concise-ta-stores (CoTS)

IETF 114 – Philadelphia – July 2022 – RATS Working Group

Russ Housley

Carl Wallace

Draft and example implementation

- https://datatracker.ietf.org/doc/html/draft-wallace-rats-concise-ta-stores-00
  - https://github.com/carl-wallace/draft-wallace-rats-concise-ta-stores
- https://github.com/carl-wallace/corim
  - Fork of https://github.com/veraison/corim

Desired path forward

- Accept as working group draft and proceed on standards track

Why define concise-ta-stores?

- Current RATS work implies use of trust anchors for many different purposes, including verification of evidence, endorsements, reference values, digital letters of approval, public key certificates, revocation information, etc.
- The concise-ta-stores spec provides a means of representing trust anchors with limitations on the contexts in which a trust anchor store may be used
- Support various combinations of TAs and CAs, i.e., single vendor TA/CA, multiple vendor TA/single vendor CA, multiple vendor TA/multi-vendor CA

Why define as an extension of CoRIM?

- Similar general purpose of conveying information to verifiers and relying parties
- Why not define as a profile of CoRIM?
  - The lifecycle of TAs and CAs is different than the lifecycle of reference data
  - The use cases for trust anchors in RATS are broader than CoRIM
  - The verification-map in CoRIM is tied to CoMIDs, leaving no easy path to support non-CoMID-centric use cases

Basic structure

```
concise-ta-store-map = {
    ? tastore.language => language-type
    ? tastore.store-identity => tag-identity-map
    tastore.environments => environment-group-list
    ? tastore.purposes => [+ $tas-list-purpose]
    ? tastore.perm_claims => [+ $$claims-set-claims]
    ? tastore.excl_claims => [+ $$claims-set-claims]
    tastore.keys => cas-and-tas-map
  }
```

- concise-ta-stores are arrays of the concise-ta-store-map, which defines a trust anchor (TA) store
- Each TA store may be defined with optional constraints
- Optional store-identity facilitates linking from other artifacts
- Each TA store contains at least one TA, which may also optionally constrained

Basic structure: store identity

```
tag-identity-map = {
      &(tag-id: 0) => $tag-id-type-choice
      ? &(tag-version: 1) => tag-version-type
}
```

- Defined in CoRIM.
- Allows for identifying a store using a UUID or textual identifier with an optional version value

Basic structure: environments

```
environment-group-list-map = {
    ? tastore.environment_map => environment-map,
    ? tastore.concise_swid_tag => abbreviated-swid-tag,
    ? tastore.named_ta_store => named-ta-store,
  }
```

- environment-map is from CoRIM. Features class, instance, and group.
- abbreviated-swid-tag is modified from CoSWID to allow all fields except entity to be optional.
- named-ta-store is freeform text name

Basic structure: constraints

```
$$tas-list-purpose /= "cots"
$$tas-list-purpose /= "corim"
$$tas-list-purpose /= "coswid"
$$tas-list-purpose /= "eat"
$$tas-list-purpose /= "key-attestation"
$$tas-list-purpose /= "certificate"
$$tas-list-purpose /= "dloa"
```

- The purpose field is similar to the PKIX extended key usage extension
- Represents constraints as abstract names, i.e., corim, eat, dloa, etc.
  - Corresponding EKU values will be defined for use in certificates
  - Will propose a registry for purpose values

Basic structure: constraints (cont.)

```
? tastore.perm_claims => [+ $$claims-set-claims]
? tastore.excl_claims => [+ $$claims-set-claims]
```

- The perm_claims and excl_claims fields can carry EAT claims to represent acceptable or unacceptable values for associated TA(s)
- $$claims-set-claims is a group socket defined in EAT
  - Claims are registered in the CBOR Web Token (CWT) Claims registry: http://www.iana.org/assignments/cwt

Basic structure: keys

```
trust-anchor = {
   format => $pkix-ta-type
   data => bstr
}
cas-and-tas-map = {
   tastore.tas => [ + trust-anchor ]
   ? tastore.cas => [ + pkix-cert-data ]
}
```

- Provides means to convey trust anchors and, optionally, intermediate CA certificates
- TAs can be represented as bare public key (i.e., SubjectPublicKeyInfo), a Certificate, or a TrustAnchorInfo
  - TrustAnchorInfo allows for per-trust anchor constraints, which would be in addition to any TA store constraints

Security mechanisms

- Inherits signed structure from CoRIM, which uses from COSE
- Recommend verification to a trust anchor with the CoTS purpose

Things left to other specifications

- Use of constraints represented in a TA store definition or TA definition is not covered in this specification

Questions

1. Is CoRIM extension the right way forward?
2. Should environment be simplified to focus on some identity characteristics shared by CoMID/CoSWID?
3. Do constraints mechanisms adequately cover the TA landscape implied by RATS architecture?
4. Other questions…