNvbTBGBggrBgEFBQcwAoY6aHR0cDovL2NhY2VydHMuZGlnaWNlcnQuY29t
L0RpZ2lDZXJ0U0hBMlNlY3VyZVNlcnZlckNBLmNydDAMBgNVHRMBAf8EAjAAMIIB
fwYKKwYBBAHWeQIEAgSCAW8EggFrAWkAdwCkuQmQtBhYFIe7E6LMZ3AKPDWYBPkb
37jjd80OyA3cEAAAAWdcMZVGAAAEAwBIMEYCIQCEZIG3IR36Gkj1dq5L6EaGVycX
sHvpO7dKV0JsooTEbAIhALuTtf4wxGTkFkx8blhTV+7sf6pFT78ORo7+cP39jkJC
AHYAh3W/51l8+IxDmV+9827/Vo1HVjb/SrVgwbTq/16ggw8AAAFnXDGWFQAABAMA
RzBFAiBvqnfSHKeUwGMtLrOG3UGLQIoaL3+uZsGTX3MfSJNQEQIhANL5nUiGBR6g
l0QlCzzqzvorGXyB/yd7nttYttzo8EpOAHYAb1N2rDHwMRnYmQCkURX/dxUcEdkC
wQApBo2yCJo32RMAAAFnXDGWnAAABAMARzBFAiEA5Hn7Q4SOyqHkT+kDsHq7ku7z
RDuM7P4UDX2ft2Mpny0CIE13WtxJAUr0aASFYZ/XjSAMMfrB0/RxClvWVss9LHKM
MA0GCSqGSIb3DQEBCwUAA4IBAQBzcIXvQEGnakPVeJx7VUjmvGuZhrr7DQOLeP4R
8CmgDM1pFAvGBHiyzvCH1QGdxFl6cf7wbp7BoLCRLR/qPVXFMwUMzcE1GLBqaGZM
v1Yh2lvZSLmMNSGRXdx113pGLCInpm/TOhfrvr0TxRImc8BdozWJavsn1N2qdHQu
N+UBO6bQMLCD0KHEdSGFsuX6ZwAworxTg02/1qiDu7zW7RyzHvFYA4IAjpzvkPIa
X6KjBtpdvp/aXabmL95YgBjT8WJ7pqOfrqhpcmOBZa6Cg6O1l4qbIFH/Gj9hQB5I
0Gs4+eH6F9h3SojmPTYkT+8KuZ9w84Mn+M8qBXUQoYoKgIjN
-----END CERTIFICATE-----
```

### A.1. `_443._tcp.www.example.com`

```
_443._tcp.www.example.com.  3600  IN  TLSA  ( 3 1 1
        8bd1da95272f7fa4ffb24137fc0ed03aae67e5c4d8b3c50734e1050a7920b
        922 )
_443._tcp.www.example.com.  3600  IN  RRSIG  ( TLSA 13 5 3600
        20201202000000 20181128000000 1870 example.com.
        rqY69NnTf4CN3GBGQjKEJCLAMsRkUrXe0JW8IqDb5rQHHzxNqqPeEoi+2vI6S
        z2BhaswpGLVVuoijuVdzxYjmw== )
example.com.  3600  IN  DNSKEY  ( 257 3 13
        JnA1XgyJTZz+psWvbrfUWLV6ULqIJyUS2CQdhUH9VK35bslWeJpRzrlxCUs7s
        /TsSfZMaGWVvlsuieh5nHcXzA== ) ; Key ID = 1870
example.com.  3600  IN  RRSIG  ( DNSKEY 13 2 3600
        20201202000000 20181128000000 1870 example.com.
        nYisnu/26Sw1qmGuREa9o/fLgYuA4oNPt4+6PMBZoN0MS8Gjtli9NVRYeSIzt
        QHPGSpvRxTUC4tZi62z1UgGDw== )
example.com.  172800  IN  DS  ( 1870 13 2 e9b533a049798e900b5c29c90cd
       25a986e8a44f319ac3cd302bafc08f5b81e16)
example.com.  172800  IN  RRSIG  ( DS 13 2 172800
        20201202000000 20181128000000 34327 com.
        sEAKvX4H6pJfN8nKcclB1NRcRSPOztx8omr4fCSHu6lp+uESP/Le4iF2sKukO
        J1hhWSB6jgubEVl17rGNOA/YQ== )
com.  172800  IN  DNSKEY  ( 256 3 13
        7IIE5Dol8jSMUqHTvOOiZapdEbQ9wqRxFi/zQcSdufUKLhpByvLpzSAQTqCWj
        3URIZ8L3Fa2gBLMOZUzZ1GQCw== ) ; Key ID = 34327
com.  172800  IN  DNSKEY  ( 257 3 13
        RbkcO+96XZmnp8jYIuM4lryAp3egQjSmBaSoiA7H76Tm0RLHPNPUxlVk+nQ0f
        Ic3I8xfZDNw8Wa0Pe3/g2QA/w== ) ; Key ID = 18931
com.  172800  IN  DNSKEY  ( 257 3 13
        szc7biLo5J4OHlkan1vZrF4aD4YYf+NHA/GAqdNslY9xxK9Izg68XHkqck4Rt
        DiVk37lNAQmgSlHbrGu0yOTkA== ) ; Key ID = 28809
com.  172800  IN  RRSIG  ( DNSKEY 13 1 172800 20201202000000
        20181128000000 18931 com.
        LJ4p5ORS2ViILwTotSlWixElqRXHY5tOdIuHlPWTdBGPMq3y40QNr1V+ZOyA5
        7LFdPKpcvb8BvhM+GqKWGBEsg== )
com.  172800  IN  RRSIG  ( DNSKEY 13 1 172800 20201202000000
        20181128000000 28809 com.
        sO+4X2N21yS6x8+dBVBzbRo9+55MM8n7+RUvdBuxRFVh6JaBlqDOC5LLkl7Ev
        mDXqz6KEhhQjT+aQWDt6WFHlA== )
com.  86400  IN  DS  ( 18931 13 2 20f7a9db42d0e2042fbbb9f9ea015941202
        f9eabb94487e658c188e7bcb52115 )
com.  86400  IN  DS  ( 28809 13 2 ad66b3276f796223aa45eda773e92c6d98e
        70643bbde681db342a9e5cf2bb380 )
com.  86400  IN  RRSIG  ( DS 13 1 86400 20201202000000
        20181128000000 31918 .
        nDiDlBjXEE/6AudhC++Hui1ckPcuAnGbjEASNoxA3ZHjlXRzL050UzePko5Pb
        vBKTf6pk8JRCqnfzlo2QY+WXA== )
.  86400  IN  DNSKEY  ( 256 3 13
        zKz+DCWkNA/vuheiVPcGqsH40U84KZAlrMRIyozj9WHzf8PsFp/oR8j8vmjjW
        P98cbte4d8NvlGLxzbUzo3+FA== ) ; Key ID = 31918
.  86400  IN  DNSKEY  ( 256 3 13
        8wMZZ4lzHdyKZ4fv8kys/t3QMlgvEadbsbyqWrMhwddSXCZYGRrsAbPpireRW
        xbVcd1VtOrlFBcRDMTN0R0XEQ== ) ; Key ID = 2635
.  86400  IN  DNSKEY  ( 257 3 13
        yvX+VNTUjxZiGvtr060hVbrPV9H6rVusQtF9lIxCFzbZOJxMQBFmbqlc8Xclv
        Q+gDOXnFOTsgs/frMmxyGOtRg== ) ; Key ID = 47005
.  86400  IN  RRSIG  ( DNSKEY 13 0 86400 20201202000000
        20181128000000 47005 .
```

```
        0EPW1ca+N/ZhZPKla77STG734cTeIOjUwq7eW0HsnOfudWmnCEVeco2wLLq9m
        nBT1dtNjIczvLG9pQTnOKUsHQ== )
```

A hex dump of the `extension_data` of the server's `dnssec_chain` extension representation of this with an ExtSupportLifetime value of 0 is:

```
0000:  00 00 04 5f 34 34 33 04  5f 74 63 70 03 77 77 77
0010:  07 65 78 61 6d 70 6c 65  03 63 6f 6d 00 00 34 00
0020:  01 00 00 0e 10 00 23 03  01 01 8b d1 da 95 27 2f
0030:  7f a4 ff b2 41 37 fc 0e  d0 3a ae 67 e5 c4 d8 b3
0040:  c5 07 34 e1 05 0a 79 20  b9 22 04 5f 34 34 33 04
0050:  5f 74 63 70 03 77 77 77  07 65 78 61 6d 70 6c 65
0060:  03 63 6f 6d 00 00 2e 00  01 00 00 0e 10 00 5f 00
0070:  34 0d 05 00 00 0e 10 5f  c6 d9 00 5b fd da 80 07
0080:  4e 07 65 78 61 6d 70 6c  65 03 63 6f 6d 00 ce 1d
0090:  3a de b7 dc 7c ee 65 6d  61 cf b4 72 c5 97 7c 8c
00a0:  9c ae ae 9b 76 51 55 c5  18 fb 10 7b 6a 1f e0 35
00b0:  5f ba af 75 3c 19 28 32  fa 62 1f a7 3a 8b 85 ed
00c0:  79 d3 74 11 73 87 59 8f  cc 81 2e 1e f3 fb 07 65
00d0:  78 61 6d 70 6c 65 03 63  6f 6d 00 00 30 00 01 00
00e0:  00 0e 10 00 44 01 01 03  0d 26 70 35 5e 0c 89 4d
00f0:  9c fe a6 c5 af 6e b7 d4  58 b5 7a 50 ba 88 27 25
0100:  12 d8 24 1d 85 41 fd 54  ad f9 6e c9 56 78 9a 51
0110:  ce b9 71 09 4b 3b b3 f4  ec 49 f6 4c 68 65 95 be
0120:  5b 2e 89 e8 79 9c 77 17  cc 07 65 78 61 6d 70 6c
0130:  65 03 63 6f 6d 00 00 2e  00 01 00 00 0e 10 00 5f
0140:  00 30 0d 02 00 00 0e 10  5f c6 d9 00 5b fd da 80
0150:  07 4e 07 65 78 61 6d 70  6c 65 03 63 6f 6d 00 46
0160:  28 38 30 75 b8 e3 4b 74  3a 20 9b 27 ae 14 8d 11
0170:  0d 4e 1a 24 61 38 a9 10  83 24 9c b4 a1 2a 2d 9b
0180:  c4 c2 d7 ab 5e b3 af b9  f5 d1 03 7e 4d 5d a8 33
0190:  9c 16 2a 92 98 e9 be 18  07 41 a8 ca 74 ac cc 07
01a0:  65 78 61 6d 70 6c 65 03  63 6f 6d 00 00 2b 00 01
01b0:  00 02 a3 00 00 24 07 4e  0d 02 e9 b5 33 a0 49 79
01c0:  8e 90 0b 5c 29 c9 0c d2  5a 98 6e 8a 44 f3 19 ac
01d0:  3c d3 02 ba fc 08 f5 b8  1e 16 07 65 78 61 6d 70
01e0:  6c 65 03 63 6f 6d 00 00  2e 00 01 00 02 a3 00 00
01f0:  57 00 2b 0d 02 00 02 a3  00 5f c6 d9 00 5b fd da
0200:  80 86 17 03 63 6f 6d 00  a2 03 e7 04 a6 fa cb eb
0210:  13 fc 93 84 fd d6 de 6b  50 de 56 59 27 1f 38 ce
0220:  81 49 86 84 e6 36 31 72  d4 7e 23 19 fd b4 a2 2a
0230:  58 a2 31 ed c2 f1 ff 4f  b2 81 1a 18 07 be 72 cb
0240:  52 41 aa 26 fd ae e0 39  03 63 6f 6d 00 00 30 00
0250:  01 00 02 a3 00 00 44 01  00 03 0d ec 82 04 e4 3a
0260:  25 f2 34 8c 52 a1 d3 bc  e3 a2 65 aa 5d 11 b4 3d
0270:  c2 a4 71 16 2f f3 41 c4  9d b9 f5 0a 2e 1a 41 ca
0280:  f2 e9 cd 20 10 4e a0 96  8f 75 11 21 9f 0b dc 56
0290:  b6 80 12 cc 39 95 33 67  51 90 0b 03 63 6f 6d 00
02a0:  00 30 00 01 00 02 a3 00  00 44 01 01 03 0d 45 b9
02b0:  1c 3b ef 7a 5d 99 a7 a7  c8 d8 22 e3 38 96 bc 80
02c0:  a7 77 a0 42 34 a6 05 a4  a8 88 0e c7 ef a4 e6 d1
02d0:  12 c7 3c d3 d4 c6 55 64  fa 74 34 7c 87 37 23 cc
02e0:  5f 64 33 70 f1 66 b4 3d  ed ff 83 64 00 ff 03 63
02f0:  6f 6d 00 00 30 00 01 00  02 a3 00 00 44 01 01 03
0300:  0d b3 37 3b 6e 22 e8 e4  9e 0e 1e 59 1a 9f 5b d9
0310:  ac 5e 1a 0f 86 18 7f e3  47 03 f1 80 a9 d3 6c 95
0320:  8f 71 c4 af 48 ce 0e bc  5c 79 2a 72 4e 11 b4 38
0330:  95 93 7e e5 34 04 26 81  29 47 6e b1 ae d3 23 93
0340:  90 03 63 6f 6d 00 00 2e  00 01 00 02 a3 00 00 57
0350:  00 30 0d 01 00 02 a3 00  5f c6 d9 00 5b fd da 80
0360:  49 f3 03 63 6f 6d 00 18  a9 48 eb 23 d4 4f 80 ab
0370:  c9 92 38 fc b4 3c 5a 18  de be 57 00 4f 73 43 59
0380:  3f 6d eb 6e d7 1e 04 65  4a 43 3f 7a a1 97 21 30
```

```
0390:  d9 bd 92 1c 73 dc f6 3f  cf 66 5f 2f 05 a0 aa eb
03a0:  af b0 59 dc 12 c9 65 03  63 6f 6d 00 00 2e 00 01
03b0:  00 02 a3 00 00 57 00 30  0d 01 00 02 a3 00 5f c6
03c0:  d9 00 5b fd da 80 70 89  03 63 6f 6d 00 61 70 e6
03d0:  95 9b d9 ed 6e 57 58 37  b6 f5 80 bd 99 db d2 4a
03e0:  44 68 2b 0a 35 96 26 a2  46 b1 81 2f 5f 90 96 b7
03f0:  5e 15 7e 77 84 8f 06 8a  e0 08 5e 1a 60 9f c1 92
0400:  98 c3 3b 73 68 63 fb cc  d4 d8 1f 5e b2 03 63 6f
0410:  6d 00 00 2b 00 01 00 01  51 80 00 24 49 f3 0d 02
0420:  20 f7 a9 db 42 d0 e2 04  2f bb b9 f9 ea 01 59 41
0430:  20 2f 9e ab b9 44 87 e6  58 c1 88 e7 bc b5 21 15
0440:  03 63 6f 6d 00 00 2b 00  01 00 01 51 80 00 24 70
0450:  89 0d 02 ad 66 b3 27 6f  79 62 23 aa 45 ed a7 73
0460:  e9 2c 6d 98 e7 06 43 bb  de 68 1d b3 42 a9 e5 cf
0470:  2b b3 80 03 63 6f 6d 00  00 2e 00 01 00 01 51 80
0480:  00 53 00 2b 0d 01 00 01  51 80 5f c6 d9 00 5b fd
0490:  da 80 7c ae 00 12 2e 27  6d 45 d9 e9 81 6f 79 22
04a0:  ad 6e a2 e7 3e 82 d2 6f  ce 0a 4b 71 86 25 f3 14
04b0:  53 1a c9 2f 8a e8 24 18  df 9b 89 8f 98 9d 32 e8
04c0:  0b c4 de ab a7 c4 a7 c8  f1 72 ad b5 7c ed 7f b5
04d0:  e7 7a 78 4b 07 00 00 30  00 01 00 01 51 80 00 44
04e0:  01 00 03 0d cc ac fe 0c  25 a4 34 0f ef ba 17 a2
04f0:  54 f7 06 aa c1 f8 d1 4f  38 29 90 25 ac c4 48 ca
0500:  8c e3 f5 61 f3 7f c3 ec  16 9f e8 47 c8 fc be 68
0510:  e3 58 ff 7c 71 bb 5e e1  df 0d be 51 8b c7 36 d4
0520:  ce 8d fe 14 00 00 30 00  01 00 01 51 80 00 44 01
0530:  00 03 0d f3 03 19 67 89  73 1d dc 8a 67 87 ef f2
0540:  4c ac fe dd d0 32 58 2f  11 a7 5b b1 bc aa 5a b3
0550:  21 c1 d7 52 5c 26 58 19  1a ec 01 b3 e9 8a b7 91
0560:  5b 16 d5 71 dd 55 b4 ea  e5 14 17 11 0c c4 cd d1
0570:  1d 17 11 00 00 30 00 01  00 01 51 80 00 44 01 01
0580:  03 0d ca f5 fe 54 d4 d4  8f 16 62 1a fb 6b d3 ad
0590:  21 55 ba cf 57 d1 fa ad  5b ac 42 d1 7d 94 8c 42
05a0:  17 36 d9 38 9c 4c 40 11  66 6e a9 5c f1 77 25 bd
05b0:  0f a0 0c e5 e7 14 e4 ec  82 cf df ac c9 b1 c8 63
05c0:  ad 46 00 00 2e 00 01 00  01 51 80 00 53 00 30 0d
05d0:  00 00 01 51 80 5f c6 d9  00 5b fd da 80 b7 9d 00
05e0:  de 7a 67 40 ee ec ba 4b  da 1e 5c 2d d4 89 9b 2c
05f0:  96 58 93 f3 78 6c e7 47  f4 1e 50 d9 de 8c 0a 72
0600:  df 82 56 0d fb 48 d7 14  de 32 83 ae 99 a4 9c 0f
0610:  cb 50 d3 aa ad b1 a3 fc  62 ee 3a 8a 09 88 b6 be
```

## A.2. `_25._tcp.example.com` NSEC Wildcard

```
_25._tcp.example.com.  3600  IN  TLSA  ( 3 1 1
        8bd1da95272f7fa4ffb24137fc0ed03aae67e5c4d8b3c50734e1050a7920b
        922 )
_25._tcp.example.com.  3600  IN  RRSIG  ( TLSA 13 3 3600
        20201202000000 20181128000000 1870 example.com.
        BZawXvte5SyF8hnXviKDWql15E2v+RMXqaSE+NOcAMlZOrSMUkfyPqvkv53K2
        rfL4DFP8rO3VMgI0v+ogrox0w== )
*._tcp.example.com.  3600  IN  NSEC  ( smtp.example.com. RRSIG
        NSEC TLSA )
*._tcp.example.com.  3600  IN  RRSIG  ( NSEC 13 3 3600
        20201202000000 20181128000000 1870 example.com.
        K6u8KrR8ca5bjtbce3w8yjMXr9vw12225lAwyIHpxptY43OMLCUCenwpYW5qd
        mpFvAacqj4+tSkKiN279SI9pA== )
example.com.  3600  IN  DNSKEY  ( 257 3 13
        JnA1XgyJTZz+psWvbrfUWLV6ULqIJyUS2CQdhUH9VK35bslWeJpRzrlxCUs7s
        /TsSfZMaGWVvlsuieh5nHcXzA== ) ; Key ID = 1870
example.com.  3600  IN  RRSIG  ( DNSKEY 13 2 3600
        20201202000000 20181128000000 1870 example.com.
        nYisnu/26Sw1qmGuREa9o/fLgYuA4oNPt4+6PMBZoN0MS8Gjtli9NVRYeSIzt
        QHPGSpvRxTUC4tZi62z1UgGDw== )
example.com.  172800  IN  DS  ( 1870 13 2 e9b533a049798e900b5c29c90cd
        25a986e8a44f319ac3cd302bafc08f5b81e16 )
example.com.  172800  IN  RRSIG  ( DS 13 2 172800
        20201202000000 20181128000000 34327 com.
        sEAKvX4H6pJfN8nKcclB1NRcRSPOztx8omr4fCSHu6lp+uESP/Le4iF2sKukO
        J1hhWSB6jgubEVl17rGNOA/YQ== )
com.  172800  IN  DNSKEY  ( 256 3 13
        7IIE5Dol8jSMUqHTvOOiZapdEbQ9wqRxFi/zQcSdufUKLhpByvLpzSAQTqCWj
        3URIZ8L3Fa2gBLMOZUzZ1GQCw== ) ; Key ID = 34327
com.  172800  IN  DNSKEY  ( 257 3 13
        RbkcO+96XZmnp8jYIuM4lryAp3egQjSmBaSoiA7H76Tm0RLHPNPUxlVk+nQ0f
        Ic3I8xfZDNw8Wa0Pe3/g2QA/w== ) ; Key ID = 18931
com.  172800  IN  DNSKEY  ( 257 3 13
        szc7biLo5J4OHlkan1vZrF4aD4YYf+NHA/GAqdNslY9xxK9Izg68XHkqck4Rt
        DiVk37lNAQmgSlHbrGu0yOTkA== ) ; Key ID = 28809
com.  172800  IN  RRSIG  ( DNSKEY 13 1 172800 20201202000000
        20181128000000 18931 com.
        LJ4p5ORS2ViILwTotSlWixElqRXHY5tOdIuHlPWTdBGPMq3y40QNr1V+ZOyA5
        7LFdPKpcvb8BvhM+GqKWGBEsg== )
com.  172800  IN  RRSIG  ( DNSKEY 13 1 172800 20201202000000
        20181128000000 28809 com.
        sO+4X2N21yS6x8+dBVBzbRo9+55MM8n7+RUvdBuxRFVh6JaBlqDOC5LLkl7Ev
        mDXqz6KEhhQjT+aQWDt6WFHlA== )
com.  86400  IN  DS  ( 18931 13 2 20f7a9db42d0e2042fbbb9f9ea015941202
        f9eabb94487e658c188e7bcb52115 )
com.  86400  IN  DS  ( 28809 13 2 ad66b3276f796223aa45eda773e92c6d98e
        70643bbde681db342a9e5cf2bb380 )
com.  86400  IN  RRSIG  ( DS 13 1 86400 20201202000000
        20181128000000 31918 .
        nDiDlBjXEE/6AudhC++Hui1ckPcuAnGbjEASNoxA3ZHjlXRzL050UzePko5Pb
        vBKTf6pk8JRCqnfzlo2QY+WXA== )
.  86400  IN  DNSKEY  ( 256 3 13
        zKz+DCWkNA/vuheiVPcGqsH40U84KZAlrMRIyozj9WHzf8PsFp/oR8j8vmjjW
        P98cbte4d8NvlGLxzbUzo3+FA== ) ; Key ID = 31918
.  86400  IN  DNSKEY  ( 256 3 13
        8wMZZ4lzHdyKZ4fv8kys/t3QMlgvEadbsbyqWrMhwddSXCZYGRrsAbPpireRW
        xbVcd1VtOrlFBcRDMTN0R0XEQ== ) ; Key ID = 2635
```

```
.  86400  IN  DNSKEY  ( 257 3 13
       yvX+VNTUjxZiGvtr060hVbrPV9H6rVusQtF9lIxCFzbZOJxMQBFmbqlc8Xclv
       Q+gDOXnFOTsgs/frMmxyGOtRg== ) ; Key ID = 47005
.  86400  IN  RRSIG  ( DNSKEY 13 0 86400 20201202000000
       20181128000000 47005 .
       0EPW1ca+N/ZhZPKla77STG734cTeIOjUwq7eW0HsnOfudWmnCEVeco2wLLq9m
       nBT1dtNjIczvLG9pQTnOKUsHQ== )
```

### A.3. `_25._tcp.example.org` NSEC3 Wildcard

```
_25._tcp.example.org.  3600  IN  TLSA  ( 3 1 1
        8bd1da95272f7fa4ffb24137fc0ed03aae67e5c4d8b3c50734e1050a7920b
        922 )
_25._tcp.example.org.  3600  IN  RRSIG  ( TLSA 13 3 3600
        20201202000000 20181128000000 56566 example.org.
        lNp6th/CJel5WsYlLsLadcQ/YdSTJAIOttzYKnNkNzeZ0jxtDyEP818Q1R4lL
        cYzJ7vCvqb9gFCiCJjK2gAamw== )
dlm7rss9pejqnh0ev6h7k1ikqqcl5mae.example.org.  3600  IN  NSEC3  (
        1 0 1 - t6lf7uuoi0qofq0nvdjroavo46pp20im RRSIG TLSA )
dlm7rss9pejqnh0ev6h7k1ikqqcl5mae.example.org.  3600  IN  RRSIG  (
        NSEC3 13 3 3600 20201202000000 20181128000000 56566
        example.org.
        guUyy9LIZlYb0FZttAdYJGrFNKpKu91Tm+dPOz98rnpwIlwwvLifXIvIl90nE
        X38cWzEQOpreJu3t4WAfPsxdg== )
example.org.  3600  IN  DNSKEY  ( 256 3 13
        NrbL6utGqIW1wrhhjeexdA6bMdD1lC1hj0Fnpevaa1AMyY2uy83TmoGnR996N
        UR5TlG4Zh+YPbbmUIixe4nS3w== ) ; Key ID = 56566
example.org.  3600  IN  DNSKEY  ( 257 3 13
        uspaqp17jsMTX6AWVgmbog/3Sttz+9ANFUWLn6qKUHr0BOqRuChQWj8jyYUUr
        Wy9txxesNQ9MkO4LUrFght1LQ== ) ; Key ID = 44384
example.org.  3600  IN  RRSIG  ( DNSKEY 13 2 3600
        20201202000000 20181128000000 44384 example.org.
        ttse9pYp9PSu0pJ+TOpIVFLWJ6NKOMWZX4Q/SlU6ZfaiKQc0Bg7Tut9+wPunk
        6OPPvyHjVXMAsvk0tqV0B+/ag== )
example.org.  86400  IN  DS  ( 44384 13 2 ec307e2efc8f0117ed96ab48a51
        3c8003e1d9121f1ff11a08b4cdd348d090aa6 )
example.org.  86400  IN  RRSIG  ( DS 13 2 86400 20201202000000
        20181128000000 9523 org.
        m86Xz0CEa2sWG40a0bS2kqLKPmIlyiVyDeoWXAq3djeGiPaikLuKORNzWXu62
        clpAfvZHx59Ackst4X+zXYpUA== )
org.  86400  IN  DNSKEY  ( 256 3 13
        fuLp60znhSSEr9HowILpTpyLKQdM6ixcgkTE0gqVdsLx+DSNHSc69o6fLWC0e
        HfWx7kzlBBoJB0vLrvsJtXJ6g== ) ; Key ID = 47417
org.  86400  IN  DNSKEY  ( 256 3 13
        zTHbb7JM627Bjr8CGOySUarsic91xZU3vvLJ5RjVix9YH6+iwpBXb6qfHyQHy
        mlMiAAoaoXh7BUkEBVgDVN8sQ== ) ; Key ID = 9523
org.  86400  IN  DNSKEY  ( 257 3 13
        Uf24EyNt51DMcLV+dHPInhSpmjPnqAQNUTouU+SGLu+lFRRlBetgw1bJUZNI6
        Dlger0VJTm0QuX/JVXcyGVGoQ== ) ; Key ID = 49352
org.  86400  IN  DNSKEY  ( 257 3 13
        0SZfoe8Yx+eoaGgyAGEeJax/ZBV1AuG+/smcOgRm+F6doNlgc3lddcM1MbTvJ
        HTjK6Fvy8W6yZ+cAptn8sQheg== ) ; Key ID = 12651
org.  86400  IN  RRSIG  ( DNSKEY 13 1 86400 20201202000000
        20181128000000 12651 org.
        Gq9wf+z3pasXXUwE210jYc0LhJnMAhcwXydnvkHtCVY6/0jUafHO4RksN84Zt
        us0pUgWngbT/OWXskdMYXZU4A== )
org.  86400  IN  RRSIG  ( DNSKEY 13 1 86400 20201202000000
        20181128000000 49352 org.
        VGEkEMWBJ2IbOpm2Z56Qxu2NGPcVUDWCbYRyk+Qk1+HzGtyd2qPEKkpgMs/0p
        vZEMj1YXD+dIqb2nUK9PGBAXw== )
org.  86400  IN  DS  ( 12651 13 2 3979a51f98bbf219fcaf4a4176e766dfa8f
        9db5c24a75743eb1e704b97a9fabc )
org.  86400  IN  DS  ( 49352 13 2 03d11a1aa114abbb8f708c3c0ff0db765fe
        f4a2f18920db5f58710dd767c293b )
org.  86400  IN  RRSIG  ( DS 13 1 86400 20201202000000
        20181128000000 31918 .
        adiFuP2UIulQw5Edsb/7WSPqr5nkRSTVXbZ2tkBeZRQcMjdCD3pyonWO5JPRV
```

```
        EemgaE357S4pX5D0tVZzeZJ6A== )
.  86400  IN  DNSKEY  ( 256 3 13
        zKz+DCWkNA/vuheiVPcGqsH40U84KZAlrMRIyozj9WHzf8PsFp/oR8j8vmjjW
        P98cbte4d8NvlGLxzbUzo3+FA== ) ; Key ID = 31918
.  86400  IN  DNSKEY  ( 256 3 13
        8wMZZ4lzHdyKZ4fv8kys/t3QMlgvEadbsbyqWrMhwddSXCZYGRrsAbPpireRW
        xbVcd1VtOrlFBcRDMTN0R0XEQ== ) ; Key ID = 2635
.  86400  IN  DNSKEY  ( 257 3 13
        yvX+VNTUjxZiGvtr060hVbrPV9H6rVusQtF9lIxCFzbZOJxMQBFmbqlc8Xclv
        Q+gDOXnFOTsgs/frMmxyGOtRg== ) ; Key ID = 47005
.  86400  IN  RRSIG  ( DNSKEY 13 0 86400 20201202000000
        20181128000000 47005 .
        0EPW1ca+N/ZhZPKla77STG734cTeIOjUwq7eW0HsnOfudWmnCEVeco2wLLq9m
        nBT1dtNjIczvLG9pQTnOKUsHQ== )
```

### A.4. `_443._tcp.www.example.org` CNAME

```
_443._tcp.www.example.org.  3600  IN  CNAME  (
        dane311.example.org. )
_443._tcp.www.example.org.  3600  IN  RRSIG  ( CNAME 13 5 3600
        20201202000000 20181128000000 56566 example.org.
        R0dUe6Rt4G+2ablrQH9Zw8j9NhBLMgNYTI5+H7nO8SNz5Nm8w0NZrXv3Qp7gx
        Qb/a90O696120NsYaZX2+ebBA== )
dane311.example.org.  3600  IN  TLSA  ( 3 1 1
        8bd1da95272f7fa4ffb24137fc0ed03aae67e5c4d8b3c50734e1050a7920b
        922 )
dane311.example.org.  3600  IN  RRSIG  ( TLSA 13 3 3600
        20201202000000 20181128000000 56566 example.org.
        f6TbTZTpu3h6MYpLkKQwWILAkYQ3EUY+Nsoa6any6yt+aeuunMUjw+IJB2QLm
        0x0PrD7m39JA3NUSkUp9riNNQ== )
example.org.  3600  IN  DNSKEY  ( 256 3 13
        NrbL6utGqIW1wrhhjeexdA6bMdD1lC1hj0Fnpevaa1AMyY2uy83TmoGnR996N
        UR5TlG4Zh+YPbbmUIixe4nS3w== ) ; Key ID = 56566
example.org.  3600  IN  DNSKEY  ( 257 3 13
        uspaqp17jsMTX6AWVgmbog/3Sttz+9ANFUWLn6qKUHr0BOqRuChQWj8jyYUUr
        Wy9txxesNQ9MkO4LUrFght1LQ== ) ; Key ID = 44384
example.org.  3600  IN  RRSIG  ( DNSKEY 13 2 3600
        20201202000000 20181128000000 44384 example.org.
        ttse9pYp9PSu0pJ+TOpIVFLWJ6NKOMWZX4Q/SlU6ZfaiKQc0Bg7Tut9+wPunk
        6OPPvyHjVXMAsvk0tqV0B+/ag== )
example.org.  86400  IN  DS  ( 44384 13 2 ec307e2efc8f0117ed96ab48a51
        3c8003e1d9121f1ff11a08b4cdd348d090aa6 )
example.org.  86400  IN  RRSIG  ( DS 13 2 86400 20201202000000
        20181128000000 9523 org.
        m86Xz0CEa2sWG40a0bS2kqLKPmIlyiVyDeoWXAq3djeGiPaikLuKORNzWXu62
        clpAfvZHx59Ackst4X+zXYpUA== )
org.  86400  IN  DNSKEY  ( 256 3 13
        fuLp60znhSSEr9HowILpTpyLKQdM6ixcgkTE0gqVdsLx+DSNHSc69o6fLWC0e
        HfWx7kzlBBoJB0vLrvsJtXJ6g== ) ; Key ID = 47417
org.  86400  IN  DNSKEY  ( 256 3 13
        zTHbb7JM627Bjr8CGOySUarsic91xZU3vvLJ5RjVix9YH6+iwpBXb6qfHyQHy
        mlMiAAoaoXh7BUkEBVgDVN8sQ== ) ; Key ID = 9523
org.  86400  IN  DNSKEY  ( 257 3 13
        Uf24EyNt51DMcLV+dHPInhSpmjPnqAQNUTouU+SGLu+lFRRlBetgw1bJUZNI6
        Dlger0VJTm0QuX/JVXcyGVGoQ== ) ; Key ID = 49352
org.  86400  IN  DNSKEY  ( 257 3 13
        0SZfoe8Yx+eoaGgyAGEeJax/ZBV1AuG+/smcOgRm+F6doNlgc3lddcM1MbTvJ
        HTjK6Fvy8W6yZ+cAptn8sQheg== ) ; Key ID = 12651
org.  86400  IN  RRSIG  ( DNSKEY 13 1 86400 20201202000000
        20181128000000 12651 org.
        Gq9wf+z3pasXXUwE210jYc0LhJnMAhcwXydnvkHtCVY6/0jUafHO4RksN84Zt
        us0pUgWngbT/OWXskdMYXZU4A== )
org.  86400  IN  RRSIG  ( DNSKEY 13 1 86400 20201202000000
        20181128000000 49352 org.
        VGEkEMWBJ2IbOpm2Z56Qxu2NGPcVUDWCbYRyk+Qk1+HzGtyd2qPEKkpgMs/0p
        vZEMj1YXD+dIqb2nUK9PGBAXw== )
org.  86400  IN  DS  ( 12651 13 2 3979a51f98bbf219fcaf4a4176e766dfa8f
        9db5c24a75743eb1e704b97a9fabc )
org.  86400  IN  DS  ( 49352 13 2 03d11a1aa114abbb8f708c3c0ff0db765fe
        f4a2f18920db5f58710dd767c293b )
org.  86400  IN  RRSIG  ( DS 13 1 86400 20201202000000
        20181128000000 31918 .
        adiFuP2UIulQw5Edsb/7WSPqr5nkRSTVXbZ2tkBeZRQcMjdCD3pyonWO5JPRV
        EemgaE357S4pX5D0tVZzeZJ6A== )
```

```
.  86400  IN  DNSKEY  ( 256 3 13
        zKz+DCWkNA/vuheiVPcGqsH40U84KZAlrMRIyozj9WHzf8PsFp/oR8j8vmjjW
        P98cbte4d8NvlGLxzbUzo3+FA== ) ; Key ID = 31918
.  86400  IN  DNSKEY  ( 256 3 13
        8wMZZ4lzHdyKZ4fv8kys/t3QMlgvEadbsbyqWrMhwddSXCZYGRrsAbPpireRW
        xbVcd1VtOrlFBcRDMTN0R0XEQ== ) ; Key ID = 2635
.  86400  IN  DNSKEY  ( 257 3 13
        yvX+VNTUjxZiGvtr060hVbrPV9H6rVusQtF9lIxCFzbZOJxMQBFmbqlc8Xclv
        Q+gDOXnFOTsgs/frMmxyGOtRg== ) ; Key ID = 47005
.  86400  IN  RRSIG  ( DNSKEY 13 0 86400 20201202000000
        20181128000000 47005 .
        0EPW1ca+N/ZhZPKla77STG734cTeIOjUwq7eW0HsnOfudWmnCEVeco2wLLq9m
        nBT1dtNjIczvLG9pQTnOKUsHQ== )
```

### A.5. _443._tcp.www.example.net DNAME

```
example.net.  3600  IN  DNAME  example.com.
example.net.  3600  IN  RRSIG  ( DNAME 13 2 3600 20201202000000
        20181128000000 48085 example.net.
        o3uV5k5Ewp5fdrOZt0n4QuH+/Hpku2Lo3CzGRt9/MS2zZt2Qb/AXz435UFQBx
        OI/pDnjJcLSd/gBLtqR52WLMA== )
; _443._tcp.www.example.net.  3600  IN  CNAME  (
;         _443._tcp.www.example.com. )
_443._tcp.www.example.com.  3600  IN  TLSA  ( 3 1 1
        8bd1da95272f7fa4ffb24137fc0ed03aae67e5c4d8b3c50734e1050a7920b
        922 )
_443._tcp.www.example.com.  3600  IN  RRSIG  ( TLSA 13 5 3600
        20201202000000 20181128000000 1870 example.com.
        rqY69NnTf4CN3GBGQjKEJCLAMsRkUrXe0JW8IqDb5rQHHzxNqqPeEoi+2vI6S
        z2BhaswpGLVVuoijuVdzxYjmw== )
example.net.  3600  IN  DNSKEY  ( 257 3 13
        X9GHpJcS7bqKVEsLiVAbddHUHTZqqBbVa3mzIQmdp+5cTJk7qDazwH68Kts8d
        9MvN55HddWgsmeRhgzePz6hMg== ) ; Key ID = 48085
example.net.  3600  IN  RRSIG  ( DNSKEY 13 2 3600
        20201202000000 20181128000000 48085 example.net.
        CkwqgEt1p97oMa3w5LctIjKIuG5XVSapKrfwuHhb5p04fWXRMNsXasG/kd2F/
        wlmMWiq38gOQaYCLNm+cjQzpQ== )
example.net.  172800  IN  DS  ( 48085 13 2 7c1998ce683df60e2fa41460c4
        53f88f463dac8cd5d074277b4a7c04502921be )
example.net.  172800  IN  RRSIG  ( DS 13 2 172800
        20201202000000 20181128000000 10713 net.
        w0JxDeiBJZNlpCdxKtRENlqfTpSxcs6Vftscsyfo/hyeTPYcIt4yItDkYsYK+
        KQ6FYAVE4nisA3vDQoZVL4wow== )
net.  172800  IN  DNSKEY  ( 256 3 13
        061EoQs4sBcDsPiz17vt4nFSGLmXAGguqLStOesmKNCimi4/lw/vtyfqALuLF
        JiFjtCK3HMPi8HQ1jbGEwbGCA== ) ; Key ID = 10713
net.  172800  IN  DNSKEY  ( 257 3 13
        LkNCPE+v3S4MVnsOqZFhn8n2NSwtLYOZLZjjgVsAKgu4XZncaDgq1R/7ZXRO5
        oVx2zthxuu2i+mGbRrycAaCvA== ) ; Key ID = 485
net.  172800  IN  RRSIG  ( DNSKEY 13 1 172800 20201202000000
        20181128000000 485 net.
        031jXg06zSuDwI5zqYuYFJg1O5p+zy85csMXagvRxB9W2lL/wJRi6Gn9BcaCV
        RnDId5WR+yCADhsbKfSrrd9vQ== )
net.  86400  IN  DS  ( 485 13 2 ab25a2941aa7f1eb8688bb783b25587515a0c
        d8c247769b23adb13ca234d1c05 )
net.  86400  IN  RRSIG  ( DS 13 1 86400 20201202000000
        20181128000000 31918 .
        vOXoTjxggGTYKIwssQ3kpML0ag6D0Hcm+Syy7++4zT7gaFHfRH9a6uZekIWdb
        oss8y7q4onW4rxKdtw2S28hwQ== )
.  86400  IN  DNSKEY  ( 256 3 13
        zKz+DCWkNA/vuheiVPcGqsH40U84KZAlrMRIyozj9WHzf8PsFp/oR8j8vmjjW
        P98cbte4d8NvlGLxzbUzo3+FA== ) ; Key ID = 31918
.  86400  IN  DNSKEY  ( 256 3 13
        8wMZZ4lzHdyKZ4fv8kys/t3QMlgvEadbsbyqWrMhwddSXCZYGRrsAbPpireRW
        xbVcd1VtOrlFBcRDMTN0R0XEQ== ) ; Key ID = 2635
.  86400  IN  DNSKEY  ( 257 3 13
        yvX+VNTUjxZiGvtr060hVbrPV9H6rVusQtF9lIxCFzbZOJxMQBFmbqlc8Xclv
        Q+gDOXnFOTsgs/frMmxyGOtRg== ) ; Key ID = 47005
.  86400  IN  RRSIG  ( DNSKEY 13 0 86400 20201202000000
        20181128000000 47005 .
        0EPW1ca+N/ZhZPKla77STG734cTeIOjUwq7eW0HsnOfudWmnCEVeco2wLLq9m
        nBT1dtNjIczvLG9pQTnOKUsHQ== )
example.com.  3600  IN  DNSKEY  ( 257 3 13
```

```
        JnA1XgyJTZz+psWvbrfUWLV6ULqIJyUS2CQdhUH9VK35bslWeJpRzrlxCUs7s
        /TsSfZMaGWVvlsuieh5nHcXzA== ) ; Key ID = 1870
example.com.  3600  IN  RRSIG  ( DNSKEY 13 2 3600
        20201202000000 20181128000000 1870 example.com.
        nYisnu/26Sw1qmGuREa9o/fLgYuA4oNPt4+6PMBZoN0MS8Gjtli9NVRYeSIzt
        QHPGSpvRxTUC4tZi62z1UgGDw== )
example.com.  172800  IN  DS  ( 1870 13 2 e9b533a049798e900b5c29c90cd
        25a986e8a44f319ac3cd302bafc08f5b81e16 )
example.com.  172800  IN  RRSIG  ( DS 13 2 172800
        20201202000000 20181128000000 34327 com.
        sEAKvX4H6pJfN8nKcclB1NRcRSPOztx8omr4fCSHu6lp+uESP/Le4iF2sKukO
        J1hhWSB6jgubEVl17rGNOA/YQ== )
com.  172800  IN  DNSKEY  ( 256 3 13
        7IIE5Dol8jSMUqHTvOOiZapdEbQ9wqRxFi/zQcSdufUKLhpByvLpzSAQTqCWj
        3URIZ8L3Fa2gBLMOZUzZ1GQCw== ) ; Key ID = 34327
com.  172800  IN  DNSKEY  ( 257 3 13
        RbkcO+96XZmnp8jYIuM4lryAp3egQjSmBaSoiA7H76Tm0RLHPNPUxlVk+nQ0f
        Ic3I8xfZDNw8Wa0Pe3/g2QA/w== ) ; Key ID = 18931
com.  172800  IN  DNSKEY  ( 257 3 13
        szc7biLo5J4OHlkan1vZrF4aD4YYf+NHA/GAqdNslY9xxK9Izg68XHkqck4Rt
        DiVk37lNAQmgSlHbrGu0yOTkA== ) ; Key ID = 28809
com.  172800  IN  RRSIG  ( DNSKEY 13 1 172800 20201202000000
        20181128000000 18931 com.
        LJ4p5ORS2ViILwTotSlWixElqRXHY5tOdIuHlPWTdBGPMq3y40QNr1V+ZOyA5
        7LFdPKpcvb8BvhM+GqKWGBEsg== )
com.  172800  IN  RRSIG  ( DNSKEY 13 1 172800 20201202000000
        20181128000000 28809 com.
        sO+4X2N21yS6x8+dBVBzbRo9+55MM8n7+RUvdBuxRFVh6JaBlqDOC5LLkl7Ev
        mDXqz6KEhhQjT+aQWDt6WFHlA== )
com.  86400  IN  DS  ( 18931 13 2 20f7a9db42d0e2042fbbb9f9ea015941202
        f9eabb94487e658c188e7bcb52115 )
com.  86400  IN  DS  ( 28809 13 2 ad66b3276f796223aa45eda773e92c6d98e
        70643bbde681db342a9e5cf2bb380 )
com.  86400  IN  RRSIG  ( DS 13 1 86400 20201202000000
        20181128000000 31918 .
        nDiDlBjXEE/6AudhC++Hui1ckPcuAnGbjEASNoxA3ZHjlXRzL050UzePko5Pb
        vBKTf6pk8JRCqnfzlo2QY+WXA== )
```

## A.6. `_25._tcp.smtp.example.com` NSEC Denial of Existence

```
smtp.example.com.  3600  IN  NSEC  ( www.example.com. A AAAA
        RRSIG NSEC )
smtp.example.com.  3600  IN  RRSIG  ( NSEC 13 3 3600
        20201202000000 20181128000000 1870 example.com.
        rH/K4wghCOm4jpEHwQKiyZzvFIa7qpFySuKIGGetW4SE4O2Mh5jPxcEzf78Hf
        crlsQZmnAUlfmBNCygxAd7JNw== )
example.com.  3600  IN  DNSKEY  ( 257 3 13
        JnA1XgyJTZz+psWvbrfUWLV6ULqIJyUS2CQdhUH9VK35bslWeJpRzrlxCUs7s
        /TsSfZMaGWVvlsuieh5nHcXzA== ) ; Key ID = 1870
example.com.  3600  IN  RRSIG  ( DNSKEY 13 2 3600
        20201202000000 20181128000000 1870 example.com.
        nYisnu/26Sw1qmGuREa9o/fLgYuA4oNPt4+6PMBZoN0MS8Gjtli9NVRYeSIzt
        QHPGSpvRxTUC4tZi62z1UgGDw== )
example.com.  172800  IN  DS  ( 1870 13 2 e9b533a049798e900b5c29c90cd
        25a986e8a44f319ac3cd302bafc08f5b81e16 )
example.com.  172800  IN  RRSIG  ( DS 13 2 172800
        20201202000000 20181128000000 34327 com.
        sEAKvX4H6pJfN8nKcclB1NRcRSPOztx8omr4fCSHu6lp+uESP/Le4iF2sKukO
        J1hhWSB6jgubEVl17rGNOA/YQ== )
com.  172800  IN  DNSKEY  ( 256 3 13
        7IIE5Dol8jSMUqHTvOOiZapdEbQ9wqRxFi/zQcSdufUKLhpByvLpzSAQTqCWj
        3URIZ8L3Fa2gBLMOZUzZ1GQCw== ) ; Key ID = 34327
com.  172800  IN  DNSKEY  ( 257 3 13
        RbkcO+96XZmnp8jYIuM4lryAp3egQjSmBaSoiA7H76Tm0RLHPNPUxlVk+nQ0f
        Ic3I8xfZDNw8Wa0Pe3/g2QA/w== ) ; Key ID = 18931
com.  172800  IN  DNSKEY  ( 257 3 13
        szc7biLo5J4OHlkan1vZrF4aD4YYf+NHA/GAqdNslY9xxK9Izg68XHkqck4Rt
        DiVk37lNAQmgSlHbrGu0yOTkA== ) ; Key ID = 28809
com.  172800  IN  RRSIG  ( DNSKEY 13 1 172800 20201202000000
        20181128000000 18931 com.
        LJ4p5ORS2ViILwTotSlWixElqRXHY5tOdIuHlPWTdBGPMq3y40QNr1V+ZOyA5
        7LFdPKpcvb8BvhM+GqKWGBEsg== )
com.  172800  IN  RRSIG  ( DNSKEY 13 1 172800 20201202000000
        20181128000000 28809 com.
        sO+4X2N21yS6x8+dBVBzbRo9+55MM8n7+RUvdBuxRFVh6JaBlqDOC5LLkl7Ev
        mDXqz6KEhhQjT+aQWDt6WFHlA== )
com.  86400  IN  DS  ( 18931 13 2 20f7a9db42d0e2042fbbb9f9ea015941202
        f9eabb94487e658c188e7bcb52115 )
com.  86400  IN  DS  ( 28809 13 2 ad66b3276f796223aa45eda773e92c6d98e
        70643bbde681db342a9e5cf2bb380 )
com.  86400  IN  RRSIG  ( DS 13 1 86400 20201202000000
        20181128000000 31918 .
        nDiDlBjXEE/6AudhC++Hui1ckPcuAnGbjEASNoxA3ZHjlXRzL050UzePko5Pb
        vBKTf6pk8JRCqnfzlo2QY+WXA== )
.  86400  IN  DNSKEY  ( 256 3 13
        zKz+DCWkNA/vuheiVPcGqsH40U84KZAlrMRIyozj9WHzf8PsFp/oR8j8vmjjW
        P98cbte4d8NvlGLxzbUzo3+FA== ) ; Key ID = 31918
.  86400  IN  DNSKEY  ( 256 3 13
        8wMZZ4lzHdyKZ4fv8kys/t3QMlgvEadbsbyqWrMhwddSXCZYGRrsAbPpireRW
        xbVcd1VtOrlFBcRDMTN0R0XEQ== ) ; Key ID = 2635
.  86400  IN  DNSKEY  ( 257 3 13
        yvX+VNTUjxZiGvtr060hVbrPV9H6rVusQtF9lIxCFzbZOJxMQBFmbqlc8Xclv
        Q+gDOXnFOTsgs/frMmxyGOtRg== ) ; Key ID = 47005
.  86400  IN  RRSIG  ( DNSKEY 13 0 86400 20201202000000
        20181128000000 47005 .
        0EPW1ca+N/ZhZPKla77STG734cTeIOjUwq7eW0HsnOfudWmnCEVeco2wLLq9m
        nBT1dtNjIczvLG9pQTnOKUsHQ== )
```

### A.7. `_25._tcp.smtp.example.org` NSEC3 Denial of Existence

```
vkv62jbv85822q8rtmfnbhfnmnat9ve3.example.org.  3600  IN  NSEC3  (
        1 0 1 - 93u63bg57ppj6649al2n31l92iedkjd6 A AAAA RRSIG )
vkv62jbv85822q8rtmfnbhfnmnat9ve3.example.org.  3600  IN  RRSIG  (
        NSEC3 13 3 3600 20201202000000 20181128000000 56566
        example.org.
        wn3cePVdc5VPPniYzGp+1CBPOY2m83/A3cjnAb7FTZuwL45B25fwVUyjKQksh
        gQeV5KgP1cdvPt1BEowKqK4Sw== )
dlm7rss9pejqnh0ev6h7k1ikqqcl5mae.example.org.  3600  IN  NSEC3  (
        1 0 1 - t6lf7uuoi0qofq0nvdjroavo46pp20im RRSIG TLSA )
dlm7rss9pejqnh0ev6h7k1ikqqcl5mae.example.org.  3600  IN  RRSIG  (
        NSEC3 13 3 3600 20201202000000 20181128000000 56566
        example.org.
        guUyy9LIZlYb0FZttAdYJGrFNKpKu91Tm+dPOz98rnpwIlwwvLifXIvIl90nE
        X38cWzEQOpreJu3t4WAfPsxdg== )
a73bi8coh6dvf1arqdeuogf95r0828mk.example.org.  3600  IN  NSEC3  (
        1 0 1 - c1p0lp7l1l8gdn0jl13pp1o41h35untj CNAME RRSIG )
a73bi8coh6dvf1arqdeuogf95r0828mk.example.org.  3600  IN  RRSIG  (
        NSEC3 13 3 3600 20201202000000 20181128000000 56566
        example.org.
        ePBUuWdj8Bc+/41gHBm2Bx/IK/j/Q4W7A5uTgSj/0Sd57mP/NTWRZq3p8yBNe
        FPC2mBJ2oWQFi6/V9dmyiBh2A== )
example.org.  3600  IN  DNSKEY  ( 256 3 13
        NrbL6utGqIW1wrhhjeexdA6bMdD1lC1hj0Fnpevaa1AMyY2uy83TmoGnR996N
        UR5TlG4Zh+YPbbmUIixe4nS3w== ) ; Key ID = 56566
example.org.  3600  IN  DNSKEY  ( 257 3 13
        uspaqp17jsMTX6AWVgmbog/3Sttz+9ANFUWLn6qKUHr0BOqRuChQWj8jyYUUr
        Wy9txxesNQ9MkO4LUrFght1LQ== ) ; Key ID = 44384
example.org.  3600  IN  RRSIG  ( DNSKEY 13 2 3600
        20201202000000 20181128000000 44384 example.org.
        ttse9pYp9PSu0pJ+TOpIVFLWJ6NKOMWZX4Q/SlU6ZfaiKQc0Bg7Tut9+wPunk
        6OPPvyHjVXMAsvk0tqV0B+/ag== )
example.org.  86400  IN  DS  ( 44384 13 2 ec307e2efc8f0117ed96ab48a51
        3c8003e1d9121f1ff11a08b4cdd348d090aa6 )
example.org.  86400  IN  RRSIG  ( DS 13 2 86400 20201202000000
        20181128000000 9523 org.
        m86Xz0CEa2sWG40a0bS2kqLKPmIlyiVyDeoWXAq3djeGiPaikLuKORNzWXu62
        clpAfvZHx59Ackst4X+zXYpUA== )
org.  86400  IN  DNSKEY  ( 256 3 13
        fuLp60znhSSEr9HowILpTpyLKQdM6ixcgkTE0gqVdsLx+DSNHSc69o6fLWC0e
        HfWx7kzlBBoJB0vLrvsJtXJ6g== ) ; Key ID = 47417
org.  86400  IN  DNSKEY  ( 256 3 13
        zTHbb7JM627Bjr8CGOySUarsic91xZU3vvLJ5RjVix9YH6+iwpBXb6qfHyQHy
        mlMiAAoaoXh7BUkEBVgDVN8sQ== ) ; Key ID = 9523
org.  86400  IN  DNSKEY  ( 257 3 13
        Uf24EyNt51DMcLV+dHPInhSpmjPnqAQNUTouU+SGLu+lFRRlBetgw1bJUZNI6
        Dlger0VJTm0QuX/JVXcyGVGoQ== ) ; Key ID = 49352
org.  86400  IN  DNSKEY  ( 257 3 13
        0SZfoe8Yx+eoaGgyAGEeJax/ZBV1AuG+/smcOgRm+F6doNlgc3lddcM1MbTvJ
        HTjK6Fvy8W6yZ+cAptn8sQheg== ) ; Key ID = 12651
org.  86400  IN  RRSIG  ( DNSKEY 13 1 86400 20201202000000
        20181128000000 12651 org.
        Gq9wf+z3pasXXUwE210jYc0LhJnMAhcwXydnvkHtCVY6/0jUafHO4RksN84Zt
        us0pUgWngbT/OWXskdMYXZU4A== )
org.  86400  IN  RRSIG  ( DNSKEY 13 1 86400 20201202000000
        20181128000000 49352 org.
        VGEkEMWBJ2IbOpm2Z56Qxu2NGPcVUDWCbYRyk+Qk1+HzGtyd2qPEKkpgMs/0p
        vZEMj1YXD+dIqb2nUK9PGBAXw== )
```

```
org.  86400  IN  DS  ( 12651 13 2 3979a51f98bbf219fcaf4a4176e766dfa8f
        9db5c24a75743eb1e704b97a9fabc )
org.  86400  IN  DS  ( 49352 13 2 03d11a1aa114abbb8f708c3c0ff0db765fe
        f4a2f18920db5f58710dd767c293b )
org.  86400  IN  RRSIG  ( DS 13 1 86400 20201202000000
        20181128000000 31918 .
        adiFuP2UIulQw5Edsb/7WSPqr5nkRSTVXbZ2tkBeZRQcMjdCD3pyonWO5JPRV
        EemgaE357S4pX5D0tVZzeZJ6A== )
.  86400  IN  DNSKEY  ( 256 3 13
        zKz+DCWkNA/vuheiVPcGqsH40U84KZAlrMRIyozj9WHzf8PsFp/oR8j8vmjjW
        P98cbte4d8NvlGLxzbUzo3+FA== ) ; Key ID = 31918
.  86400  IN  DNSKEY  ( 256 3 13
        8wMZZ4lzHdyKZ4fv8kys/t3QMlgvEadbsbyqWrMhwddSXCZYGRrsAbPpireRW
        xbVcd1VtOrlFBcRDMTN0R0XEQ== ) ; Key ID = 2635
.  86400  IN  DNSKEY  ( 257 3 13
        yvX+VNTUjxZiGvtr060hVbrPV9H6rVusQtF9lIxCFzbZOJxMQBFmbqlc8Xclv
        Q+gDOXnFOTsgs/frMmxyGOtRg== ) ; Key ID = 47005
.  86400  IN  RRSIG  ( DNSKEY 13 0 86400 20201202000000
        20181128000000 47005 .
        0EPW1ca+N/ZhZPKla77STG734cTeIOjUwq7eW0HsnOfudWmnCEVeco2wLLq9m
        nBT1dtNjIczvLG9pQTnOKUsHQ== )
```

### A.8. `_443._tcp.www.insecure.example` NSEC3 Opt-Out Insecure Delegation

```
c1kgc91hrn9nqi2qjh1ms78ki8p7s75o.example.  43200  IN  NSEC3  (
        1 1 1 - shn05itmoa45mmnv74lc4p0nnfmimtjt NS SOA RRSIG DNSKEY
        NSEC3PARAM )
c1kgc91hrn9nqi2qjh1ms78ki8p7s75o.example.  43200  IN  RRSIG  (
        NSEC3 13 2 43200 20201202000000 20181128000000 15903
        example.
        pW16gQOLhLpKYgXpGt4XB4o92W/QoPYyG5CjQ+t+g7LBVcCiPQv8ars1j9UOg
        RpXUsJhZBDax2dfDhK7zOk7ow== )
shn05itmoa45mmnv74lc4p0nnfmimtjt.example.  43200  IN  NSEC3  (
        1 1 1 - a3ib1dvf1bdtfmd91usrdem5fiiepi6p NS DS RRSIG )
shn05itmoa45mmnv74lc4p0nnfmimtjt.example.  43200  IN  RRSIG  (
        NSEC3 13 2 43200 20201202000000 20181128000000 15903
        example.
        5Aq//A8bsWNwcXbT91pMX2Oqf8VpJQRjqH4D2yZElW00wKmt85mhgu2qYPrvH
        QwGEB4STMz2Nefq01/GY6NHKg== )
example.  432000  IN  DNSKEY  ( 257 3 13
        yrkqXSbVwXOoUxCjr/E9yg8XUzbZNlwPllVsoUPd73TLOnBQQ+03Qw4/k+Nme
        /66WIw+ZTlHYcTNalxiGYm0uQ== ) ; Key ID = 15903
example.  432000  IN  RRSIG  ( DNSKEY 13 1 432000
        20201202000000 20181128000000 15903 example.
        wwEo3ri6JBuCqx5b33w8axFWOhIen1l+/mm0Isyc9FciuLhBiP+IqSgt+Igc8
        9nR8zRpJpo1D6XR/qJxZgnfaA== )
example.  86400  IN  DS  ( 15903 13 2 7e0ebaf1cc0d309d4a73ca7d711719d
        d940f4da87b3d72865167650fc73ea577 )
example.  86400  IN  RRSIG  ( DS 13 1 86400 20201202000000
        20181128000000 31918 .
        B5vx4zZaS+bOYfz0PzpaPfk9VxxBvYbGjIvGhpUZV3diXzfCguXxN4JIT1Sz8
        eJX6BYT5QPIrbG/N35U1sIskw== )
.  86400  IN  DNSKEY  ( 256 3 13
        zKz+DCWkNA/vuheiVPcGqsH40U84KZAlrMRIyozj9WHzf8PsFp/oR8j8vmjjW
        P98cbte4d8NvlGLxzbUzo3+FA== ) ; Key ID = 31918
.  86400  IN  DNSKEY  ( 256 3 13
        8wMZZ4lzHdyKZ4fv8kys/t3QMlgvEadbsbyqWrMhwddSXCZYGRrsAbPpireRW
        xbVcd1VtOrlFBcRDMTN0R0XEQ== ) ; Key ID = 2635
.  86400  IN  DNSKEY  ( 257 3 13
        yvX+VNTUjxZiGvtr060hVbrPV9H6rVusQtF9lIxCFzbZOJxMQBFmbqlc8Xclv
        Q+gDOXnFOTsgs/frMmxyGOtRg== ) ; Key ID = 47005
.  86400  IN  RRSIG  ( DNSKEY 13 0 86400 20201202000000
        20181128000000 47005 .
        0EPW1ca+N/ZhZPKla77STG734cTeIOjUwq7eW0HsnOfudWmnCEVeco2wLLq9m
        nBT1dtNjIczvLG9pQTnOKUsHQ== )
```

## Acknowledgments

Many thanks to Adam Langley for laying the groundwork for this extension in [SERIALIZECHAIN]. The original idea is his, but our acknowledgment in no way implies his endorsement. This document also benefited from discussions with and review from the following people: Daniel Kahn Gillmor, Jeff Hodges, Allison Mankin, Patrick McManus, Rick van Rein, Ilari Liusvaara, Eric Rescorla, Gowri Visweswaran, Duane Wessels, Nico Williams, and Richard Barnes.

## Authors' Addresses

**Viktor Dukhovni**
Two Sigma
Email: ietf-dane@dukhovni.org

**Shumon Huque**
Salesforce
3rd Floor
415 Mission Street
San Francisco, CA 94105
United States of America
Email: shuque@gmail.com

**Willem Toorop**
NLnet Labs
Science Park 400
1098 XH Amsterdam
Netherlands
Email: willem@nlnetlabs.nl

**Paul Wouters**
Aiven
Toronto
Canada
Email: paul.wouters@aiven.io

**Melinda Shore**
Fastly
Email: mshore@fastly.com