Entity Attestation Token
draft-mandyam-rats-eat-00
(draft-mandyam-eat-01)

Laurence Lundblade

March 2019
Entity Attestation Token

- Chip & device manufacturer
- Device ID (e.g. serial number)
- Boot state, debug state...
- Firmware, OS & app names and versions
- Geographic location
- Measurement, rooting & malware detection...

All Are Optional

Cryptographically secured by signing
EAT Overall System

Entity Manufacturer (e.g. chip or device vendor)

Interaction to obtain public key and related data for token verification.

Device status & characteristics determination

EAT Target for standardization

EAT Token

Key ID or Cert
- Nonce
- Claim 1
- Claim 2
- ...
Signature

EntityManager

Entity (e.g., Chip, Device…)

Immutable private key for signing. Stored securely on device

Claims

Token creation & signing

Nonce

Relying Party (e.g., Server / Service)

Signature and public key verification process

Claims

Interaction to obtain public key and related data for token verification.

Device status & characteristics determination

Manufacturing process to put seed, private and/or public key, cert or other on device (this is intentionally open-ended)
**EAT Format (basically CWT)**
draft-mandyam-eat-01

<table>
<thead>
<tr>
<th><strong>Overall structure: COSE_Sign1</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Algorithm</strong> -- Examples: ECDSA 256, RSA 2048, ECDAA</td>
</tr>
<tr>
<td><strong>Signing Scheme</strong> -- Examples: IEEE IDevID, EPID, X.509 Hierarchy</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Key ID</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>-- identifies the key needed to verify signature</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Certs (optional)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>-- to chain up to a root for some signing schemes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Signed payload</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>• CBOR formatted map of claims that describe device and its disposition</td>
</tr>
<tr>
<td>• Few and simple or many, complex, nested...</td>
</tr>
<tr>
<td>• All claims are optional -- no minimal set</td>
</tr>
<tr>
<td>• The format and meaning of a basic set of claims should be standardized for interoperability</td>
</tr>
<tr>
<td>• Should be adaptable to cover many different use cases from tiny IoT devices to complex mobile phones</td>
</tr>
<tr>
<td>• Privacy issues must be taken into account</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>signature</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>-- Examples: 64 byte ECDSA signature, 256 byte RSA signature</td>
</tr>
</tbody>
</table>

- COSE format for signing
- Small message size for IoT
- Allows for varying signing algorithms, carries headers, sets overall format
- CBOR format for claims
- Small message size for IoT
- Labelling of claims
- Very flexible data types for all kinds of different claims.
- Translates to JSON
- Signature proves device and claims (critical)
- Accommodate different end-end signing schemes because of device manufacturing issues
- Privacy requirements also drive variance in signing schemes
COSE Signing Scheme Flexibility

EAT does not define any signing schemes, key types or such so the claims it defines can be used with lots of signing schemes

Claims and signing schemes are orthogonal

- Many standard algorithms already supported
  - RSA, ECDSA and Edwards-Curve Signing (public key)
  - HMAC and AES-based MACs (symmetric key)

- Extensible for future algorithms
  - IANA registry for algorithms exists today

- Extensible for special case schemes
  - Proprietary simple HMACs schemes, perhaps HW based
  - Possibly Intel EPID
  - (non-standard algorithms will of course be less interoperable)
Example Token

COSE binary ~130 bytes including sig

COSE ECDSA signing overhead is about 87 bytes: 23 for headers and structure, 64 bytes for ECDSA sig

JSON text ~500 bytes including a JOSE sig

Payload Translated to JSON

- Integer labels mapped to strings
- Binary data base 64 encoded
- Floating point numbers turned into strings

```json
{
    "UEID": "kyf9d98Mk979077L38Uw34kKFRHJgd18f==",
    "secureBoot": true,
    "debugDisable": true,

    "integrity": {
        "status": true,
        "timestamp": "2015-10-5T05:09:04Z",
    },

    "location": {
        "lat": "32.9024843386",
        "long": "-117.192956976"
    }
}
```
Privacy

• Entity Attestation Tokens are intended for many use cases with varying privacy requirements
  ◦ Some will be simple with only 2 or 3 claims, others may have 100 claims
  ◦ Simple, single-use IoT devices, have fewer privacy issues and may be able to include claims that complex devices like Android phones cannot

• Options for handling privacy
  ◦ Omit privacy-violating claims
  ◦ Redesign claims especially to work with privacy regulation
  ◦ Obtain user permission to include claims that would otherwise be privacy-violating

• Some signing schemes will be privacy-preserving (e.g. group key, ECDAA) and some will not (e.g., per-device ECDSA signing key).
EAT Defines an Initial Set of Claims

<table>
<thead>
<tr>
<th>Claim</th>
<th>Description</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>U Eid</td>
<td>Identify a particular individual device, similar to a serial number</td>
<td>Basic</td>
</tr>
<tr>
<td>OEM ID</td>
<td>Identify the manufacturer of the device</td>
<td>Basic</td>
</tr>
<tr>
<td>Boot and debug state</td>
<td>Is secure/trusted/authenticated boot turned on? Is debug disabled?</td>
<td>Basic</td>
</tr>
<tr>
<td>Geographic location</td>
<td>GPS coordinates, speed, altitude</td>
<td>Basic</td>
</tr>
<tr>
<td>Security level</td>
<td>Rich OS, TEE, secure element...</td>
<td>Basic</td>
</tr>
<tr>
<td>Nonce</td>
<td>Token freshness</td>
<td>Basic</td>
</tr>
<tr>
<td>Origination</td>
<td>Identifies authority that can verify the token</td>
<td>Basic</td>
</tr>
<tr>
<td>Time stamp</td>
<td>Time and / or age of the token</td>
<td>Basic</td>
</tr>
<tr>
<td>Submodules</td>
<td>How to deal with claims from different subcomponents of a module. For example, the TEE and Rich OS are separate submodules.</td>
<td>Submods</td>
</tr>
<tr>
<td>Nested tokens</td>
<td>Putting one EAT inside another as a way of handling subcomponents</td>
<td>Submods</td>
</tr>
</tbody>
</table>

Intended only as initial set. Expansion should include SW components, measurement, public keys (similar to Android attestation) and other.