

Internet Engineering Task Force  
Internet-Draft  
Updates: 6698, 7671 (if approved)  
Intended status: Standards Track  
Expires: 18 August 2022

S. Huque  
Salesforce  
V. Dukhovni  
Two Sigma  
A. Wilson  
Valimail  
14 February 2022

TLS Client Authentication via DANE TLSA records  
draft-huque-dane-client-cert-08

Abstract

The DANE TLSA protocol [RFC6698] [RFC7671] 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 additionally use the TLSA record to publish client certificates or public keys, and also the rules and considerations for using them with TLS.

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 18 August 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components

extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

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

## 1. Introduction and Motivation

The Transport Layer Security (TLS) protocol [RFC5246] [RFC8446] optionally supports the authentication of clients using X.509 certificates [RFC5280] or raw public keys [RFC7250]. TLS applications that perform 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) 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.

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

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 RFC 6698 and RFC 7671, and carries the same meaning.

### 2.2. Format 2: DevId: IOT Device Identity

The DevID form of the TLSA record has the following structure:

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

It is loosely based on the proposed PKI Certificate Identifier Format for Devices [CERTDEVID], but is simpler in form. It makes no distinction between manufacturer issued and locally issued certificates, and does away with the "serial" and "type" labels. The `"_device"` label that precedes the organization domain name allows all the device identities to be delegated to a subzone or to another party.

## 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. 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).

#### 4. Client Identifiers in X.509 certificates

If the TLS DANE Client Identity extension (see Section 5) is not being used, the client certificate **MUST** have the client's DNS name specified in the Subject Alternative Name extension's `dNSName` type.

If the TLS DANE Client Identity extension is in use, then with DANE-EE(3), the subject name need not be present in the certificate.

#### 5. Signaling the Client's DANE Identity in TLS

The client **SHOULD** explicitly signal that it has a DANE identity. The most important reason is that the 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.

The DANE Client Identity TLS extension [TLSCLIENTID] 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 helpful when the client certificate contains multiple identities, and only a specific one has a DANE record.

An additional case where such client signaling is helpful, is one where DANE client authentication is optional, and there is a population of buggy client software that does not react gracefully to receipt of a Certificate Request message from the TLS server. This extension allows TLS servers to deal with this situation by selectively sending a Certificate Request message only to clients that have sent this extension.

## 6. Example TLSA records for clients

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

### 6.1. 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 )
```

### 6.2. 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 )
```

## 7. Changes to Client and Server behavior

A TLS Client conforming to this specification MUST have a signed DNS 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. The client should not offer ciphersuites that are incompatible with its certificate or public key. If the client's certificate has a DANE record with a certificate usage other than DANE-EE, then the presented client certificate MUST have the client's DNS name specified in the Subject Alternative Name extension's dNSName type.

Additionally, when using raw public key authentication, the client MUST send the TLS DANE Client Identity extension [TLSCLIENTID] in its Client Hello message. When using X.509 certificate authentication, it SHOULD send this extension.

A TLS Server implementing this specification performs the following steps:

- \* Request a client certificate in the TLS handshake (the "Client Certificate Request" message). This could be done unconditionally, or only when it receives the TLS DANE Client Identity extension from the client.
- \* If the client has sent a non-empty DANE Client Identity extension, then extract the client's domain name from the extension. Otherwise, extract the client identity from the Subject Alternative Name extension's `dNSName` type.
- \* 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, identities from the client certificate are used.
- \* Look up the TLSA record set 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.
- \* 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** treat the client as unauthenticated (e.g. it could terminate the session, or proceed with the session giving the client access to resources as a generic unauthenticated user).
- \* 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 not present, 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`.

If the presented client certificate has multiple `dNSName` identities, then the client **MUST** use the TLS DANE client identity extension to unambiguously indicate its intended name to the server.

Specific applications may be designed to require additional 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

When using raw public keys in TLS [RFC7250], this specification requires the use of the 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].

## 9. Acknowledgements

TBD.

## 10. IANA Considerations

This document includes no request to IANA.

## 11. Security Considerations

This document updates RFC 6698 by defining the use of the TLSA record for client TLS certificates. There are no security considerations for this document beyond those described in RFC 6698 and RFC 7671 and in the specifications for TLS and DTLS [RFC8446], [RFC5246], [RFC6347].

## 12. References

### 12.1. Normative References

[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>>.

- [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>>.
- [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>>.
- [TLSCLIENTID] Huque, S. and V. Dukhovni, "TLS Extension for DANE Client Identity", <<https://tools.ietf.org/html/draft-huque-tls-dane-clientid>>.

## 12.2. Informative References

- [CERTDEVID] Friel, O. and R. Barnes, "PKI Certificate Identifier Format for Devices", <<https://tools.ietf.org/id/draft-friel-pki-for-devices-00.html>>.
- [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  
Two Sigma

Email: ietf-dane@dukhovni.org

Ash Wilson  
Valimail

Email: ash.wilson@valimail.com

Internet Engineering Task Force  
Internet-Draft  
Intended status: Standards Track  
Expires: 18 August 2022

S. Huque  
Salesforce  
V. Dukhovni  
Two Sigma  
A. Wilson  
Valimail  
14 February 2022

TLS Extension for DANE Client Identity  
draft-huque-tls-dane-clientid-06

Abstract

This document specifies a TLS and DTLS extension to convey a DNS-Based Authentication of Named Entities (DANE) Client Identity to a TLS or DTLS server. This is useful for applications that perform TLS client authentication via DANE TLSA records.

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 18 August 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Overview . . . . .	2
3. DANE Client Identity Extension . . . . .	3
4. Security Considerations . . . . .	4
5. IANA Considerations . . . . .	4
6. Normative References . . . . .	4
Authors' Addresses . . . . .	5

## 1. Introduction

This document specifies a Transport Layer Security (TLS) extension [RFC6066] to convey a DANE [RFC6698] Client Identity to the TLS server. This is useful for applications that perform TLS client authentication via DANE TLSA records, as described in [DANECLIENT]. The extension could be empty to indicate to the server that the client has a DANE record and that the server can perform DANE authentication of the client with the identity extracted from the client certificate. Or the extension can contain the full client identity, in the form of the DNS domain name that is expected to have a DANE TLSA record published for it.

This extension supports both TLS [RFC5246] [RFC8446] and DTLS [RFC6347], and the term TLS in this document is used generically to describe both protocols.

## 2. Overview

When TLS clients use X.509 client certificates or raw public keys that are authenticated via DANE TLSA records, it is useful for them to convey their intent to be authenticated via DANE, or even to convey their complete DANE identity to the server. The TLS extension defined in this document is used to accomplish this.

In the case of X.509 client certificates, a TLS server can learn the client's identity by examining subject alternative names included in the certificate itself. However, without a mechanism such as the one defined in this extension, the TLS server cannot know apriori that

the client has a published TLSA record, and thus may unnecessarily issue DNS queries for DANE TLSA records in-band with the TLS handshake even in cases where the client has no TLSA record associated with it. When multiple identities are present in the certificate, a client must use this extension to specify exactly which one the server should use. An additional situation in which this extension helps is where some TLS servers may need to selectively prompt for client certificate credentials only for clients that are equipped to provide certificates.

When TLS raw public keys [RFC7250] are being used to authenticate the client, the client uses this extension to explicitly indicate to the server what its domain name identity is (since there is no X.509 certificate from which the identity can be extracted).

Detailed protocol behavior of TLS clients and servers is described in [DANECLIENT].

### 3. 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 empty) has the following format:

opaque ClientName<1..2<sup>8</sup>-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.

A TLS server implementing this specification MUST send an empty extension of type "dane\_clientid" to indicate that it understands the extension and is capable of performing DANE client authentication. In TLS 1.2, the empty extension is sent in the ServerHello message. In TLS 1.3, it is sent in the CertificateRequest message.

A TLS client implementing this specification SHOULD send an extension of type "dane\_clientid". If the client only needs to indicate that it has a DANE record and that the client's domain name identity can be obtained 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.

In TLS 1.2, the client extension is sent in the ClientHello message. In TLS 1.3, it is sent in the Certificate message. Additionally, in TLS 1.3, the client is only permitted to send the extension if it sees the corresponding empty extension in the server's CertificateRequest message.

#### 4. Security Considerations

In TLS 1.3, this extension is sent in the CertificateRequest and Certificate messages, which are encrypted.

In TLS 1.2, this extension cannot be encrypted. When used with TLS 1.2, to prevent unnecessary privacy leakage of the client's name in cleartext, a TLS client implementing this specification should be configured to only send this extension to TLS servers it intends to perform client authentication with.

#### 5. IANA Considerations

This extension requires the registration of a new value in the TLS ExtensionsType registry.

#### 6. Normative References

[DANECLIENT]

Huque, S., Dukhovni, V., and A. Wilson, "TLS Client Authentication via DANE TLSA Records", 2 May 2021, <<https://tools.ietf.org/html/draft-huque-dane-client-cert>>.

[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>>.

[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>>.

[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>>.
- [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>>.

#### Authors' Addresses

Shumon Huque  
Salesforce

Email: [shuque@gmail.com](mailto:shuque@gmail.com)

Viktor Dukhovni  
Two Sigma

Email: [ietf-dane@dukhovni.org](mailto:ietf-dane@dukhovni.org)

Ash Wilson  
Valimail

Email: [ash.wilson@valimail.com](mailto:ash.wilson@valimail.com)

DANCE  
Internet-Draft  
Intended status: Informational  
Expires: 25 April 2022

A. Wilson  
Valimail  
S. Huque  
Salesforce  
O. Johansson  
Edvina.net  
22 October 2021

An Architecture for DNS-Bound Client and Sender Identities  
draft-wilson-dance-architecture-00

Abstract

TODO

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 25 April 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Conventions and Definitions . . . . .	3
3. Communication Patterns . . . . .	4
3.1. Client/Server . . . . .	4
3.2. Peer2peer . . . . .	5
3.3. Decoupled . . . . .	5
4. Use Cases . . . . .	5
4.1. Mutual TLS Authentication . . . . .	5
4.1.1. Device to cloud . . . . .	5
4.1.2. Oauth2 . . . . .	6
4.1.3. Edge Computing . . . . .	6
4.1.4. SIP and WebRTC inter-domain privacy . . . . .	6
4.1.5. DNS over TLS client authentication . . . . .	7
4.1.6. SMTP, STARTTLS . . . . .	7
4.1.7. Network Access . . . . .	7
4.1.8. LoRaWAN . . . . .	7
4.2. Object Security . . . . .	7
4.2.1. Structured data messages: JOSE/COSE . . . . .	8
4.3. Operational anomaly reporting . . . . .	8
4.3.1. MUD reporting for improper provisioning . . . . .	8
4.3.2. XARF for abuse reporting . . . . .	8
4.4. Adjacent Ecosystem Components . . . . .	8
4.4.1. Certification Authority . . . . .	8
5. Security Considerations . . . . .	8
5.1. Confidentiality . . . . .	8
5.2. Availability . . . . .	9
5.2.1. DNS Scalability . . . . .	9
5.2.2. Change of ownership . . . . .	9
6. IANA Considerations . . . . .	9
Appendix A. Acknowledgments . . . . .	10
Authors' Addresses . . . . .	10

## 1. Introduction

A digital identity, in an abstract sense, possesses at least two features: an identifier (or name), and a means of proving ownership of the identifier. One of the most resilient mechanisms for tying an identifier to a method for proving ownership of the identifier is the digital certificate, issued by a well-run Certification Authority (CA). The CA acts as a mutually trusted third party, a root of trust.

Certificate-based identities are limited in scope by the issuing CA, or by the namespace of the application responsible for issuing or validating the identity.



An example of this limitation is well-illustrated by organizational Public Key Infrastructure (PKI). Organizational PKI is very often coupled with email and LDAP systems, and can be used for associating a human or machine identity identifier with a public key. Within the organization, authentication systems already agree on the roots of trust for validating entity certificates issued by organizational PKI.

Attempting to use organizational PKI outside the organization can be challenging. In order to authenticate a certificate, the certificate's CA must be trusted. CAs have no way of controlling identifiers in certificates issued by other CAs. Consequently, trusting multiple CAs at the same time can enable entity identifier collisions. Asking an entity to trust your CA implies trust in anything that your CA signs. This is why many organizations operate a private CA, and require devices connecting to the organization's networks or applications to possess certificates issued by the organization's CA.

These limitations make the implementation and ongoing maintenance of a PKI costly, and have a chilling effect on the broader adoption of certificate-based IoT device identity. If certificate-based device identity were easier to manage, more broadly trusted, and less operationally expensive, more organizations and applications would be able to use it.

The lack of trust between PKI domains has lead to a lack of simple and globally scalable solutions for secure end-to-end inter-domain communication between entities, such as SIP phones, email and chat accounts and IoT devices belonging to different organizations.

DANCE seeks to make PKI-based IoT device identity universally discoverable, more broadly recognized, and less expensive to maintain by using DNS as the constraining namespace and lookup mechanism. DANCE builds on patterns established by the original DANE RFCs to enable client and sending entity certificate, public key, and trust anchor discovery. DANCE allows entities to possess a first-class identity, which, thanks to DNS, may be trusted by any application also trusting the DNS. A first-class identity is an application-independent identity.

## 2. Conventions and Definitions

```
{::boilerplate bcpl4-tagged}
```

\*This section will be interesting to define. We have great examples of identity terminology in the <https://datatracker.ietf.org/doc/html/draft-sarikaya-t2trg-sbootstrapping-06> document, but this document also admits that there is semantic drift on terms like "bootstrapping", depending on who's talking.\*

\*Identity provisioning:\* This refers to the set of tasks required to securely provision an asymmetric key pair for the device, sign the certificate (if the public credential is not simply a raw public key), and publish the public key or certificate in DNS. Under some circumstances, these steps are not all performed by the same party or organization. A manufacturer may instantiate the key pair, and a systems integrator may be responsible for issuing (and publishing) the device certificate in DNS. In some circumstances, a manufacturer may also publish device identity records in DNS. In this case, the system integrator needs to perform network and application access configuration, since the identity already exists in DNS.

\*Security Domain:\* DNS-bound client identity allows the device to establish secure communications with any server with a DNS-bound identity, as long as a network path exists, the entity is configured to trust its communicating peer by name, and agreement on protocols can be achieved. The act of joining a security domain, in the past, may have involved certificate provisioning. Now, it can be as simple as using a manufacturer-provisioned identity to join the device to the network and application. Is the security domain defined by how broadly the identity is recognized, or by the breadth of the application or network access policy?

### 3. Communication Patterns

#### 3.1. Client/Server

Client/server communication patterns imply a direct connection between an entity which provides a service (the server), and an entity which initiates a connection to the server, called a client. A secure implementation of this pattern includes a TLS-protected session directly between the client and the server. A secure implementation may also include public key-based mutual authentication.

Extending DANE to include client identity allows the server to authenticate clients independent of the private PKI used to issue the client certificate. This reduces the complexity of managing the CA certificate collection, and mitigates the possibility of client identifier collision.

### 3.2. Peer2peer

The extension also allows an application to find an application identity and set up a secure communication channel directly. This pattern can be used in mesh networking, IoT and in many communication protocols for multimedia sessions, chat and messaging.

### 3.3. Decoupled

Decoupled architecture, sometimes referred to as store-and-forward, provides no direct connection between the producer and consumer of information. The producer and consumer are typically separated by at least one layer of messaging-oriented middleware. The Messaging-oriented middleware components may act as a server for the purpose of establishing TLS sessions for the producer and consumer. This allows the assertion of identity between the middleware and producer, and the middleware and consumer. The trust relationship between the producer and consumer is based on the presumed trustworthiness of the middleware, unless an identity can be attached to the message itself, independent of transport and middleware components.

Within many existing store-and-forward protocols, certificates may be transmitted within the signed message itself. An example of this is S/MIME. Within IoT applications, we find that networks may be more constrained. Including certificates in message payloads can present an unnecessary overhead on constrained network links. Decoupled applications benefit from an out-of-band public key discovery mechanism, which may enable the retrieval of certificates only when needed, and sometimes using a less expensive network connection.

## 4. Use Cases

### 4.1. Mutual TLS Authentication

#### 4.1.1. Device to cloud

Direct device-to-cloud communication is common in simple IoT applications. Authentication in these applications is usually accomplished using shared credentials like API keys, or using client certificates. Client certificate authentication requires the implementer to manage a certification authority. The CA trust anchor certificate is installed into the cloud application, and used in the TLS authentication process.

Using DANE for device identity can allow parties other than the implementer to operate the CA. A hardware secure element provider can provide a pre-established identity, with the certificate or public key already published in DNS. This can make PKI-based identity more approachable for small organizations which currently lack the resources to operate an organizational CA.

#### 4.1.2. OAuth2

[This can be a broad topic. Should we include, or wait until a re-chartering to update?]

#### 4.1.3. Edge Computing

<https://datatracker.ietf.org/doc/html/draft-hong-t2trg-iot-edge-computing-01> (Edge Computing) may require devices to mutually authenticate in the field. A practical example of this pattern is the edge computing in construction use case [<https://datatracker.ietf.org/doc/html/draft-hong-t2trg-iot-edge-computing-01#section-6.2.1>]. Using traditional certificate-based identity, the sensor and the gateway may have certificates issued by the same organizational PKI. By using DANE for client and sender identity, the sensor and the gateway may have identities represented by the equipment supplier, and still be able to mutually authenticate. Important sensor measurements forwarded by the gateway to the cloud may bear the DNS name and signature of the originating sensor, and the cloud application may authenticate the measurement independent of the gateway which forwarded the information to the application.

#### 4.1.4. SIP and WebRTC inter-domain privacy

End to end security in SIP is currently based on a classical S/MIME model which has not received much implementation. There are also SIP standards that build upon a trust chained anchored on the HTTP trust chain (SIP identity, STIR). WebRTC has a trust model between the web browser and the servers using TLS, but no inter-domain trust infrastructure. WebRTC lacks a definition of namespace to map to DNS, where SIP is based on an email-style addressing scheme. For WebRTC the application developer needs to define the name space and mapping to DNS.

By using DNS as a shared root of trust SIP and WebRTC end points can anchor the keys used for DTLS/SRTP media channel setup. In addition, SIP devices can establish security in the SIP messaging by using DNS to find the callee's and the callers digital identity.

[<https://datatracker.ietf.org/doc/html/draft-johansson-sipcore-dane-sip>]

\*NOTE: include reference to earlier drafts for SIP + DANE\*

#### 4.1.5. DNS over TLS client authentication

#### 4.1.6. SMTP, STARTTLS

#### 4.1.7. Network Access

##### 4.1.7.1. EAP-TLS

<https://datatracker.ietf.org/doc/html/rfc5216> (EAP-TLS) is frequently used for network authentication, for IoT and traditional IT equipment. RADIUS is a protocol and server technology frequently used for supporting the server side of EAP-TLS authentication. Guidance for implementing RADIUS strongly encourages the use of a single common CA for all supplicants. The use of DANE for client identity can allow the safe use of any number of CAs. DNS acts as a constraining namespace, which prevents two unrelated CAs from issuing valid certificates bearing the same identifier. Certificates represented in DNS are valid, and all others are un-trusted.

##### 4.1.7.2. RADSEC

The RADIUS protocol has a few recognized security problems. <https://datatracker.ietf.org/doc/html/rfc6614> (RADSEC) addresses the challenges related to the weakness of MD5-based authentication and confidentiality over untrusted networks by establishing a TLS session between the client and the RADIUS server. RADIUS datagrams are then transmitted within the TLS session. Updating the RADSEC standard to include the use of DANE for client and server identity would allow a RADIUS server and client to mutually authenticate, independent of the client's and server's issuing CAs. The benefit for this use case is that a hosted RADIUS service may mutually authenticate any client device, like a WiFi access point or ethernet switch, via RADSEC, without requiring the distribution of CA certificates.

##### 4.1.8. LoRaWAN

\*We should ask S. if he wants to contribute to this section\*

#### 4.2. Object Security

#### 4.2.1. Structured data messages: JOSE/COSE

JOSE and COSE provide formats for exchanging authenticated and encrypted structured data. JOSE defines the x5u field in <https://datatracker.ietf.org/doc/html/rfc7515#section-4.1.5> (RFC7515), and COSE defines a field of the same name in <https://datatracker.ietf.org/doc/html/draft-ietf-cose-x509-08#section-2> (draft-ietf-cose-x509) which can be used for out-of-band x.509 certificate discovery. By adopting DANE for out-of-band certificate discovery, CBOR and JSON data may be authenticated, even if the originating entity does not have IP connectivity, provided that the originating entity's certificate is discoverable in DNS and the authenticating system has access to the DNS.

#### 4.3. Operational anomaly reporting

##### 4.3.1. MUD reporting for improper provisioning

##### 4.3.2. XARF for abuse reporting

#### 4.4. Adjacent Ecosystem Components

##### 4.4.1. Certification Authority

### 5. Security Considerations

#### 5.1. Confidentiality

DNS clients should use DNS over TLS with trusted DNS resolvers to protect the identity of authenticating peers. Integrity The integrity of public keys represented in DNS is most important. An altered public key can enable device impersonation, and the denial of existence for a valid identity can cause devices to become un-trusted by the network or the application.

Compartmentalizing failure domains within an application is a well-known architectural best practice. Within the context of protecting DNS-based identities, this compartmentalization may manifest by hosting an identity zone on a DNS server which only supports the resource record types essential for representing device identities. This can prevent a compromised identity zone DNS server from presenting records essential for impersonating web sites under the organization's domain name.

The naming pattern suggested in (client identity draft) includes an underscore label (\_device) which also prevents the issuance of Web PKI-validating certificates in the event a DNS server hosting a client identity zone, which is capable of presenting A and AAAA records, is compromised.

## 5.2. Availability

### 5.2.1. DNS Scalability

In the use case for IoT an implementation must be scalable to a large amount of devices. In many cases, identities may also be very short lived as revocation is performed by simply removing a DNS record. A zone will have to manage a large amount of changes as devices are constantly added and de-activated.

In these cases it is important to consider the architecture of the DNS zone and when possible use a tree-like structure with many subdomain parts, much like reverse DNS records or how telephone numbers are represented in the ENUM standard (RFC 6116).

If an authoritative resolver were configured to respond quite slowly (think slow loris), is it possible to cause a DoS on the TLS server via complete exhaustion of TCP connections?

The availability of a client identity zone is essential to permitting clients to authenticate. If the DNS infrastructure hosting client identities becomes unavailable, then the clients represented by that zone cannot be authenticated.

\*OEJ: We may want to have a discussion with the IETF DNS directorate. The scalability section above is from a discussion with one of the members...\*

### 5.2.2. Change of ownership

What happens if an organization owning the client identity goes out of business? What's the best way to transfer an identifier to another zone? <note: there may be an opportunity here to take advantage of EST, or another protocol supporting certificate renewal, to allow client devices to rotate to another zone>

## 6. IANA Considerations

This document has no IANA actions.

## Appendix A. Acknowledgments

TODO acknowledge.

### Authors' Addresses

Ash Wilson  
Valimail

Email: ash.d.wilson@gmail.com

Shumon Huque  
Salesforce

Email: shuque@gmail.com

Olle Johansson  
Edvina.net

Email: oej@edvina.net