Internet Engineering Task Force
Internet-Draft
Updates: 6698 (if approved)
Intended status: Standards Track
Expires: January 06, 2016

S. Huque
Verisign Labs
D. James
Verisign, Inc.
V. Dukhovni
Two Sigma
July 05, 2015

Client Certificates in DANE TLSA Records
draft-huque-dane-client-cert-01

Abstract

The current DNS TLSA record format [RFC6698] describes how to specify TLS server certificates or their public keys in the DNS. This document makes a narrowly focused update to RFC 6698. It describes how to additionally use the TLSA record to specify client certificates, and also the rules and considerations for using them with the TLS protocol.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 06, 2016.

Table of Contents

1. Introduction and Motivation

The Transport Layer Security (TLS) protocol [RFC5246] optionally supports the authentication of clients using X.509 certificates [RFC5280]. TLS Applications currently employing DANE authentication of servers using TLSA records may also desire to authenticate clients using the same mechanism, especially if the client identity is in the form of or can be represented by a DNS domain name. Some design patterns from the Internet of Things (IoT) make use of this form of authentication, where large networks of physical objects identified by DNS names may authenticate themselves using TLS to centralized device management and control platforms.

In this document, the term TLS is used generically to describe both the TLS and DTLS (Datagram Transport Layer Security) [RFC6347] protocols.

2. Associating Client Identities in TLSA Records

When specifying client identities (i.e. client domain names) in TLSA records, the owner name of the TLSA record has the following format:

_service.[client-domain-name]

The first label identifies the application service name. The remaining labels are composed of the client domain name.

Encoding the application service name into the owner name allows the same client domain name to have different authentication credentials for different application services. There is no need to encode the transport label - the same name form is usable with both TLS and DTLS.

The _service label could be a custom string for an application, but more commonly is expected to be a service name registered in the IANA Service Name Registry [SRVREG].

The RDATA or data field portion of the TLSA record is formed exactly as specified in RFC 6698, and carries the same meaning.

3. Authentication Model

The authentication model assumed in this document is the following:

The client is assigned an identity corresponding to a DNS domain name. This domain name doesn’t necessarily have any relation to its network layer addresses. Clients often have dynamic or unpredictable addresses, and may move around the network, so tying their identity to network addresses is not feasible or wise in the general case.

The client generates (or has generated for it) a private and public key pair, and a certificate binding the name to its public key. This certificate has a corresponding TLSA record published in the DNS, which allows it to be authenticated directly via the DNS (using the DANE-TA or DANE-EE usage modes) or via a PKIX public CA system constraint (using the PKIX-TA or PKIX-EE usage modes).

4. Client Identifiers in X.509 certificates

The client certificate MUST have have the client’s DNS name specified in the Subject Alternative Name extension’s dNSName type. Or, if an application specific identity is preferred or needed, the SRV-ID (PKIX OtherName SRVName) MUST be used to specify the application service and the client’s name, e.g. "_smtp-client.device1.example.com". See [RFC6125] and [RFC4985] for a discussion of application specific identifiers in X.509 certificates.

The initial revision of this document talks mainly about dNSName identifiers, because SRV-ID has not seen much adoption in the

Internet to date. However, with TLSA usage modes except for DANE-EE, if there is a need to isolate multiple application specific credentials from each other on the same client (i.e. with the same underlying base domain name), then SRV-ID would need to be employed.

5. Signaling the Client’s DANE Identity in TLS

The protocol described in the initial version of this document assumes either that client authentication is mandatory, or that where it is optional, clients can handle a Client Certificate Request message from the server without issues if they are not equipped with client certificates. Technically, the TLS protocol specification states that the client may respond with a Client Certificate message with no certificate, and that the server may at its discretion continue the handshake without client authentication. However in practice, problems may arise. There are deployed client software implementations that do not react gracefully when encountering a certificate request that they did not expect.

More importantly, a server may want an explicit indication from the client that it has a DANE record, so as to avoid unnecessary DNS queries in-band with the TLS handshake for clients that don’t support this.

Hence, to address this issue generally, a client identity signaling solution will need to be devised, whereby the client indicates its DANE identity (i.e. its domain name identity and the fact that this identity has an associated TLSA record) to the server. Application specific protocol enhancements are one way to achieve this, e.g. a new SMTP command. A more general way would be to develop a new TLS extension to convey this information.

[Another internet draft is currently being written to define such a TLS extension to convey DANE client identity.]

6. Example TLSA records for clients

The following examples are provided in the textual presentation format of the TLSA record.

An example TLSA record for the client "device1.example.com." and the application "smtp-client". This record specifies the SHA-256 hash of a PKIX CA certificate to authenticate the client’s certificate.

```
_smtp-client.device1.example.com. IN TLSA (
   0 0 1 d2abde240d7cd3ee6b4b28c54df034b9
         7983a1d16e8a410e4561cb106618e971 )
```

An example TLSA record for the client "client2.example.com." and the application "localsvc". This record specifies the SHA-512 hash of the subject public key component of the client's certificate. The usage mode for this record is 3 (DANE-EE) and hence no PKIX validation for this certificate should be performed.

```
_localsvc.client2.example.com. IN TLSA (
   3 1 2 0f8b48ff5fd94117f21b6550aaee89c8
         d8adbc3f433c8e587a85a14e54667b25
         f4dcd8c4ae6162121ea9166984831b57
         b408534451fd1b9702f8de0532ecd03c )
```

7. Changes to Client and Server behavior

[Note: As the client identity signaling solution is developed, this section will undergo enhancements to use it. A future revision of this document will explicitly address the additional use case of raw public keys instead of X.509 certificates.]

A TLS Client conforming to this specification MUST have a signed DNS TLSA record published corresponding to its DNS name and X.509 certificate. The client presents this certificate in the TLS handshake with the server. The presented client certificate MUST have have the client's DNS name specified either in the Subject Alternative Name extension's dNSName type, or the SRVName type.

A TLS Server implementing this specification performs the following steps:

S1 Request a client certificate in the TLS handshake (the "Client Certificate Request" message).

S2 Extract the client identity from the Subject Alternative Name extension's dNSName or SRVName type in the client certificate. (If no client certificate is provided, then the server may terminate the connection, or at its discretion may continue the handshake without client authentication.)

S3 Construct the DNS query name for the corresponding TLSA record. For dNSName, the underscored application service label is prepended to the domain name, corresponding to the application in

use. For SRVName, the DNS query name is identical to the content of the SRVName identifier. See Section 2 for the proposed owner name format.

S4 Look up the TLSA record in the DNS. The response MUST be cryptographically validated using DNSSEC. The server could perform the DNSSEC validation itself. It could also be configured to trust responses obtained via a validating resolver to which it has a secure connection.

S5 Extract the RDATA of the TLSA record and match it to the presented client certificate according to the rules specified in the DANE TLS protocol [RFC6698]. If successfully matched, the client is authenticated and the TLS session proceeds. If not, the session is terminated with a "bad_certificate" alert message.

S6 If there are multiple records in the TLSA record set, then the client is authenticated as long as at least one of the TLSA records matches.

If the presented client certificate has multiple distinct reference identifier types (e.g. a dNSName, and an rfc822Name) then TLS servers configured to perform DANE authentication according to this specification should only examine and authenticate the dNSName or SRVName identity. If the certificate contains both dNSName and SRVName identities, SRVName should be preferred. See [RFC6125] for a description of reference identifiers and matching rules.

If the presented client certificate has multiple dNSName or SRVName identities, then the client MUST use an identity signalling mechanism to indicate the intended name to the server.

Specific applications may be designed to require more detailed validation steps. For example, a server might want to verify the client’s IP address is associated with the certificate in some manner, e.g. by confirming that a secure reverse DNS lookup of that address ties it back to the same domain name, or by requiring an iPAddress component to be included in the certificate. Such details are outside the scope of this document, and should be outlined in other documents specific to the applications that require this behavior.

Servers may have their own whitelisting and authorization rules for which certificates they accept. For example a TLS server may be configured to only allow TLS sessions from clients with certificate identities within a specific domain or set of domains.

8. Raw Public Keys

This specification can also support the use of raw public keys in TLS [RFC7250]. This use case employs only usage mode 3 (DANE-EE) and a selector value of 1 (SPKI) in the DANE TLSA record, as described in [DANEOPS]. It requires the use of the new client identity signaling solution discussed previously.

9. Open Issues

Should this document also consider client identities in the form of e-mail addresses? The use case might be an SMTP client talking to an SMTP submission server. In that case, the email address of a user would most likely be conveyed in the certificate in a subject alt name rfc822Name type. The corresponding TLSA record would have to then have an owner name format similar to the OPENPGPKEY or SMIMEA records. This use case might be best left to the SMIMEA specification to consider.

10. Acknowledgements

This document benefited from discussions with the following people: Duane Wessels, Allison Mankin, Casey Deccio, and Warren Kumari.

11. IANA Considerations

This document includes no request to IANA.

12. Security Considerations

This document makes a narrow update to RFC 6698 by defining the usage of the TLSA record for client TLS certificates. There are no security considerations for this document beyond those described in RFC 6698 and in the specifications for TLS and DTLS [RFC5246], [RFC6347].

13. References

13.1. Normative References

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

[RFC4985] Santesson, S., "Internet X.509 Public Key Infrastructure Subject Alternative Name for Expression of Service Name", RFC 4985, August 2007.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, March 2011.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, August 2012.

[RFC7218] Gudmundsson, O., "Adding Acronyms to Simplify Conversations about DNS-Based Authentication of Named Entities (DANE)", RFC 7218, April 2014.

[RFC7250] Wouters, P., Tschofenig, H., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, June 2014.

13.2. Informative References

[DANEOPS] Dukhovni, V., "Updates to and Operational Guidance for the DANE Protocol", , <https://tools.ietf.org/html/draft-ietf-dane-ops>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

[SRVREG] IANA, ., "Service Name and Transport Protocol Port Number Registry", , <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt>.

Authors' Addresses

Shumon Huque
Verisign Labs

Email: shuque@verisign.com

Dan James
Verisign, Inc.

Email: djames@verisign.com

Viktor Dukhovni
Two Sigma

Email: ietf-dane@dukhovni.org

TLS
Internet-Draft
Intended status: Standards Track
Expires: January 7, 2016

M. Shore
No Mountain Software
R. Barnes
Mozilla
S. Huque
Verisign Labs
W. Toorop
NLNet Labs

July 6, 2015

A DANE Record and DNSSEC Authentication Chain Extension for TLS
draft-shore-tls-dnssec-chain-extension-01

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 certificate 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

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] of a TLS server certificate without performing perform 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 endpoint and a DNS server. And lastly, it allows a TLS client to validate DANE records itself

without 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 records such as TLSA, SMIMEA, etc.

This mechanism is useful for TLS applications that need to address the problems described above, typically web browsers or VoIP and XMPP services. 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, and are less likely to experience traffic interference from misconfigured middleboxes. Furthermore, SMTP MTAs usually employ Opportunistic Security [RFC7435], 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 [DANESMTP] should not use this mechanism.

The extension described here allows a TLS client to request in the client hello message that the DNS validation chain be returned in the (extended) server hello message. If the server is configured for DANE authentication, then it performs the appropriate DNS queries, builds the validation 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 [AGL] and it incorporates his ideas and some of his text. It modifies his approach by using DNS wire formats and assumes that in implementation, the serialized DNSSEC object will be prepared by a DNS-specific module and the validation actions on serialized DNSSEC will also be carried out by a DNS-specific module. An appendix (empty in the 00 version) provides a Python code example of interfacing with a DNS-specific module.

3. DNSSEC Authentication Chain Extension

3.1. Protocol

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

[Placeholder: an upcoming revision of this specification will support the ability for the client to include a set of unexpired cached records it possesses, and correspondingly allow the server to return an authentication chain with those records omitted.]

Servers receiving a "dnssec_chain" extension in the client hello SHOULD return a serialized authentication chain in the extended ServerHello message, using the format described below. If a server is unable to return a authentication chain, or does not wish to return a 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.

3.2. DNSSEC Authentication Chain Data

The "extension_data" field of the "dnssec_chain" extension represents a sequence of DNS resource record sets, which provide a chain from the DANE record being provided to a trust anchor chosen by the server. The "extension_data" field MUST contain a DNSSEC Authentication Chain encoded in the following form:

```
struct {
  opaque rrset<0..2^16-1>;
  opaque rrsig<0..2^16-1>;
} RRset

RRset AuthenticationChain<0..2^16-1>;
```

Each RRset in the authentication chain encodes an RRset along with a signature on that RRset. The "rrsig" field contains the RDATA for the RRSIG record, defined in Section 3.1 of RFC 4034 [RFC4034]. The "rrset" field contains the covered resource records, in the format defined in Section 3.1.8.1 of RFC 4034 [RFC4034]:

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

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

The first RRset in the chain MUST contain the DANE records being presented. The subsequent RRsets MUST be an sequence of DNSKEY and DS RRsets, starting with a DNSKEY RRset. Each RRset MUST authenticate the preceding RRset:

For a DNSKEY RRset, one of the covered DNSKEY RRs MUST be the public key used to verify the previous RRset.

For a DS RRset, the set of key hashes MUST overlap with the preceding set of DNSKEY records.

In addition, a DNSKEY RRset followed by a DS RRset MUST be self-signed, in the sense that its RRSIG MUST verify under one of the keys in the DNSKEY RRSET.

The final RRset in the authentication chain, representing the trust anchor, SHOULD be omitted. In this case, the client MUST verify that the key tag and owner name in the final RRSIG record correspond to a trust anchor.

For example, for an HTTPS server at www.example.com, where there are zone cuts at "com." and "example.com.", the AuthenticationChain structure would comprise the following RRsets (and their corresponding RRSIG signatures):

```
_443._tcp.www.example.com. TLSA
example.com. DNSKEY
example.com. DS
com. DNSKEY
com. DS
. DNSKEY
```

[Some names involving CNAME and DNAMEs may involve multiple branches of the DNS tree. The authors are contemplating enhancements to the AuthenticationChain structure to accommodate these for a future revision of the draft.]

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] of the server’s X.509 certificate, the DNS record set to be serialized is a TLSA record set corresponding to the server’s domain name.

The domain name of the server MUST be that included in the TLS Server Name Indication extension [RFC6066] when present. If the Server Name Indication extension is not present, or if the server does not recognize the provided name and wishes to proceed with the handshake rather than aborting 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 built by starting at the target record and its corresponding RRSIG. Then traversing the DNS tree upwards towards the trust anchor zone (normally the DNS root), for each zone cut, the DS and DNSKEY RRsets and their signatures are added.

In order to meet these formatting requirements, the server must perform some preprocessing on the resource records it receives. It must first compute the uncompressed representation of the RRs, removing DNS name compression [RFC1035], if present. It then extracts the relevant fields from the resource records and assembles them into an RRset.

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 can cache and reuse it in multiple TLS connection handshakes. However, it should keep track of the TTLs and signature validity periods, and requery the records and rebuild the authentication chain as needed. A server implementation could carefully track these parameters and requery the chain correspondingly. Alternatively, it could be configured to rebuild the 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 certificate. In order to do this, it uses the mechanism specified by the DNSSEC protocol [RFC4035]. This mechanism is sometimes implemented in a DNSSEC validation engine or library.

If the authentication chain is correctly verified, the client then performs DANE authentication of the server according to the DANE TLS protocol [RFC6698], and the additional protocol requirements outlined in [DANEOPS].

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. Managed key rollovers typically use a process that can be tracked by verifiers allowing them to automatically update their trust anchors, as described in [RFC5011]. TLS clients using this specification are also expected to use such a mechanism to keep their trust anchors updated. Some operating systems may have a system-wide service to maintain and keep up-to-date the root trust anchor. It may be possible for the TLS client application to simply reference that as its trust anchor, periodically checking whether it has changed.

8. Security Considerations

The security considerations of the normatively referenced RFCs (1035, 4034, 4035, 5246, 6066, 6698) all pertain to this extension. Since the server is delivering a chain of DNS records and signatures to the client, it must take care to 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 [RFC4330], which are already widely used today. TLS clients must support a mechanism to track and rollover the trust anchor key as described in Section 7.

9. 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].

10. 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: Allison Mankin, Duane Wessels, Jeff Hodges, Patrick McManus, and Gowri Visweswaran. We are particularly grateful to Viktor Dukhovni for his detailed review.

11. Test Vectors

[TO BE ADDED LATER. THE ORIGINAL CONTENT WAS OBSOLETE.]

12. References

12.1. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC6066] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, August 2012.

12.2. Informative References

[RFC4330] Mills, D., "Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI", RFC 4330, January 2006.

[RFC5011] StJohns, M., "Automated Updates of DNS Security (DNSSEC) Trust Anchors", STD 74, RFC 5011, September 2007.

[RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

[RFC7120] Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, January 2014.

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, December 2014.

[AGL] Langley, A., "Serializing DNS Records with DNSSEC Authentication", <https://tools.ietf.org/id/draft-agl-dane-serializechain-01.txt>.

[DANESMTP] Dukhovni, V. and W. Hardaker, "SMTP Security via opportunistic DANE TLS", <https://tools.ietf.org/html/draft-ietf-dane-smtp-with-dane-19>.

[DANEOPS] Dukhovni, V., "Updates to and Operational Guidance for the DANE Protocol", <https://tools.ietf.org/html/draft-ietf-dane-ops>.

Appendix A. Pseudocode example

[code goes here]

Appendix B. Test vector

[data go here]

Authors' Addresses

Melinda Shore
No Mountain Software

EMail: melinda.shore@nomountain.net

Richard Barnes
Mozilla

EMail: rlb@ipv.sx

Shumon Huque
Verisign Labs

EMail: shuque@verisign.com

Willem Toorop
NLNet Labs

EMail: willem@nlnetlabs.nl