Internet Engineering Task Force
Internet-Draft
Updates: 6698, 7671 (if approved)
Intended status: Standards Track
Expires: 3 September 2026

S. Huque
Salesforce
V. Dukhovni
OpenSSL Corporation

2 March 2026

TLS Client Authentication via DANE TLSA records
draft-ietf-dance-client-auth-10

Abstract

The DANE TLSA protocol describes how to publish Transport Layer Security (TLS) server certificates or public keys in the DNS. This document updates RFC 6698 and RFC 7671. It describes how to use the TLSA record to publish client certificates or public keys, and also the rules and considerations for using them with TLS. In addition, it defines a new TLS extension, DANE CLient Identity, to convey the client's domain name identity to the server.

Status of This Memo

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

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

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

This Internet-Draft will expire on 3 September 2026.

Copyright Notice

Table of Contents

1. Introduction and Motivation 2
1.1. Requirements Language 3
2. Associating Client Identities in DNS TLSA Records 3
2.1. Format 1: Service specific client identity 3
2.2. Format 2: IOT Device Identity 3
3. Example TLSA records for clients 4
3.1. Format 1: Service Specific Client Identity 4
3.2. Format 2: DevID 4
4. Authentication Model 5
5. Client Identifiers in X.509 certificates 5
6. Signaling the Client's DANE Identity in TLS 5
7. TLS DANE Client Identity Extension 5
8. Changes to Client and Server behavior 6
9. Raw Public Keys 8
10. Acknowledgements 8
11. Security Considerations 8
12. IANA Considerations 9
13. References 9
13.1. Normative References 10
13.2. Informative References 11
Authors' Addresses 11

1. Introduction and Motivation

The Transport Layer Security (TLS) [RFC8446] and DTLS [RFC9147] protocols optionally support the authentication of clients using X.509 certificates [RFC5280] or raw public keys [RFC7250]. TLS applications that perform DANE [RFC6698] [RFC7671] 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) plan to 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. Other potential applications include authenticating the client side of SMTP transport security.

In this document, the term TLS is used generically to describe both the TLS and DTLS (Datagram Transport Layer Security) [RFC6347] protocols. The protocol changes described can also be used with QUIC since QUIC re-uses the TLS handshake.

1.1. Requirements Language

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. Associating Client Identities in DNS TLSA Records

Different applications may have quite different conventions for naming clients via domain names. This document thus does not proscribe a single format, but mentions a few that may have wide applicability.

2.1. Format 1: Service specific client identity

In this format, the owner name of the client TLSA record has the following structure:

[_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 [RFC6698] and [RFC7671], and carries the same meaning.

2.2. Format 2: IOT Device Identity

The Device Identity form of the TLSA record has the following structure:

[devicename]._device.[org-domain-name]

The "_device" label interposed between the client device name labels and the organization domain labels allows management of all client identities to be delegated to a subzone or to another party.

3. Example TLSA records for clients

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

3.1. Format 1: Service Specific Client Identity

An example TLSA record for the client "device1.example.com." and the application "smtp-client". This record specifies the SHA-256 hash of the subject public key component of the end-entity certificate corresponding to the client. The certificate usage for this record is 3 (DANE-EE) and thus is validated in accordance with section 5.1 of RFC 7671.

```
_smtp-client.device1.example.com. IN TLSA (
   3 1 1 d2abde240d7cd3ee6b4b28c54df034b9
         7983a1d16e8a410e4561cb106618e971 )
```

3.2. Format 2: DevID

An example TLSA record for the device named "sensor7" managed by the organization "example.com" This record specifies the SHA-512 hash of the subject public key component of an EE certificate corresponding to the client.

```
sensor7._device.example.com. IN TLSA (
   3 1 2 0f8b48ff5fd94117f21b6550aaee89c8
         d8adbc3f433c8e587a85a14e54667b25
         f4dcd8c4ae6162121ea9166984831b57
         b408534451fd1b9702f8de0532ecd03c )
```

The example below shows a wildcard TLSA record mapped to a TLSA record with a DANE-TA usage mode. This allows all client identifiers matching the wildcard to be authenticated by client certificates issued by an organization managed Certification Authority.

```
*._device.example.com. IN TLSA (
   2 0 1 20efa254ecd5b646e701211095bc3fe4
         423e21941b0b29efb21da57ec944a9b5 )
```

4. Authentication Model

The authentication model assumed in this document is the following:

The client is assigned an identity corresponding to a DNS domain name.

The client has a private and public key pair. Where client certificates are being used, the client also has a certificate binding the name to its public key. The certificate or public key 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 certificate usage modes) or via a PKIX public CA system constraint if the client's certificate was issued by a public CA (using the PKIX-TA or PKIX-EE DANE usage modes).

5. Client Identifiers in X.509 certificates

If the TLS DANE Client Identity extension (see Section 6) is not being used, the client certificate MUST have the client's DNS name specified in the Subject Alternative Name extension's dNSName type, and only one instance of the dNSName type is permitted.

If the client uses a non-empty TLS DANE Client Identity extension, then with DANE-EE(3), the subject name need not be present in the certificate.

6. Signaling the Client's DANE Identity in TLS

The client MUST explicitly signal that it has a DANE identity. The most important reason is that the server needs 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.

The DANE Client Identity TLS extension is used for this purpose. This extension can also be used to convey the actual DANE client identity (i.e. domain name) that the TLS server should attempt to authenticate. This is required when using TLS raw public key authentication, since there is no client certificate from which to extract the client's DNS identity. It is also required when the client certificate contains multiple identities, and only a specific one has a DANE record.

7. TLS DANE Client Identity Extension

The DANE Client Identity Extension type, "dane_clientid", will have a value assigned and registered in the IANA TLS Extensions registry. Its extension data (if not has the following format:

```
opaque ClientName<0..2^8-1>;
```

The ClientName field contains the single domain name of the client in textual presentation format, as described in RFC 1035 [RFC1035], omitting the trailing dot. The lower bound length value of 0 octets indicates that no client name is present.

The wire format of a domain name is limited to 255 octets. In keeping with the practice of most TLS extensions, this extension specifies the use of the textual presentation format of domain names instead. In theory, the presentation format can exceed 255 characters because it allows the expression of any arbitrary octet with the "\DDD" sequence of characters (where DDD is the decimal value). Applications using this extension (and the DANE TLSA Client Authentication protocol more generally) should ensure that client domain names being used do not need to resort to the \DDD syntax by limiting the alphabet suitably, such as only allowing letters, digits, hyphens, and underscores. This ensures that the presentation format client domain name will comfortably fit within the 255 octet limit.

A TLS server implementing this specification MUST send an empty extension of type "dane_clientid" in its CertificateRequest message, to indicate that it understands the extension and is capable of performing DANE client authentication.

A TLS client implementing this specification and intending to use DANE client authentication the TLS server, MUST send an extension of type "dane_clientid" in its Certificate message. Per the TLS protocol, the client is only permitted to send the extension if it sees the corresponding empty extension in the server’s CertificateRequest message. If the client only needs to indicate that it has a DANE record and that the client’s domain name identity can be obtained unambiguously from its certificate, then the extension sent can be empty. If the client needs to send its domain name identity, then the "extension_data" field of the extension MUST contain a "ClientName" data structure populated with the domain name.

8. Changes to Client and Server behavior

A TLS Client conforming to this specification MUST have a DNSSEC [RFC9364] signed TLSA record published corresponding to its DNS name and X.509 certificate or public key. The client presents this certificate or public key in the TLS handshake with the server.

A TLS Server implementing this specification performs the following steps:

1. Request a client certificate in the TLS handshake’s "Certificate Request" message, that includes an empty DANE Client Identity extension.

2. The TLS client supplies its certificate in its Certificate message, and if implementing this protocol, also includes the DANE Client Identity extension in the Certificate message.

3. If the client has sent a non-empty DANE Client Identity extension in its Certificate message, then extract the client’s domain name from the extension. Otherwise, extract the client identity from the Subject Alternative Name extension’s dNSName type.

4. Construct the DNS query name for the corresponding TLSA record. If the TLS DANE client identity extension was present, then this name should be used. Otherwise, the single identity from the client certificate is used.

5. Look up the TLSA record set in the DNS. The response MUST be cryptographically validated using DNSSEC. The server could perform DNSSEC validation itself, authenticating the full chain back to a configured trust anchor (normally the DNS root). Alternatively, it could also be configured to trust responses obtained via a validating resolver to which it has a secure connection, by requiring the Authenticated Data (AD) bit to be set in the responses. If DNSSEC validation fails, the server MUST either abort the connection with a handshake_failure TLS alert, or treat the client as unauthenticated. The option of treating the client as unauthenticated should typically be a configurable option designed to support the case where client authentication for the application in question is optional.

6. Extract the RDATA of the TLSA records and match them to the presented client certificate according to the rules specified in the DANE TLS protocol [RFC6698] [RFC7671]. If successfully matched, the client is authenticated and the TLS session proceeds. If unsuccessful, the server MUST again either terminate the session with a handshake_failure TLS alert, or if configured to do so, treat the client as unauthenticated and proceed with the session.

7. 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, subject to RFC7671 digest agility, which SHOULD be implemented.

If the DANE Client Identity extension is empty, and 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, and only one such dNSName identity should be present in the certificate, otherwise the server MUST abort the connection with a bad_certificate TLS alert.

If the presented client certificate has multiple dNSName identities, then the client MUST use a non-empty TLS DANE client identity extension to unambiguously indicate one of these identities as its client name to the server. If the name in the TLS DANE client identity extension does not match one of the dNSNames in the certificate, then the server MUST abort the connection with a bad_certificate TLS alert.

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. If such rules are not met, the TLS server again could terminate the connection with a handshake_failure TLS alert.

9. Raw Public Keys

When using raw public keys in TLS [RFC7250], this specification requires the use of a non-empty TLS DANE Client Identity extension. The associated DANE TLSA records employ only certificate usage 3 (DANE-EE) and a selector value of 1 (SPKI), as described in [RFC7671].

10. Acknowledgements

The authors thank the many members of the IETF DANCE and TLS working groups for helpful comments and input.

11. Security Considerations

This document updates RFC 6698 by defining the use of the TLSA record for clients. Placing client identities in the DNS may pose privacy issues for certain applications, depending on the nature of the clients, and the structure and content of the client names. Applications employing this protocol should carefully assess those potential issues.

A design goal of TLS 1.3 is that the client identity is encrypted in the Certificate message, and thus protected from disclosure on the wire. DANE authentication however relies on the peer (the TLS server

in this case) looking up the client's DANE record in the DNS. Although, protocol specifications and implementions to encrypt DNS transport exist, they are very far from ubiquitously deployed. Deployers of DANCE client authentication should thus evaluate the risks of the client name being leaked in this manner, until encrypted DNS transport becomes the norm.

One possible way to address this is for the client to construct and send its DANE record and the corresponding full DNSSEC authentication chain to the TLS server in a new Certificate extension within the handshake. A specification to do this in the other direction (from the server to the client) already exists: "TLS DNSSEC Chain Extension" [RFC9102]. However, there are several challenges when considering such an approach from the client end. It is quite a heavy weight operation that some constrained clients may have challenges with. In order to construct the DANE authentication chain, the client would need to perform DNS queries which would still leak its identity to the local network environment without encrypted DNS. Lastly, there maybe client side network impediments to making this work, e.g. middleboxes that prevent DNSSEC enabled queries from succeeding - one of the original motivations for RFC 9102 in the first place. Nevertheless, if appetite to implement this mechanism exists, a future version of this specification could define the details.

The service specific client identity form lends itself to a structure that might make it easy for the same client to have multiple identities corresponding to different applications using the same public key (e.g. by using wildcards and DANE-EE mode), which could make this protocol susceptible to corss protocol attacks where traffic is redirected from one service to another. Deployers of this protocol should avoid this by not sharing per client credentials across distinct applications.

12. IANA Considerations

IANA is requested to create the following entry in the "TLS ExtensionTypes Values" registry:

Extension Name "dane_clientid" with value TBD, "TLS 1.3" column values set to "CR, CT", "DTLS-Only" column set to "N", and "Recommended" column set to "N".

The 'N' designation in the "Recommended" column is because this extension has very specific use cases.

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>.

[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, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[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>.

[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>.

[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>.

[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>.

[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>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/info/rfc9147>.

[RFC9364] Hoffman, P., "DNS Security Extensions (DNSSEC)", BCP 237, RFC 9364, DOI 10.17487/RFC9364, February 2023, <https://www.rfc-editor.org/info/rfc9364>.

13.2. Informative References

[RFC9102] Dukhovni, V., Huque, S., Toorop, W., Wouters, P., and M. Shore, "TLS DNSSEC Chain Extension", RFC 9102, DOI 10.17487/RFC9102, August 2021, <https://www.rfc-editor.org/info/rfc9102>.

[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
Salesforce
Email: shuque@gmail.com

Viktor Dukhovni
OpenSSL Corporation
Email: ietf-dane@dukhovni.org