Adopted and renamed documents

DANE TLS Client Authentication:

draft-ietf-dance-client-auth-00

TLS Extension for DANE Client Identity:

draft-ietf-dance-tls-clientid-00
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.
to confirm to TLS 1.3 protocol requirements

RFC 8446 (TLS 1.3), Section 4.4.2

[...]

extensions: A set of extension values for the CertificateEntry. The "Extension" format is defined in Section 4.2. Valid extensions for server certificates at present include the OCSP Status extension [RFC6066] and the SignedCertificateTimestamp extension [RFC6962]; future extensions may be defined for this message as well. Extensions in the Certificate message from the server MUST correspond to ones from the ClientHello message. Extensions in the Certificate message from the client MUST correspond to extensions in the CertificateRequest message from the server. If an extension applies to the entire chain, it SHOULD be included in the first CertificateEntry.
TLS CLIENT

Key ^ ClientHello
Exch | + key_share*
| + psk_key_exchange_modes*
v + pre_shared_key*

--->

ServerHello ^ Key
+ key_share* | Exch
+ pre_shared_key* v
{EncryptedExtensions} ^ Server
{CertificateRequest} v Params
*+DANE Client ID ext}{Certificate*} ^
{CertificateVerify*} | Auth
{Finished} v

<------- [Application Data*]

^ {Certificate
+DANE Client ID ext}]
Auth | {CertificateVerify*}
v {Finished}  

[Application Data]  

--->

[Application Data]

no longer optional
Comment on list from Michael Richardson:

“I think that the introduction is very weak; I think that more references and integration with the to-be-adopted architecture document will solve that problem.

I suggest we write "IoT" rather than "IOT"
Discussion & next steps

- Protocol specification is largely done in our opinion. What’s missing or remains to be done?

- Working on the architecture doc and more detailed description of application use cases may inform other enhancements.
- As will implementation experience (see other talk today).
Additional Background Slides for Reference

(will not be presented)
Protocol Goal & History

- **Goal:** Authenticate client side of TLS connection with DANE
- **History**
  - Drafts originally developed in mid 2015
  - Target use cases: IOT device authentication & SMTP Transport security
Protocol Summary

- TLS Client has a DNS domain name identity
  - A public/private key pair & a certificate binding the public key to the domain name
  - Corresponding DANE TLSA record published in DNS

- TLS server
  - Sends Certificate Request message in handshake; extracts client identity from presented certificate, constructs TLSA query, validates DANE TLSA response with DNSSEC
Protocol Summary

- New TLS extension for conveying client’s DANE identity to the server
  - For signaling support for DANE TLS client authentication (empty extension if signal only)
  - For conveying client DNS identity when used with TLS raw public key auth (RFC 7250)
  - In TLS 1.3, this extension is carried in the (encrypted) Client Certificate message.
  - In TLS 1.2 it is carried in the first client Client Hello extension, and thus has no provision for privacy protection.
  - (Optionally, the server can also send an empty extension to signal that it supports this capability. TLS 1.3: Certificate Request message, TLS 1.2: Server Hello extension)
Client DNS Naming Convention

Draft is not proscriptive, but proposes 2 naming formats that may be generally suitable for many types of applications.

Format 1: Service specific client identity

_service.[client-domain-name]

e.g.

_app-smtp-client.relay1.example.com

1st label identifies the application service name. The remaining labels are composed of the client domain name. Allows the same client to have distinct authentication credentials for distinct application services.
Client DNS Naming Convention

Format 2: (IOT?) Device Identity

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

e.g.

a1b2c3._device.subdomain.example.net.

- “a1b2c3”: device identifier (could be multiple left most labels)
- _device: identity grouping label
- subdomain: organizational label(s) (optional)
- example.net: organizational domain
sensor7._device.example.com. IN TLSA ( 
    3 1 2 
      0f8b48ff5fd94117f21b6550aaee89c8 
      d8adbc3f433c8e587a85a14e54667b25 
      f4dcd8c4ae6162121ea9166984831b57 
      b408534451fd1b9702f8de0532ecd03c )
TLS Client
  e.g. IOT Device

---

TLS Server
  e.g. IOT Controller

---

TLS Handshake Start

---

Server Certificate; Client Certificate Request

---

Client Certificate + DANE Indication

---

Verify client’s certificate against DANE TLSA record in the DNS
Protocol annotation for TLS 1.3
**TLS CLIENT**

Key  ^ ClientHello  
Exch | + key_share*
      | + psk_key_exchange_modes*
      v + pre_shared_key*

**TLS SERVER**

ServerHello ^ Key  
      + key_share* | Exch  
      + pre_shared_key* v
      {EncryptedExtensions} ^ Server
      {CertificateRequest} v Params

      {Certificate*} ^
      {CertificateVerify*} | Auth
      {Finished} v

      ^ {Certificate}

<-------- [Application Data*]

Auth | {CertificateVerify*}
      v {Finished}  

[Application Data]  

<--------> [Application Data]
**TLS CLIENT**

Key ^ ClientHello
Exch | + key_share*
    | + psk_key_exchange_modes*
v + pre_shared_key*

**TLS SERVER**

----->

ServerHello ^ Key
    + key_share* | Exch
    + pre_shared_key* v
{EncryptedExtensions} ^ Server
{CertificateRequest} v Params
{Certificate*} ^
{CertificateVerify*} | Auth
{Finished} v

<-------- [Application Data*]

^ {Certificate}

Auth | {CertificateVerify*}
    v {Finished}

----->

[Application Data] <- ------ [Application Data]
Empty extension: convey intent to be authenticated via DANE. For raw pubkey authentication, convey client’s full domain name.
Extract client’s identity, lookup TLSA RRset and authenticate the client’s cert or pubkey.
1-Slide DANE Primer

DANE record in this example specifies the SHA256 hash of the subject public key of the certificate that should match the End-Entity certificate. Authenticated entirely in the DNS.