Conceptual Message Wrappers

CCC Attestation SIG

- Significant diversity in attestation-related data formats (TPM, EAT, (Co)SWID, CoRIM, proprietary)
- Not all parties need to understand everything. E.g.:
  - RPs in background-check topologies move Evidence from Attesters to Verifiers
  - Attesters in passport mode forward Attestation Results between Verifiers and RPs

Single wrapper for transporting *any* attestation messages in *any* "hosting" protocol (e.g., TLS, cert enrollment, Verifier APIs, etc.)

Design choices

- Simple "type & value" container
- JSON and CBOR representations
- Typing based on Media Types [RFC6838]

Why Media Types

- (Relatively) cheap registration process
- Standard, vendor (`application/vnd.*`) and "vanity" (`application/prs.*`) sub-trees
- Compressed representations also available using CoAP Content-Formats [RFC7252] and CBOR Tags [RFC9277]
- Reusable (e.g., REST API) - possibly already available (e.g., EAT)

Formats

Record (JSON & CBOR)

```
[
  type:  Media Type / Content Format
  value: CBOR byte string / Base64 URL-encoded
  ? ind: "Conceptual Messages" bitmap
]
```

Tag (CBOR only)

```
#6.<type>(value: CBOR byte string)
```

Examples

JSON Record

```
[
  "application/vnd.example.rats-conceptual-msg",
  "q82rzQ"
]
```

CBOR Record

```
82                  # array(2)
   19 7531          # unsigned(30001)
   44               # bytes(4)
      2347da55      # "#G\xDAU"
```

CBOR Tag

```
da 63747632     # tag(1668576818)
   44           # bytes(4)
      2347da55  # "#G\xDAU"
```

Collections

- Based on Simon's design for an "EAT Collection" container (in fact, binary compatible)
- Allow grouping multiple different "named" CMWs (composite attester)

```
{
  ? Collection identifier (URI / OID)

  + Label => CMW / "tunnel"
}
```

Examples

Homogeneous

```
{
  "attester A": [
    "application/eat-ucs+json",
    "e30K",
    4
  ],
  "attester B": [
    "application/eat-ucs+cbor",
    "oA",
    4
  ]
}
```

Tunnelled

```
{
  "attester A": [
    "application/eat-ucs+json",
    "e30K",
    4
  ],
  "attester B (tunnelled)": [
    "#cmw-c2j-tunnel",
    "g3gYYXBwbGljYXRpb24vZWF0LXVjcytjYm9yQaAE"
  ]
}
```

```
$ echo -n g3gYYXBwbGljYXRpb24vZWF0LXVjcytjYm9yQaAE | base64 -d | cbor2diag.rb
["application/eat-ucs+cbor", h'A0', 4]
```

Collection sealing

External integrity protection

Using the "cmw" claim in a CWT/JWT

```
{
  "cmw": {
    "cpu.0": [
      "application/vnd.A",
      "..."
    ],
    "gpu.0": [
      "application/vnd.B",
      "..."
    ]
  },
  "iss": "ecd v0.0.1",
  "exp": 2024129268,
  "eat_profile": "tag:github.com,2024:deeglaze/ecd"
}
```

Wrapping the collection in a COSE_Sign1

```
[
  / protected / h'a10126',
  / unprotected / {},
  / payload (CMW collection) / << {
    "attester A": [
      30001,
      h'2347da55',
      4
    ],
    "attester B": 1668576818(h'2347da55')
  } >>,
  / signature / h'...'
]
```

Carrying the collecion in an X.509 extension

```
-- CMW Extension OID

id-pe-cmw-collection  OBJECT IDENTIFIER  ::=
   { iso(1) identified-organization(3) dod(6) internet(1)
     security(5) mechanisms(5) pkix(7) id-pe(1) TBD }

-- CMW Extension Syntax

CMW ::= CHOICE {
    json UTF8String,
    cbor OCTET STRING
}
```

Collection sealing

Intra-collection locking

Hash locking between adjacent CMWs

```
cab = {
  "kat": [ "application/eat+cwt", bytes .cbor cose_sign1<kat> ]
  "pat": [ "application/eat+cwt", bytes .cbor cose_sign1<pat> ]

  "__cmwc_t": "tag:ietf.org,2024-02-29:rats/kat"
}
```

where:

```
pat.eat_nonce = hash(kat.kak-pub)
```

Links

IETF Datatracker

Editor copy

Issue tracker

Editors

RATS ML

FIN