DANISH

IETF 111
July 27th 2021

Defining The Problem Space

IoT Ecosystem Challenges

Private PKI everywhere

Establishing trust across private PKI domains is challenging

No technical controls prevent naming collisions across PKI

Discovery API for CA and EE certs oftentimes proprietary

PKI over-consolidation to prevent naming collisions and ease trust challenges

IoT Ecosystem Challenges

Private PKI needs a broadly useful lookup mechanism

Specifically, we need a lookup mechanism which:

...works across private PKI (enabling mutual TLS authentication)

...prevents naming collisions

...makes credential rotation easier

...works well for constrained platforms

Create safe bridges between islands of trust

Islands Of Trust

CA bundle makes web PKI work well

- Imagine the web without the browser CA bundle...

IoT typically uses private PKI for client identity

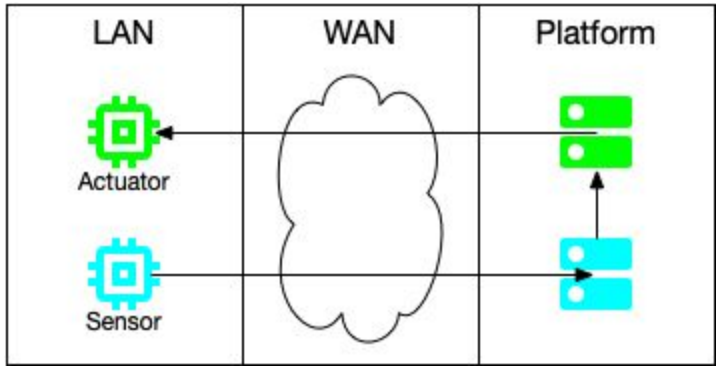
- Web PKI for this use case brings unnecessary cost and complexity

Agreement on IoT roots of trust -> CA cert distribution or consolidation

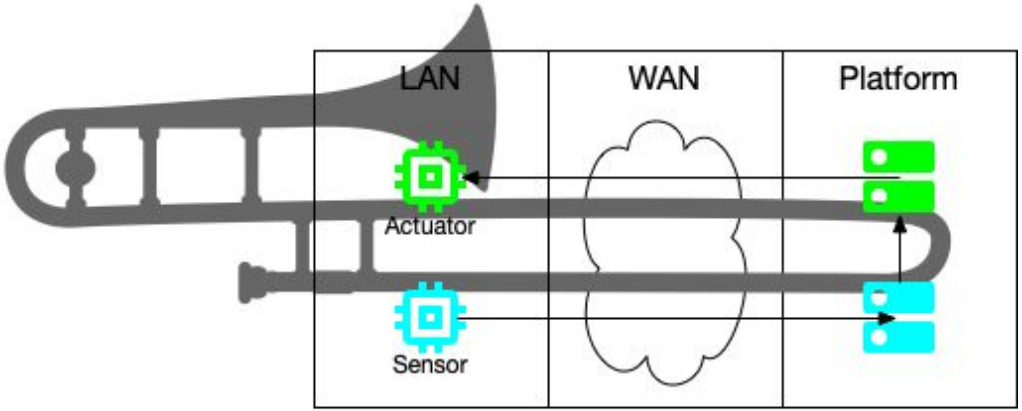
Multiple PKI invites naming collision, so consolidation is favored

Entire org on the same trust island, cross-org M2M is difficult

Suboptimal Information Flows

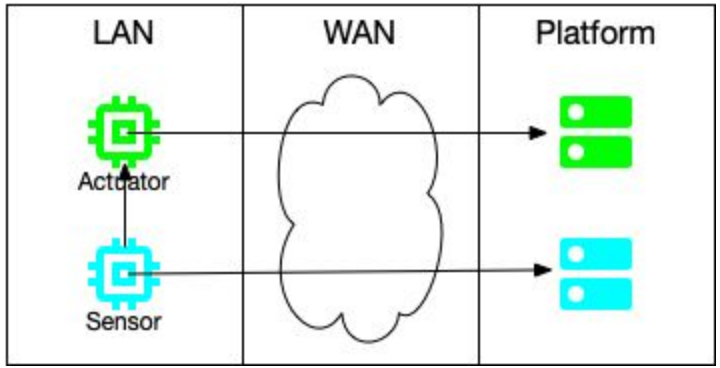


Suboptimal Information Flows (tromboning)



Suboptimal Information Flows

This is what we want:



Initial Work Areas

1-Slide DANE Primer

port, protocol, domain name

data (hex encoded) associated with the certificate or public key

```
_25._tcp.mail.example.com. IN TLSA (
                3 1 1  d2abde240d7cd3ee6b4b28c54df034b9
                       7983a1d16e8a410e4561cb106618e971 )
```

Parameters: Usage, Selector, Matching-Type

Usage 0: PKIX-CA: CA Constraint
Usage 1: PKIX-EE: Service Cert Constraint
Usage 2: DANE-TA: Trust Anchor Assertion
Usage 3: DANE-EE: Domain Issued Certificate

Selector 0: Full Certificate
Selector 1: Public Key (could be raw)

Matching-Type 0: Full Content
Matching-Type 1: SHA-256 Hash
Matching-Type 2: SHA-512 Hash

DANE record specifies the SHA256 hash of the subject public key of the certificate that should match the End-Entity certificate. Authenticated entirely in the DNS (no PKIX involved).

TLS Client Authentication with DANE

- Original drafts developed in mid 2015; refreshed late last year
  - TLS Client Authentication via DANE TLSA Records:
    - https://tools.ietf.org/html/draft-huque-dane-client-cert
  - TLS Extension to convey DANE Client Identity:
    - https://tools.ietf.org/html/draft-huque-tls-dane-clientid
- Target use cases: IOT & SMTP Transport Security

Protocol Summary

- Client has:
  - 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

- (D)TLS server
  - Sends Certificate Request message in handshake; extracts client identity from presented certificate, constructs TLSA record; queries, and validates DANE TLSA response

Protocol Summary

- New TLS extension for conveying client's identity
  - 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 flight Client extension and has no provision for privacy protection)

Client DNS Naming Convention

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

Format 1: Service specific client identity

_service.[client-domain-name]

e.g. _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

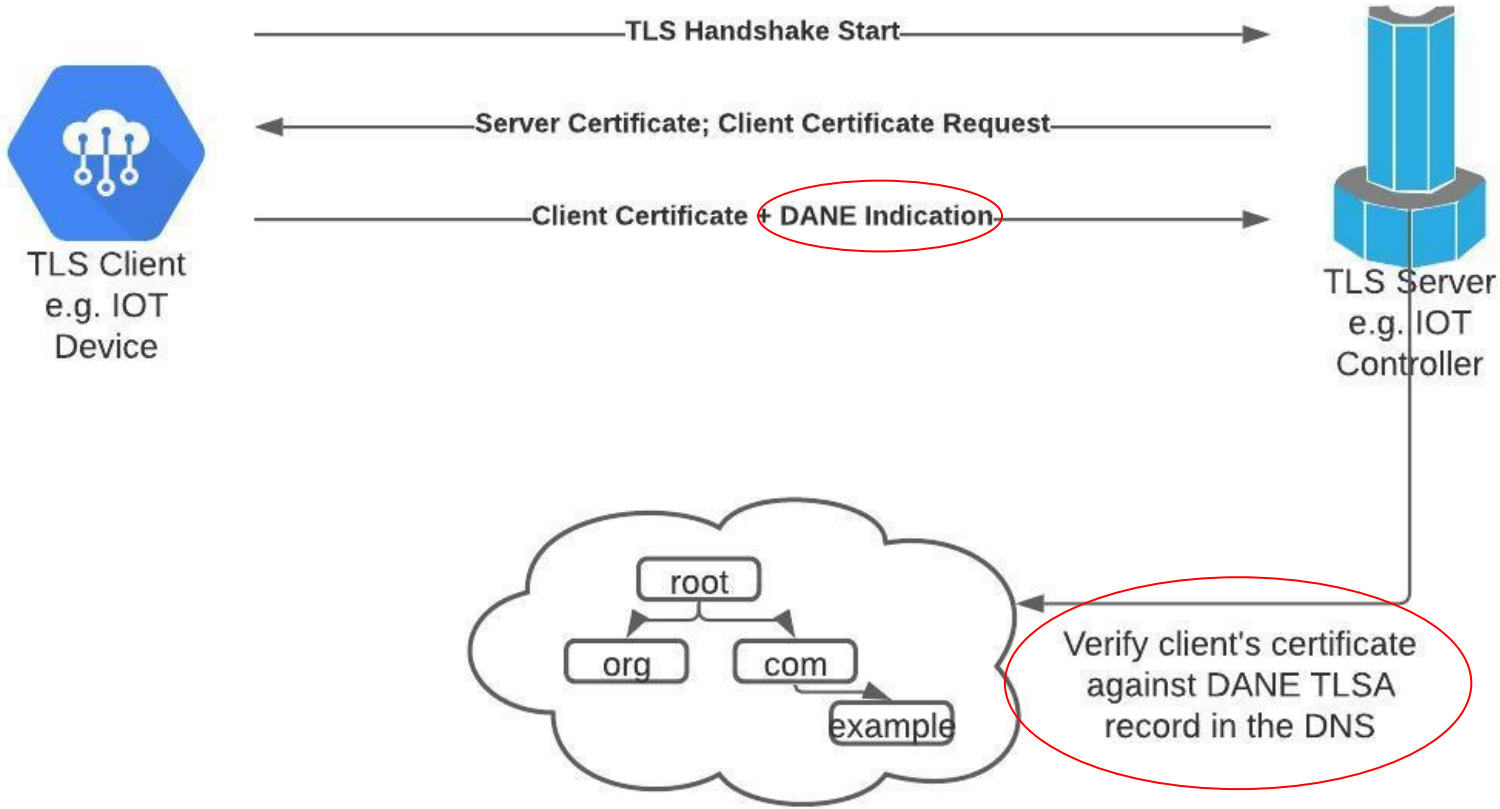
[deviceid]._device.[org-domain-name]

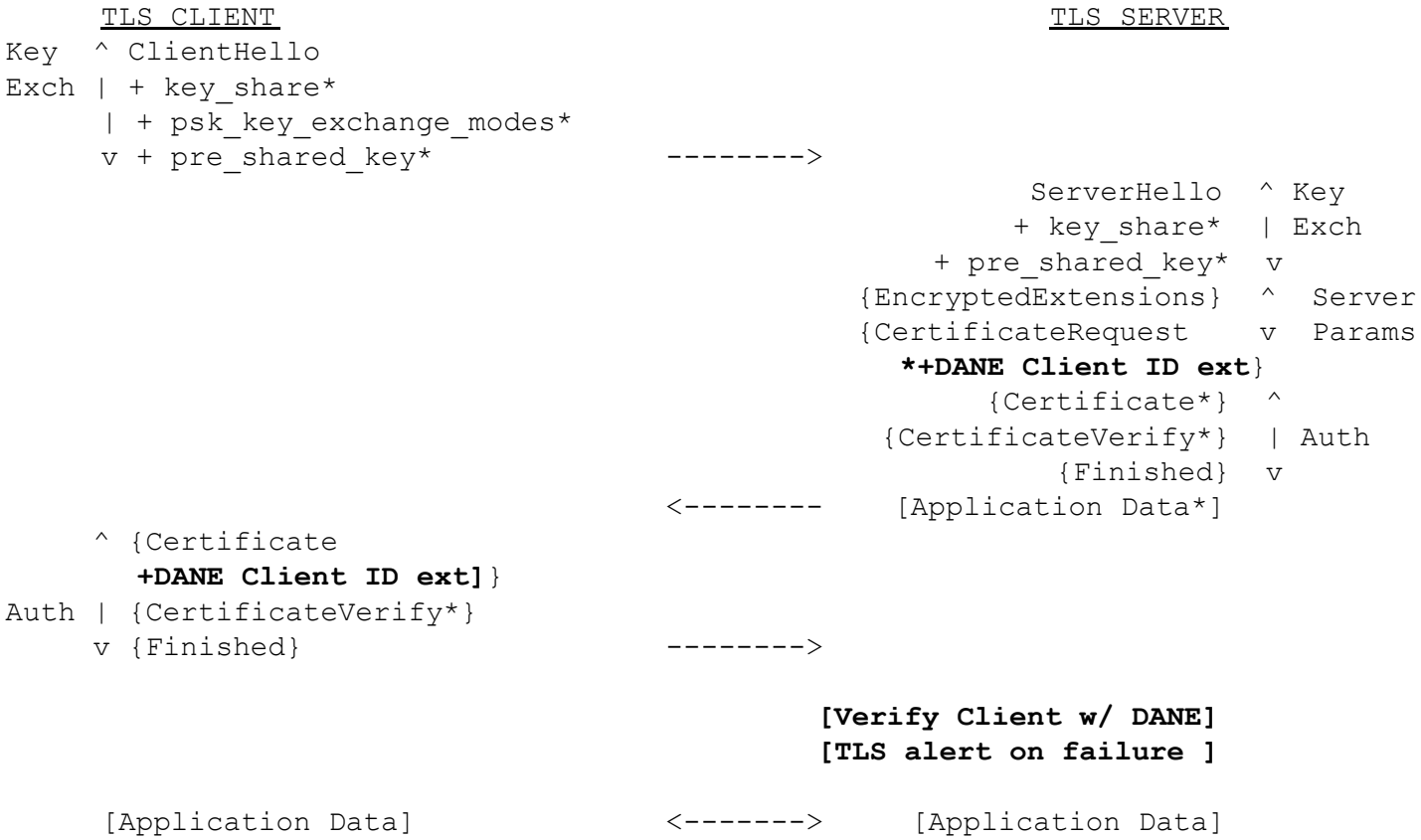
e.g.

a1b2c3._device.subdomain.example.net.

- a1b2c3: device identifier (could be multiple leftmost labels)
- _device: identity grouping label
- subdomain: organizational label (optional)
- example.net: organizational domain

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


TLS Handshake Start
Server Certificate; Client Certificate Request
Client Certificate + DANE Indication
TLS Client
e.g. IOT
Device
TLS Server
e.g. IOT
Controller
root
org
com
example
Verify client's certificate
against DANE TLSA
record in the DNS


TLS CLIENT
TLS SERVER
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}
--------->
[Verify Client w/ DANE]
[TLS alert on failure]
[Application Data]
<------->
[Application Data]

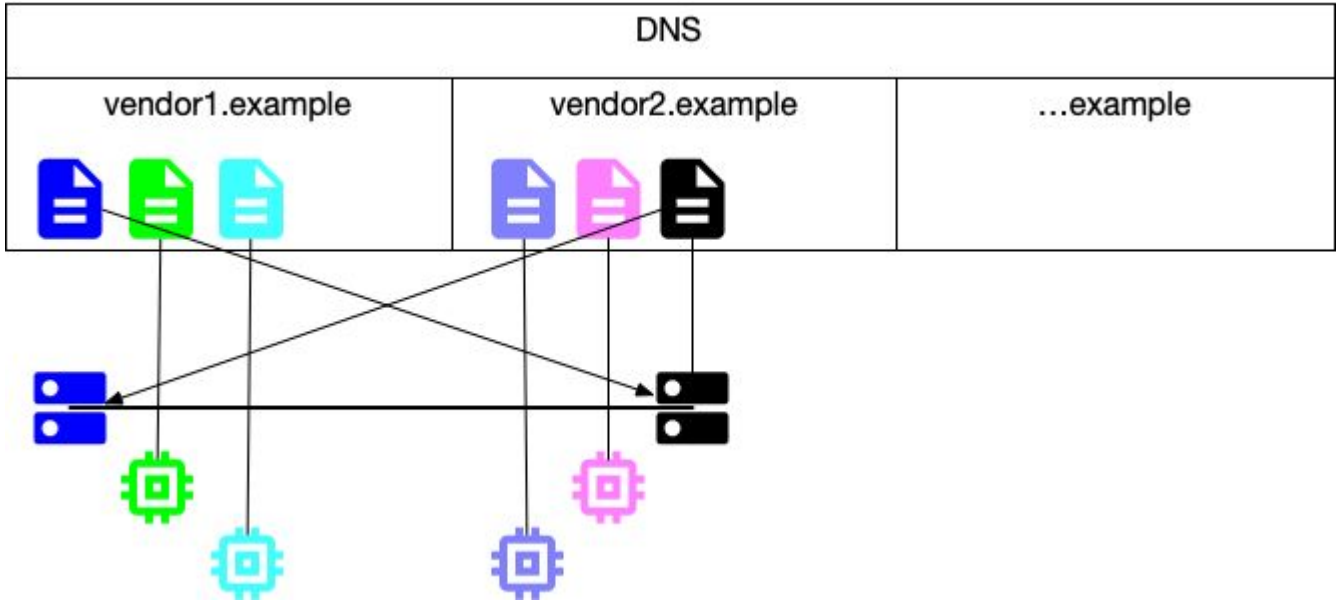
TLS Mutual Authentication With DANE

Simplify PKI management tasks:

- Certificate rotation happens via your own DNS.
- Certificate rotation happens as frequently as desired, TTL is only delay.

Attribution for authenticating peer is straightforward (DNS hierarchy)

TLS Mutual Authentication With DANE



Architecture Document

DNS-Based Device Identity with DANE

Architecture doc describes how DANE device identity fits into various use cases

Some Proposed Topics:

- Terminology
- Overlap with other architectures (like draft-hong-t2trg-iot-edge-computing)
- Network access: EAP-TLS, RADSEC
- Decoupled application authentication: LoRaWAN
- Client/Server: TLS authentication middleware behavior
- Object Security: Message signing << Should this be an update on re-charter?

Would this be better as an update to RFC 7671?

Use Case: Cloud Service Authentication

Cloud Service Client Authentication

Desired behavior:

- Use supplier-provisioned DANE identity for TLS authentication
- Pick best-of-breed suppliers
- Safely use supplier PKI

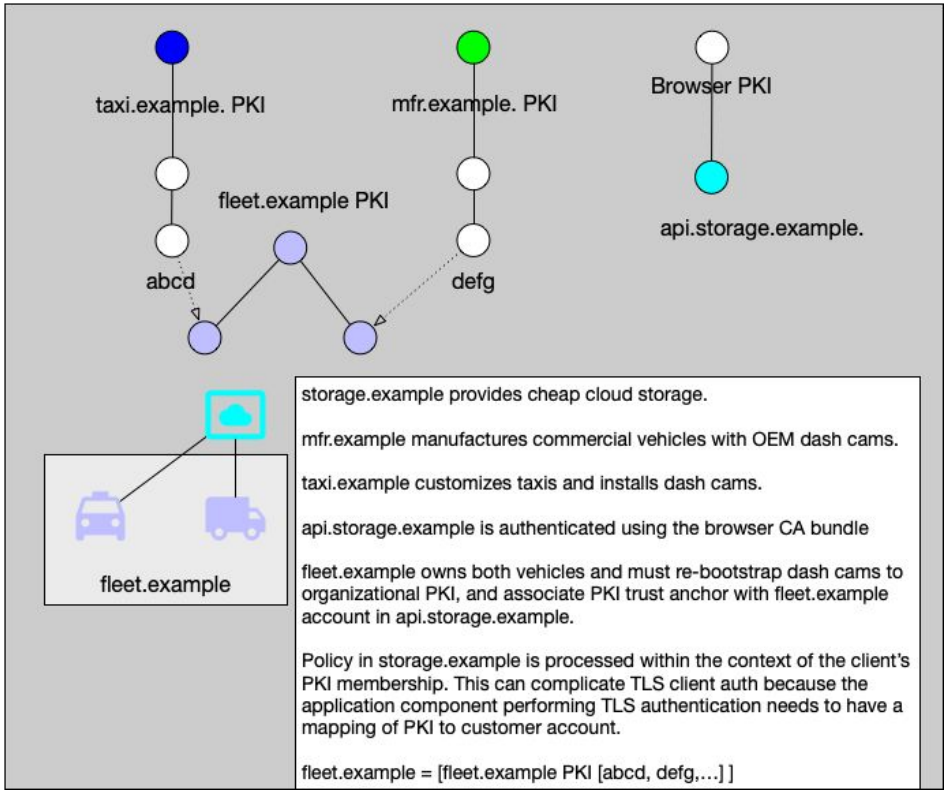
Challenges:

- Onboarding all mfr CA certs is a manual process
- Authz is complicated by different PKI naming conventions
- Prevent cross-CA impersonation

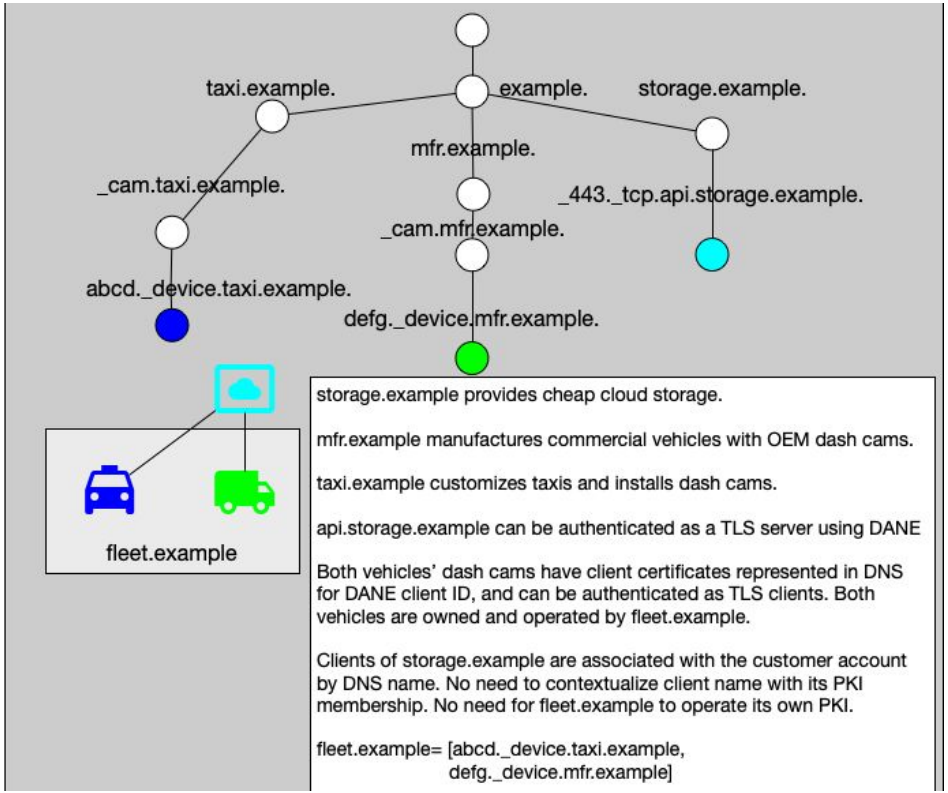
With DANE Client ID:

- Client name is bound to DNS name
- No need to onboard devices to organizational PKI
- Authenticating middleware does not need to reference many private PKI trust anchors
- Dane ClientID can be represented by authenticating middleware using HTTP headers

Cloud Service Authentication: Traditional PKI



Cloud Service Authentication: DANE TLS



Scope of Work

DANISH Core Objectives

DANE for Client Identity:

https://tools.ietf.org/html/draft-huque-tls-dane-clientid-04

https://tools.ietf.org/html/draft-huque-dane-client-cert-05

DNS-Based Identity with DANE Architecture Document:

TBD

Appendix

DANE operational guidance:
https://tools.ietf.org/html/rfc7671

DANE for Client Identity:
https://tools.ietf.org/html/draft-huque-tls-dane-clientid-04
https://tools.ietf.org/html/draft-huque-dane-client-cert-05

DANE for certificate discovery:
https://github.com/ashdwilson/dane-pkix-cd

Internet X.509 PKI Certificate and CRL Profiles:
https://tools.ietf.org/html/rfc5280

DNSSEC Chain Extension:
https://tools.ietf.org/html/draft-dukhovni-tls-dnssec-chain-02

EAP-TLS:
https://tools.ietf.org/html/rfc5216

Proxy Headers:
https://tools.ietf.org/html/rfc7239
https://tools.ietf.org/html/draft-bdc-something-something-certificate-04
https://www.haproxy.com/blog/haproxy/proxy-protocol/