TLS
Internet-Draft
Intended status: Standards Track
Expires: May 2, 2018

M. Shore
Fastly
R. Barnes
Mozilla
S. Huque
Salesforce
W. Toorop
NLnet Labs

October 29, 2017

A DANE Record and DNSSEC Authentication Chain Extension for TLS

draft-ietf-tls-dnssec-chain-extension-05

Abstract

This draft describes a new TLS extension for transport of a DNS record set serialized with the DNSSEC signatures needed to authenticate that record set. The intent of this proposal is to allow TLS clients to perform DANE authentication of a TLS server without needing to perform additional DNS record lookups. It will typically not be used for general DNSSEC validation of TLS endpoint names.

Status of This Memo

This Internet-Draft will expire on May 2, 2018.

Copyright Notice

Table of Contents

1. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Introduction

This draft describes a new TLS [RFC5246] extension for transport of a DNS record set serialized with the DNSSEC signatures [RFC4034] needed to authenticate that record set. The intent of this proposal is to allow TLS clients to perform DANE Authentication [RFC6698] [RFC7671] of a TLS server without performing additional DNS record lookups and incurring the associated latency penalty. It also provides the ability to avoid potential problems with TLS clients being unable to look up DANE records because of an interfering or broken middlebox on the path between the client and a DNS server. And lastly, it allows a TLS client to validate DANE records itself without necessarily needing access to a validating DNS resolver to which it has a secure connection. It will typically not be used for general DNSSEC validation of endpoint names, but is more appropriate for validation of DANE TLSA records.

This mechanism is useful for TLS applications that need to address the problems described above, typically web browsers or VoIP and XMPP applications. It may not be relevant for many other applications. For example, SMTP MTAs are usually located in data centers, may tolerate extra DNS lookup latency, are on servers where it is easier to provision a validating resolver, or are less likely to experience traffic interference from misconfigured middleboxes. Furthermore, SMTP MTAs usually employ Opportunistic Security [RFC7672], in which the presence of the DNS TLSA records is used to determine whether to enforce an authenticated TLS connection. Hence DANE authentication of SMTP MTAs will typically not use this mechanism.

The extension described here allows a TLS client to request in the ClientHello message that the DNS authentication chain be returned in the (extended) ServerHello message. If the server is configured for DANE authentication, then it performs the appropriate DNS queries, builds the authentication chain, and returns it to the client. The server will usually use a previously cached authentication chain, but it will need to rebuild it periodically as described in Section 5.

The client then authenticates the chain using a pre-configured trust anchor.

This specification is based on Adam Langley's original proposal for serializing DNSSEC authentication chains and delivering them in an X.509 certificate extension [I-D.agl-dane-serializechain]. It modifies the approach by using wire format DNS records in the serialized data (assuming that the data will be prepared and consumed by a DNS-specific library), and by using a TLS extension to deliver the data.

As described in the DANE specification [RFC6698] [RFC7671], this procedure applies to the DANE authentication of X.509 certificates or raw public keys [RFC7250].

3. DNSSEC Authentication Chain Extension

3.1. 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 MUST be empty.

Servers receiving a "dnssec_chain" extension in the ClientHello, and which are capable of being authenticated via DANE, MAY return a serialized authentication chain in the extended ServerHello message, using the format described below. If a server is unable to return an authentication chain, or does not wish to return an authentication chain, it does not include a dnssec_chain extension. As with all TLS extensions, if the server does not support this extension it will not return any authentication chain.

A client must not be able to force a server to perform lookups on arbitrary domain names using this mechanism. Therefore, a server MUST NOT construct chains for domain names other than its own.

3.2. Protocol, TLS 1.3

A client MAY include an extension of type "dnssec_chain" in the ClientHello. The "extension_data" field of this extension MUST be empty.

Servers receiving a "dnssec_chain" extension in the ClientHello, and which are capable of being authenticated via DANE, SHOULD return a serialized authentication chain in the extension block of the Certificate message containing the end entity certificate being validated, using the format described below.

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

3.3. Raw Public Keys

[RFC7250] specifies the use of raw public keys for both server and client authentication in TLS 1.2. It points out that in cases where raw public keys are being used, code for certificate path validation is not required. However, DANE, when used in conjunction with the dnssec_chain extension, provides a mechanism for securely binding a raw public key to a named entity in the DNS, and when using DANE for

authentication a raw key may be validated using a path chaining back to a DNSSEC trust root. This has the added benefit of mitigating an unknown key share attack, as described in [I-D.barnes-dane-uks], since it effectively augments the raw public key with the server's name and provides a means to commit both the server and the client to using that binding.

The UKS attack is possible in situations in which the association between a domain name and a public key is not tightly bound, as in the case in DANE in which a client either ignores the name in certificate (as specified in [RFC7671] or there is no attestation of trust outside of the DNS. The vulnerability arises in the following situations:

- If the client does not verify the identity in the server's certificate (as recommended in Section 5.1 of [RFC7671]), then an attacker can induce the client to accept an unintended identity for the server,

- If the client allows the use of raw public keys in TLS, then it will not receive any indication of the server's identity in the TLS channel, and is thus unable to check that the server's identity is as intended.

The mechanism for conveying DNSSEC validation chains described in this document results in a commitment by both parties, via the TLS handshake, to a domain name which has been validated as belonging to the owner name.

The mechanism for encoding DNSSEC authentication chains in a TLS extension, as described in this document, is not limited to public keys encapsulated in X.509 containers but MAY be applied to raw public keys and other representations, as well.

3.4. DNSSEC Authentication Chain Data

The "extension_data" field of the "dnssec_chain" extension MUST contain a DNSSEC Authentication Chain encoded in the following form:

```
          opaque AuthenticationChain<0..2^16-1>
```

The AuthenticationChain structure is composed of a sequence of uncompressed wire format DNS resource record sets (RRset) and corresponding signatures (RRSIG) record sets.

This sequence of native DNS wire format records enables easier generation of the data structure on the server and easier

---

verification of the data on client by means of existing DNS library functions. However this document describes the data structure in sufficient detail that implementers if they desire can write their own code to do this.

Each RRset in the chain is composed of a sequence of wire format DNS resource records. The format of the resource record is described in RFC 1035 [RFC1035], Section 3.2.1.

```
          RR(i) = owner | type | class | TTL | RDATA length | RDATA
```

Each RRset in the sequence is followed by its associated RRsig record set. The RRsig record wire format is described in RFC 4034 [RFC4034], Section 3.1. The signature portion of the RDATA, as described in the same section, is the following:

```
signature = sign(RRSIG_RDATA | RR(1) | RR(2)... )
```

where RRSIG_RDATA is the wire format of the RRSIG RDATA fields with the Signer's Name field in canonical form and the signature field excluded.

The first RRset in the chain MUST contain the TLSA record set being presented. However, if the owner name of the TLSA record set is an alias (CNAME or DNAME), then it MUST be preceded by the chain of alias records needed to resolve it. DNAME chains should omit unsigned CNAME records that may have been synthesized in the response from a DNS resolver.

The subsequent RRsets MUST contain the full set of DNS records needed to authenticate the TLSA record set from the server's trust anchor. Typically this means a set of DNSKEY and DS RRsets that cover all zones from the target zone containing the TLSA record set to the trust anchor zone. The TLS client should be prepared to receive this set of RRsets in any order.

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.

If the TLSA record set was synthesized by a DNS wildcard, the chain must include the signed NSEC or NSEC3 records that prove that there was no explicit match of the TLSA record name and no closer wildcard match.

The final DNSKEY RRset in the authentication chain corresponds to the trust anchor (typically the DNS root). This trust anchor is also preconfigured in the TLS client, but including it in the response from the server permits TLS clients to use the automated trust anchor rollover mechanism defined in RFC 5011 [RFC5011] to update their configured trust anchor.

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)
```

4. 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 extension [RFC6066] when present. If the server_name extension is not present, or if the server does not recognize the provided name and wishes to proceed with the handshake rather than to abort the connection, the server uses the domain name associated with the server IP address to which the connection has been established.

The TLSA record to be queried is constructed by prepending the _port and _transport labels to the domain name as described in [RFC6698], where "port" is the port number associated with the TLS server. The transport is "tcp" for TLS servers, and "udp" for DTLS servers. The

---

port number label is the left-most label, followed by the transport, followed by the base domain name.

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 and DS RRsets and their signatures are added. However, see Section 3.4 for specific processing needed for aliases and wildcards. If DNS responses messages contain any domain names utilizing name compression [RFC1035], then they must be uncompressed.

Newer DNS protocol enhancements, such as the EDNS Chain Query extension [RFC7901] if supported, may offer easier ways to obtain all of the chain data in one transaction with an upstream DNSSEC aware recursive server.

5. 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 MUST refresh and rebuild the chain as TTLs and signature validity periods dictate. A server implementation could carefully track these 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 or signature validity periods of the component records in the chain.

6. Verification

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

Clients MAY cache the server's validated TLS RRset or other validated portions of the chain as an optimization to save signature verification work for future connections. The period of such caching MUST NOT exceed the TTL associated with those records.

7. 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 inband 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.

8. Mandating use of this extension

Green field applications that are designed to always employ this extension, could of course unconditionally mandate its use.

If TLS applications want to mandate the use of this extension for specific servers, clients could maintain a whitelist of sites where the use of this extension is forced. The client would refuse to authenticate such servers if they failed to deliver this extension. Client applications could also employ a Trust on First Use (TOFU) like strategy, whereby they would record the fact that a server offered the extension and use that knowledge to require it for subsequent connections.

This protocol currently provides no way for a server to prove that it doesn't have a TLSA record. Hence absent whitelists, a client misdirected to a server that has fraudulently acquired a public CA issued certificate for the real server's name, could be induced to establish a PKIX verified connection to the rogue server that precluded DANE authentication. This could be solved by enhancing this protocol to require that servers without TLSA records need to provide a DNSSEC authentication chain that proves this (i.e. the chain includes NSEC or NSEC3 records that demonstrate either the absence of the TLSA record, or the absence of a secure delegation to the associated zone). Such an enhancement would be impossible to deploy incrementally though since it requires all TLS servers to support this protocol.

9. 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 5. 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 rollover the trust anchor key, or be able to avail themselves of a service that does this, as described in Section 7.

10. IANA Considerations

This extension requires the registration of a new value in the TLS ExtensionsType registry. The value requested from IANA is 53. If the draft is adopted by the WG, the authors expect to make an early allocation request as specified in [RFC7120].

11. Acknowledgments

Many thanks to Adam Langley for laying the groundwork for this extension. 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: Viktor Dukhovni, Daniel Kahn Gillmor, Jeff Hodges, Allison Mankin, Patrick McManus, Rick van Rein, Ilari Liusvaara, Gowri Visweswaran, Duane Wessels, Nico Williams, and Paul Wouters.

12. References

12.1. 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>.

[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>.

[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>.

12.2. Informative References

[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>.

[RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, DOI 10.17487/RFC7120, January 2014, <https://www.rfc-editor.org/info/rfc7120>.

[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>.

[RFC7672] Dukhovni, V. and W. Hardaker, "SMTP Security via Opportunistic DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS)", RFC 7672, DOI 10.17487/RFC7672, October 2015, <https://www.rfc-editor.org/info/rfc7672>.

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

[I-D.agl-dane-serializechain]
Langley, A., "Serializing DNS Records with DNSSEC Authentication", draft-agl-dane-serializechain-01 (work in progress), July 2011.

[I-D.barnes-dane-uks]
Barnes, R., Thomson, M., and E. Rescorla, "Unknown Key-Share Attacks on DNS-based Authentications of Named Entities (DANE)", draft-barnes-dane-uks-00 (work in

progress), October 2016.

Appendix A. Updates from -01 and -02

- Editorial updates for style and consistency
- Updated discussion of UKS attack

Appendix B. Updates from -01

- Added TLS 1.3 support
- Added section describing applicability to raw public keys
- Softened language about record order

Appendix C. Updates from -00

- Edits based on comments from Rick van Rein

- Warning about not overloading X.509 wildcards on DNSSEC wildcards (from V. Dukhovny)

- Added MUST include negative proof on wildcards (from V. Dukhovny)

- Removed "TODO" on allowing the server to deliver only one signature per RRset

- Added additional minor edits suggested by Viktor Dukhovny

Appendix D. Test vectors

The provided 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-----
MIIF8jCCBNqgAwIBAgIQDmTF+8I2reFLFyrrQceMsDANBgkqhkiG9w0BAQsFADBw
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
d3cuZGlnaWNlcnQuY29tMS8wLQYDVQQDEyZEaWdpQ2VydCBTSEEyIEhpZ2ggQXNz
dXJhbmNlIFNlcnZlciBDQTAeFw0xNTExMDMwMDAwMDBaFw0xODExMjgxMjAwMDBa
MIGlMQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTEUMBIGA1UEBxML
TG9zIEFuZ2VsZXMxPDA6BgNVBAoTM0ludGVybmV0IENvcnBvcmF0aW9uIGZvciBB
c3NpZ25lZCBOYW1lcyBhbmQgTnVtYmVyczETMBEGA1UECxMKVGVjaG5vbG9neTEY
MBYGA1UEAxMPd3d3LmV4YW1wbGUub3JnMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8A
MIIBCgKCAQEAs0CWL2FjPiXBl61lRfvvE0KzLJmG9LWAC3bcBjgsH6NiVVo2dt6u
Xfzi5bTm7F3K7srfUBYkLO78mraM9qizrHoIeyofrV/n+pZZJauQsPjCPxMEJnRo
```

```
D8Z4KpWKX0LyDu1SputoI4nlQ/htEhtiQnuoBfNZxF7WxcxGwEsZuS1KcXIkHl5V
RJOreKFHTaXcB1qcZ/QRaBIv0yhxvK1yBTwWddT4cli6GfHcCe3xGMaSL328Fgs3
jYrvG29PueB6VJi/tbbPu6qTfwp/H1brqdjh29U52Bhb0fJkM9DWxCP/Cattcc7a
z8EXnCO+LK8vkhw/kAiJWPKx4RBvgy73nwIDAQABo4ICUDCCAkwwHwYDVR0jBBgw
FoAUUWj/kK8CB3U8zNllZGKiErhZcjswHQYDVR0OBBYEFKZPYB4fLdHn8SOgKpUW
5Oia6m5IMIGBBgNVHREEejB4gg93d3cuZXhhbXBsZS5vcmeCC2V4YW1wbGUuY29t
ggtleGFtcGxlLmVkdYILZXhhbXBsZS5uZXSCC2V4YW1wbGUub3Jngg93d3cuZXhh
bXBsZS5jb22CD3d3dy5leGFtcGxlLmVkdYIPd3d3LmV4YW1wbGUubmV0MA4GA1Ud
DwEB/wQEAwIFoDAdBgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUHAwIwdQYDVR0f
BG4wbDA0oDKgMIYuaHR0cDovL2NybDMuZGlnaWNlcnQuY29tL3NoYTItaGEtc2Vy
dmVyLWc0LmNybDA0oDKgMIYuaHR0cDovL2NybDQuZGlnaWNlcnQuY29tL3NoYTIt
aGEtc2VydmVyLWc0LmNybDBMBgNVHSAERTBDMDcGCWCGSAGG/WwBATAqMCgGCCsG
AQUFBwIBFhxodHRwczovL3d3dy5kaWdpY2VydC5jb20vQ1BTMAgGBmeBDAECAjCB
gwYIKwYBBQUHAQEEdzB1MCQGCCsGAQUFBzABhhhodHRwOi8vb2NzcC5kaWdpY2Vy
dC5jb20wTQYIKwYBBQUHMAKGQWh0dHA6Ly9jYWNlcnRzLmRpZ2ljZXJ0LmNvbS9E
aWdpQ2VydFNIQTJIaWdoQXNzdXJhbmNlU2VydmVyQ0EuY3J0MAwGA1UdEwEB/wQC
MAAwDQYJKoZIhvcNAQELBQADggEBAISomhGn2L0LJn5SJHuyVZ3qMIlRCIdvqe0Q
6ls+C8ctRwRO3UU3x8q8OH+2ahxlQmpzdC5al4XQzJLiLjiJ2Q1p+hub8MFiMmVP
PZjb2tZm2ipWVuMRM+zgpRVM6nVJ9F3vFfUSHOb4/JsEIUvPY+d8/Krc+kPQwLvy
ieqRbcuFjmqfyPmUv1U9QoI4TQikpw7TZU0zYZANP4C/gj4Ry48/znmUaRvy2kvI
l7gRQ21qJTK5suoiYoYNo3J9T+pXPGU7Lydz/HwW+w0DpArtAaukI8aNX4ohFUKS
wDSiIIWIWJiJGbEeIO0TIFwEVWTOnbNl/faPXpk5IRXicapqiII=
-----END CERTIFICATE-----
```

For brevity and reproducability all DNS zones involved with the test vectors are signed using a single key with algorithm 13: ECDSA Curve P-256 with SHA-256.

The test vectors are DNSSEC valid at June 1 2017, with the following root trust anchor:

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

D.1. _443._tcp.www.example.com

```
_443._tcp.www.example.com.  3600  IN  TLSA  ( 3 1 1
                c66bef6a5c1a3e78b82016e13f314f3cc5fa25b1e52aab9adb9ec5989b165
```

```
           ada )
   _443._tcp.www.example.com.  3600  IN  RRSIG  ( TLSA 13 5 3600
           20170616000000 20170526000000 1870 example.com.
           GRsT6bcn3fokM5JMvHF0liq63N/kUX+CrZQZIr4GlFnMr/uoS4P1zOBwc0sft
           Kd8NsZJAikRr4CpaXITYQMx1w== )
   example.com.  3600  IN  DNSKEY  ( 257 3 13
           JnA1XgyJTZz+psWvbrfUWLV6ULqIJyUS2CQdhUH9VK35bslWeJpRzrlxCUs7s
           /TsSfZMaGWVvlsuieh5nHcXzA== ) ; Key ID = 1870
   example.com.  3600  IN  RRSIG  ( DNSKEY 13 2 3600
           20170616000000 20170526000000 1870 example.com.
           sB6o0XXz7AXDWibruD75rllaHI1kOu4ftoXsKOPPArjflNlTPxrJsspN8ww9r
           8q6DBlCdlRQvzD90UKZDIAqbA== )
   example.com.  900  IN  DS  ( 1870 13 2
           e9b533a049798e900b5c29c90cd25a986e8a44f319ac3cd302bafc08f5b81
           e16 )
   example.com.  900  IN  RRSIG  ( DS 13 2 900 20170605000000
           20170529000000 18931 com.
           rBV/16HTJBwmexByZq7WzYbB3EYaJ6MctpUSxuSNEpwDgzKkwIXzKECh5F5x3
           5XxvbOdLIJAcxhRS1c2VITfMA== )
   com.  900  IN  DNSKEY  ( 257 3 13
           RbkcO+96XZmnp8jYIuM4lryAp3egQjSmBaSoiA7H76Tm0RLHPNPUxlVk+nQ0f
           Ic3I8xfZDNw8Wa0Pe3/g2QA/w== ) ; Key ID = 18931
   com.  900  IN  RRSIG  ( DNSKEY 13 1 900 20170605000000
           20170529000000 18931 com.
           wjCqnHNa5QcMrbuAnKIWBESMFtVjDldmd98udKPtg35V9ESD9SuNKtRJRdHYk
           c6Nx3HLmhidf6dmt/Hi0ePBsQ== )
   com.  86400  IN  DS  ( 18931 13 2
           20f7a9db42d0e2042fbbb9f9ea015941202f9eabb94487e658c188e7bcb52
           115 )
   com.  86400  IN  RRSIG  ( DS 13 1 86400 20170612000000
           20170530000000 47005 .
           jPah4caFBSqhdt78YYhwFZn3ouKiWUKTH1t/nMB7tXzjorQJ50j1RMR23JVL+
           jGGQccnLkQnUf7zednetSWalg== )
   .  86400  IN  DNSKEY  ( 257 3 13
           yvX+VNTUjxZiGvtr060hVbrPV9H6rVusQtF9lIxCFzbZOJxMQBFmbqlc8Xclv
           Q+gDOXnFOTsgs/frMmxyGOtRg== ) ; Key ID = 47005
   .  86400  IN  RRSIG  ( DNSKEY 13 0 86400 20170612000000
           20170530000000 47005 .
           tFldEx7SQI43PIzn1ib/oZTko+Q+gRuOLcALoSA0WQRh1yXSV1752p1n3imhK
           8y3m+LZSLDSBHEocXIiRHWdFg== )
```

A hex dump of the wire format data of this content is:

```
   0000:  04 5f 34 34 33 04 5f 74  63 70 03 77 77 77 07 65
   0010:  78 61 6d 70 6c 65 03 63  6f 6d 00 00 34 00 01 00
   0020:  00 0e 10 00 23 03 01 01  c6 6b ef 6a 5c 1a 3e 78
   0030:  b8 20 16 e1 3f 31 4f 3c  c5 fa 25 b1 e5 2a ab 9a
   0040:  db 9e c5 98 9b 16 5a da  04 5f 34 34 33 04 5f 74
   0050:  63 70 03 77 77 77 07 65  78 61 6d 70 6c 65 03 63
   0060:  6f 6d 00 00 2e 00 01 00  00 0e 10 00 5f 00 34 0d
   0070:  05 00 00 0e 10 59 43 1f  80 59 27 70 00 07 4e 07
   0080:  65 78 61 6d 70 6c 65 03  63 6f 6d 00 7b be 85 af
   0090:  63 08 fd be 6e eb 68 df  85 c2 58 e6 41 37 2f 68
   00a0:  34 4f 4f ce 91 9c 4c b0  54 bb e5 31 cd 57 0c 57
   00b0:  cf 10 ce 33 13 29 7a b4  81 d9 10 d0 cf f3 32 c8
   00c0:  24 e8 06 12 59 8c 58 c5  15 6e ae e1 07 65 78 61
   00d0:  6d 70 6c 65 03 63 6f 6d  00 00 30 00 01 00 00 0e
   00e0:  10 00 44 01 01 03 0d 26  70 35 5e 0c 89 4d 9c fe
   00f0:  a6 c5 af 6e b7 d4 58 b5  7a 50 ba 88 27 25 12 d8
   0100:  24 1d 85 41 fd 54 ad f9  6e c9 56 78 9a 51 ce b9
   0110:  71 09 4b 3b b3 f4 ec 49  f6 4c 68 65 95 be 5b 2e
   0120:  89 e8 79 9c 77 17 cc 07  65 78 61 6d 70 6c 65 03
   0130:  63 6f 6d 00 00 2e 00 01  00 00 0e 10 00 5f 00 30
   0140:  0d 02 00 00 0e 10 59 43  1f 80 59 27 70 00 07 4e
   0150:  07 65 78 61 6d 70 6c 65  03 63 6f 6d 00 db ce bb
   0160:  5f 1c 4b f0 4e de 1e 36  f0 00 75 ae 79 f1 4e 7b
   0170:  42 3b ff dc c0 04 b8 3c  5f 3a e7 ac a7 0c 47 0a
   0180:  a5 3d 70 95 28 d5 c9 65  5c 6f 7c ad 25 1e d2 77
   0190:  00 2c 0a 9f f7 e9 19 a6  04 e9 cb 09 60 07 65 78
   01a0:  61 6d 70 6c 65 03 63 6f  6d 00 00 2b 00 01 00 00
   01b0:  03 84 00 24 07 4e 0d 02  e9 b5 33 a0 49 79 8e 90
   01c0:  0b 5c 29 c9 0c d2 5a 98  6e 8a 44 f3 19 ac 3c d3
   01d0:  02 ba fc 08 f5 b8 1e 16  07 65 78 61 6d 70 6c 65
   01e0:  03 63 6f 6d 00 00 2e 00  01 00 00 03 84 00 57 00
   01f0:  2b 0d 02 00 00 03 84 59  34 9f 00 59 2b 64 80 49
   0200:  f3 03 63 6f 6d 00 18 f3  6d 66 92 89 48 73 26 3a
   0210:  cd 1e 35 38 a3 97 07 1b  ed de d6 14 db 16 f0 f5
   0220:  62 27 20 c5 eb fa 66 ac  a4 a7 8e 93 33 ca 62 42
   0230:  91 7a 51 b5 15 3a 83 14  3a 80 a5 f2 b6 80 00 e5
   0240:  6b 92 ba 37 ec 2d 03 63  6f 6d 00 00 30 00 01 00
   0250:  00 03 84 00 44 01 01 03  0d 45 b9 1c 3b ef 7a 5d
   0260:  99 a7 a7 c8 d8 22 e3 38  96 bc 80 a7 77 a0 42 34
   0270:  a6 05 a4 a8 88 0e c7 ef  a4 e6 d1 12 c7 3c d3 d4
   0280:  c6 55 64 fa 74 34 7c 87  37 23 cc 5f 64 33 70 f1
   0290:  66 b4 3d ed ff 83 64 00  ff 03 63 6f 6d 00 00 2e
   02a0:  00 01 00 00 03 84 00 57  00 30 0d 01 00 00 03 84
   02b0:  59 34 9f 00 59 2b 64 80  49 f3 03 63 6f 6d 00 8d
   02c0:  21 46 95 a5 17 ab 10 0a  49 dd 25 d3 6b 7d 88 ab
   02d0:  2b 18 c9 53 da f2 76 fd  a5 82 b8 ea 14 cb 7c 25
   02e0:  4a 36 4a f0 48 9b e6 a3  0d aa 5b 98 15 8e 64 12
```

```
   02f0:  bb 1b 6e 45 3f 03 00 88  3d 48 b7 0f 78 53 2b 03
```

```
   0300:  63 6f 6d 00 00 2b 00 01  00 01 51 80 00 24 49 f3
   0310:  0d 02 20 f7 a9 db 42 d0  e2 04 2f bb b9 f9 ea 01
   0320:  59 41 20 2f 9e ab b9 44  87 e6 58 c1 88 e7 bc b5
   0330:  21 15 03 63 6f 6d 00 00  2e 00 01 00 01 51 80 00
   0340:  53 00 2b 0d 01 00 01 51  80 59 3d d9 80 59 2c b6
   0350:  00 b7 9d 00 33 56 6b d8  e2 80 50 7a e6 cf 68 27
   0360:  bb 22 5c a7 aa 41 f1 36  94 1c ae 94 9c 3f da 98
   0370:  c5 0f 56 b8 26 c7 57 44  05 a3 a5 11 ef d9 77 b3
   0380:  49 a9 50 8d 99 76 98 78  8e 4b 30 a8 70 51 63 09
   0390:  a2 b6 14 05 00 00 30 00  01 00 01 51 80 00 44 01
   03a0:  01 03 0d ca f5 fe 54 d4  d4 8f 16 62 1a fb 6b d3
   03b0:  ad 21 55 ba cf 57 d1 fa  ad 5b ac 42 d1 7d 94 8c
   03c0:  42 17 36 d9 38 9c 4c 40  11 66 6e a9 5c f1 77 25
   03d0:  bd 0f a0 0c e5 e7 14 e4  ec 82 cf df ac c9 b1 c8
   03e0:  63 ad 46 00 00 2e 00 01  00 01 51 80 00 53 00 30
   03f0:  0d 00 00 01 51 80 59 3d  d9 80 59 2c b6 00 b7 9d
   0400:  00 2b 43 e5 99 de 8d bd  e6 e1 f0 05 2d 16 a1 7a
   0410:  79 15 42 da 47 da 2f 63  0e 46 ab 7d e3 7e 9b 8a
   0420:  7d 25 e2 3f 18 bf 85 4c  17 b7 d5 3c 06 c8 18 bb
   0430:  bd 98 44 11 72 e3 18 bc  9d 99 88 e5 00 91 58 c8
   0440:  47
```

D.2. _25._tcp.example.com wildcard

```
   _25._tcp.example.com.  3600  IN  TLSA  ( 3 1 1
           c66bef6a5c1a3e78b82016e13f314f3cc5fa25b1e52aab9adb9ec5989b165
           ada )
   _25._tcp.example.com.  3600  IN  RRSIG  ( TLSA 13 3 3600
           20170616000000 20170526000000 1870 example.com.
           dVxm88Spko03MB4pLo+zijMup2nr1Ii65yPqB/cAR/1Zg41iXer7I2sGh9KfT
           ExLJC6dUMDVFUfm+1rwb+ax8Q== )
   *._tcp.example.com.  3600  IN  NSEC  (
           _443._tcp.www.example.com. RRSIG NSEC TLSA )
   *._tcp.example.com.  3600  IN  RRSIG  ( NSEC 13 3 3600
           20170616000000 20170526000000 1870 example.com.
           1lNaYYQ+FAG8YBVEx/026OGhVw5DjAyqBGrrLN9D12IZuLHtTQ4C9QPORP4na
           GWNPgASvLyNR8MoN0oXV7tbnQ== )
   example.com.  3600  IN  DNSKEY  ( 257 3 13
           JnA1XgyJTZz+psWvbrfUWLV6ULqIJyUS2CQdhUH9VK35bslWeJpRzrlxCUs7s
           /TsSfZMaGWVvlsuieh5nHcXzA== ) ; Key ID = 1870
   example.com.  3600  IN  RRSIG  ( DNSKEY 13 2 3600
           20170616000000 20170526000000 1870 example.com.
           sB6o0XXz7AXDWibruD75rllaHI1kOu4ftoXsKOPPArjflNlTPxrJsspN8ww9r
           8q6DBlCdlRQvzD90UKZDIAqbA== )
   example.com.  900  IN  DS  ( 1870 13 2
           e9b533a049798e900b5c29c90cd25a986e8a44f319ac3cd302bafc08f5b81
           e16 )
   example.com.  900  IN  RRSIG  ( DS 13 2 900 20170605000000
           20170529000000 18931 com.
           rBV/16HTJBwmexByZq7WzYbB3EYaJ6MctpUSxuSNEpwDgzKkwIXzKECh5F5x3
           5XxvbOdLIJAcxhRS1c2VITfMA== )
   com.  900  IN  DNSKEY  ( 257 3 13
           RbkcO+96XZmnp8jYIuM4lryAp3egQjSmBaSoiA7H76Tm0RLHPNPUxlVk+nQ0f
           Ic3I8xfZDNw8Wa0Pe3/g2QA/w== ) ; Key ID = 18931
   com.  900  IN  RRSIG  ( DNSKEY 13 1 900 20170605000000
           20170529000000 18931 com.
           wjCqnHNa5QcMrbuAnKIWBESMFtVjDldmd98udKPtg35V9ESD9SuNKtRJRdHYk
```

```
           c6Nx3HLmhidf6dmt/Hi0ePBsQ== )
   com.  86400  IN  DS  ( 18931 13 2
           20f7a9db42d0e2042fbbb9f9ea015941202f9eabb94487e658c188e7bcb52
           115 )
   com.  86400  IN  RRSIG  ( DS 13 1 86400 20170612000000
           20170530000000 47005 .
           jPah4caFBSqhdt78YYhwFZn3ouKiWUKTH1t/nMB7tXzjorQJ50j1RMR23JVL+
           jGGQccnLkQnUf7zednetSWalg== )
   .  86400  IN  DNSKEY  ( 257 3 13
           yvX+VNTUjxZiGvtr060hVbrPV9H6rVusQtF9lIxCFzbZOJxMQBFmbqlc8Xclv
           Q+gDOXnFOTsgs/frMmxyGOtRg== ) ; Key ID = 47005
   .  86400  IN  RRSIG  ( DNSKEY 13 0 86400 20170612000000
           20170530000000 47005 .
           tFldEx7SQI43PIzn1ib/oZTko+Q+gRuOLcALoSA0WQRh1yXSV1752p1n3imhK
           8y3m+LZSLDSBHEocXIiRHWdFg== )
```

D.3. _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
           20170616000000 20170526000000 44384 example.org.
           DN+UMxh5TWL1u6Mc6ScWMU5R9C+qqIOSH4hqQmiPWhvSg0lFd71g43UqtdmBT
           VRUbhk/f9WC8Fy5D0gE5lUcyA== )
   dane311.example.org.  3600  IN  TLSA  ( 3 1 1
           c66bef6a5c1a3e78b82016e13f314f3cc5fa25b1e52aab9adb9ec5989b165
           ada )
   dane311.example.org.  3600  IN  RRSIG  ( TLSA 13 3 3600
           20170616000000 20170526000000 44384 example.org.
           HJx59dAMQgvJSYBYtixzfodup5KRUzJ1SlRUrFJkGZz6PkpfuFdtpZwPN1vw9
           SyvXq7WqRD46aaCMwR4ApUJ+w== )
   example.org.  3600  IN  DNSKEY  ( 257 3 13
           uspaqp17jsMTX6AWVgmbog/3Sttz+9ANFUWLn6qKUHr0BOqRuChQWj8jyYUUr
           Wy9txxesNQ9MkO4LUrFght1LQ== ) ; Key ID = 44384
   example.org.  3600  IN  RRSIG  ( DNSKEY 13 2 3600
           20170616000000 20170526000000 44384 example.org.
           MPTpfbVvPBXmh2Z4fgjy2GMgcJ8RYpXeOBOBidJDglLh4XQAiFOT6YpGRR5ig
```

```
           tQGINd6gKVYdRSsEtXe1K8zxg== )
   example.org.  900  IN  DS  ( 44384 13 2
           ec307e2efc8f0117ed96ab48a513c8003e1d9121f1ff11a08b4cdd348d090
           aa6 )
   example.org.  900  IN  RRSIG  ( DS 13 2 900 20170615000000
           20170525000000 12651 org.
           MA3pxiap702Hvc81diwZDcRzLrkKssVzzTqCiJJoZFeNq40GuCOVGgEc+w6aq
           SVgkgFJrhJISei/kvIZTx8ftw== )
   org.  900  IN  DNSKEY  ( 257 3 13
           0SZfoe8Yx+eoaGgyAGEeJax/ZBV1AuG+/smcOgRm+F6doNlgc3lddcM1MbTvJ
           HTjK6Fvy8W6yZ+cAptn8sQheg== ) ; Key ID = 12651
   org.  900  IN  RRSIG  ( DNSKEY 13 1 900 20170615000000
           20170525000000 12651 org.
           o4L9nBQo8KXF0a7D5268U+Bq8SuW/aePtyuSfvQvP79nN/mzh9P11CsT/opmW
           kg0u6pqaG9KE1T37jloes8J8w== )
   org.  86400  IN  DS  ( 12651 13 2
           3979a51f98bbf219fcaf4a4176e766dfa8f9db5c24a75743eb1e704b97a9f
           abc )
   org.  86400  IN  RRSIG  ( DS 13 1 86400 20170612000000
           20170530000000 47005 .
           Mi1c7QrpHnl1MSLJTrq/WM0V0DQKwFPGaMFmHHwm8Yb/b934CUHMXU0dR+cLT
           hakZNz37edtwPxKKOzZQ6pYUw== )
   .  86400  IN  DNSKEY  ( 257 3 13
           yvX+VNTUjxZiGvtr060hVbrPV9H6rVusQtF9lIxCFzbZOJxMQBFmbqlc8Xclv
           Q+gDOXnFOTsgs/frMmxyGOtRg== ) ; Key ID = 47005
   .  86400  IN  RRSIG  ( DNSKEY 13 0 86400 20170612000000
           20170530000000 47005 .
           tFldEx7SQI43PIzn1ib/oZTko+Q+gRuOLcALoSA0WQRh1yXSV1752p1n3imhK
           8y3m+LZSLDSBHEocXIiRHWdFg== )
```

D.4. _443._tcp.www.example.net DNAME

```
   example.net.  3600  IN  DNAME  example.com.
   example.net.  3600  IN  RRSIG  ( DNAME 13 2 3600 20170616000000
           20170526000000 48085 example.net.
           sTl9oxvpd7KxOZ9e5suevha7Fr+zPc3a0pWEUfjFE5v9Umu5js/vcp1i6hdqy
           tQ2WXEQDsHeEjw9stupvMJkkg== )
   _443._tcp.www.example.net.  3600  IN  CNAME  (
           _443._tcp.www.example.com. )
   _443._tcp.www.example.com.  3600  IN  TLSA  ( 3 1 1
           c66bef6a5c1a3e78b82016e13f314f3cc5fa25b1e52aab9adb9ec5989b165
           ada )
```

```
   _443._tcp.www.example.com.  3600  IN  RRSIG  ( TLSA 13 5 3600
           20170616000000 20170526000000 1870 example.com.
           GRsT6bcn3fokM5JMvHF0liq63N/kUX+CrZQZIr4GlFnMr/uoS4P1zOBwc0sft
           Kd8NsZJAikRr4CpaXITYQMx1w== )
   example.net.  3600  IN  DNSKEY  ( 257 3 13
           X9GHpJcS7bqKVEsLiVAbddHUHTZqqBbVa3mzIQmdp+5cTJk7qDazwH68Kts8d
           9MvN55HddWgsmeRhgzePz6hMg== ) ; Key ID = 48085
   example.net.  3600  IN  RRSIG  ( DNSKEY 13 2 3600
           20170616000000 20170526000000 48085 example.net.
           8jSs5O3AypurKs8JFoAYj30qlmQ9QS29IBoqbyv2ggxtdDZoKWZE0kOuQcRxx
           q1pP707qRjp98THQSTVh+C0iQ== )
   example.net.  900  IN  DS  ( 48085 13 2
           7c1998ce683df60e2fa41460c453f88f463dac8cd5d074277b4a7c0450292
           1be )
   example.net.  900  IN  RRSIG  ( DS 13 2 900 20170615000000
           20170525000000 485 net.
           xqN9gHO5HXB+GH2x3DvjpMl6f+CdsVvON2K7G0FDVIL5iFMNLPqCAITlFofWF
           Ty6VXFKPoy9TZresE/JCL/PFA== )
   net.  900  IN  DNSKEY  ( 257 3 13
           LkNCPE+v3S4MVnsOqZFhn8n2NSwtLYOZLZjjgVsAKgu4XZncaDgq1R/7ZXRO5
           oVx2zthxuu2i+mGbRrycAaCvA== ) ; Key ID = 485
   net.  900  IN  RRSIG  ( DNSKEY 13 1 900 20170615000000
           20170525000000 485 net.
           jEI8WucG9EzJ1Euv0Pq3aNFhoYbvQeLUs19r9YImkWi8QlmH76ZJuLTCGHTDh
           /Il5cZWukKc3ScptxVA57uRyQ== )
   net.  86400  IN  DS  ( 485 13 2
           ab25a2941aa7f1eb8688bb783b25587515a0cd8c247769b23adb13ca234d1
           c05 )
   net.  86400  IN  RRSIG  ( DS 13 1 86400 20170612000000
           20170530000000 47005 .
           ZR/UTP2OrYwJQhsCAWsKoIs9OSiUDdBFXzFqYSrV41G1oQsKVSi/NU1tT1UZW
           CENddWt90XLXZAlSYnv6s8Ceg== )
   .  86400  IN  DNSKEY  ( 257 3 13
           yvX+VNTUjxZiGvtr060hVbrPV9H6rVusQtF9lIxCFzbZOJxMQBFmbqlc8Xclv
           Q+gDOXnFOTsgs/frMmxyGOtRg== ) ; Key ID = 47005
   .  86400  IN  RRSIG  ( DNSKEY 13 0 86400 20170612000000
```

```
           20170530000000 47005 .
           tFldEx7SQI43PIzn1ib/oZTko+Q+gRuOLcALoSA0WQRh1yXSV1752p1n3imhK
           8y3m+LZSLDSBHEocXIiRHWdFg== )
   example.com.  3600  IN  DNSKEY  ( 257 3 13
           JnA1XgyJTZz+psWvbrfUWLV6ULqIJyUS2CQdhUH9VK35bslWeJpRzrlxCUs7s
```

```
           /TsSfZMaGWVvlsuieh5nHcXzA== ) ; Key ID = 1870
   example.com.  3600  IN  RRSIG  ( DNSKEY 13 2 3600
           20170616000000 20170526000000 1870 example.com.
           sB6o0XXz7AXDWibruD75rllaHI1kOu4ftoXsKOPPArjflNlTPxrJsspN8ww9r
           8q6DBlCdlRQvzD90UKZDIAqbA== )
   example.com.  900  IN  DS  ( 1870 13 2
           e9b533a049798e900b5c29c90cd25a986e8a44f319ac3cd302bafc08f5b81
           e16 )
   example.com.  900  IN  RRSIG  ( DS 13 2 900 20170605000000
           20170529000000 18931 com.
           rBV/16HTJBwmexByZq7WzYbB3EYaJ6MctpUSxuSNEpwDgzKkwIXzKECh5F5x3
           5XxvbOdLIJAcxhRS1c2VITfMA== )
   com.  900  IN  DNSKEY  ( 257 3 13
           RbkcO+96XZmnp8jYIuM4lryAp3egQjSmBaSoiA7H76Tm0RLHPNPUxlVk+nQ0f
           Ic3I8xfZDNw8Wa0Pe3/g2QA/w== ) ; Key ID = 18931
   com.  900  IN  RRSIG  ( DNSKEY 13 1 900 20170605000000
           20170529000000 18931 com.
           wjCqnHNa5QcMrbuAnKIWBESMFtVjDldmd98udKPtg35V9ESD9SuNKtRJRdHYk
           c6Nx3HLmhidf6dmt/Hi0ePBsQ== )
   com.  86400  IN  DS  ( 18931 13 2
           20f7a9db42d0e2042fbbb9f9ea015941202f9eabb94487e658c188e7bcb52
           115 )
   com.  86400  IN  RRSIG  ( DS 13 1 86400 20170612000000
           20170530000000 47005 .
           jPah4caFBSqhdt78YYhwFZn3ouKiWUKTH1t/nMB7tXzjorQJ50j1RMR23JVL+
           jGGQccnLkQnUf7zednetSWalg== )
```

Authors' Addresses

Melinda Shore
Fastly

EMail: mshore@fastly.com

Richard Barnes
Mozilla

EMail: rlb@ipv.sx

Shumon Huque
Salesforce

EMail: shuque@gmail.com

Willem Toorop
NLnet Labs

EMail: willem@nlnetlabs.nl