Attestation in TLS with the Entity Attestation Token

<draft-tschofenig-tls-cwt>

History

- SSL/TLS was initially used with server-side authentication using X.509 certificates, as needed by the web.
- Over time TLS became the choice of protocol developers for securing many applications, not just web applications.
  - Client-side authentication also became popular.
  - E.g.: IoT, server-to-server communication
- With RFC 7250 the ability to use raw public keys in TLS was introduced
  - Motivated by the reduction of the handshake size.
  - It opened the door for the use of key types other than X.509 certificates (e.g. IEEE 1609.2 certificates).

RFC 7250 “Using Raw Public Keys in TLS/DTLS”

- In TLS, the ClientHello and ServerHello messages are used to negotiate support for “features”.
  - TLS 1.3 allows the EncryptedExtensions to convey information that was previously found in the ServerHello.
- Two extensions allow the client and the server to indicate support for alternative certificate types:
  - Client_certificate_type in the ClientHello indicates the certificate type the client is able to provide to the server.
  - Server_certificate_type in the ClientHello indicates the certificate type the client is able to process when sent from the server.
  - These extensions are echoed in the ServerHello, if supported by the server.

Example: TLS server uses a raw public key

```
client_hello,
server_certificate_type=(RawPublicKey) // (1)
                        ->
                        <- server_hello,
                           server_certificate_type=RawPublicKey, // (2)
                           certificate, // (3)
                           server_key_exchange,
                           server_hello_done

client_key_exchange,
change_cipher_spec,
finished                ->

                        <- change_cipher_spec,
                           finished

Application Data        <-------> Application Data
```

TLS-CWT

draft-tschofenig-tls-cwt

- Adds CWTs to the RFC 7250-created IANA “TLS Certificate Types” registry.
- Draft (currently) requires CWTs with proof-of-possession (PoP) keys.
- PoP tokens introduced with RFC 8747.
  - Uses the cnf claim to carry a COSE_Key.
  - Symmetric and asymmetric COSE_Keys can be used.
- *EAT token could, however, also be used (although not described).*

```
{
/iss/ 1 : "coaps://server.example.com",
/aud/ 3 : "coaps://client.example.org",
/exp/ 4 : 1879067471,
/cnf/ 8 :{
  /COSE_Key/ 1 :{
    /kty/ 1 : /EC2/ 2,
    /crv/ -1 : /P-256/ 1,
    /x/ -2 : h'd7cc072de2205bdc1537a543
             d53c60a6acb62eccd890c7fa
             27c9e354089bbe13',
    /y/ -3 : h'f95e1d4b851a2cc80fff87d8
             e23f22afb725d535e515d020
             731e79a3b4e47120'
  }
 }
}
```

PoP Token example (without signature)

Relationship to Attestation

- An Entity Attestation Token, draft-ietf-rats-eat, is a CWT (or a JWT/UCS) with many claims related to attestation defined.
  - ~ 25 claims defined.
- Reference implementation of a CWT in https://github.com/laurencelundblade/ctoken/
  - Utilizes QCBOR (see https://github.com/laurencelundblade/QCBOR) for the CBOR implementation.
  - Relies on t_cose (see https://github.com/laurencelundblade/t_cose) for the COSE implementation.
  - t_cose offers flexible crypto backends with OpenSSL and the PSA Crypto API.

Example: TLS client uses a CWT and TLS server uses X.509

```
client_hello,
server_certificate_type=(X.509)
client_certificate_type=(CWT) // (1)
                          ->
                          <-  server_hello,
                              server_certificate_type=X.509 // (2)
                              certificate, // (3)
                              client_certificate_type=CWT // (4)
                              certificate_request, // (5)
                              server_key_exchange,
                              server_hello_done
certificate, // (6)
client_key_exchange,
change_cipher_spec,
finished                   ->

                           <- change_cipher_spec,
                              finished

Application Data        <------->     Application Data
```

Next Steps

- Interest in attestation increased with the work on confidential computing.
- There is some implementation work to do.
  - Mbed TLS library contains an implementation of TLS 1.3 (since end of 2021).
  - Arm has contributed to t_cose (supporting digital signatures, and encryption*)
  - Planning to prototype TLS-CWT in Mbed TLS (and to advance the TLS-CWT specification).
- Welcome involvement from others (specification writing, prototyping).

(*) Encryption support in t_cose is work in progress at https://github.com/laurencelundblade/t_cose/pull/46