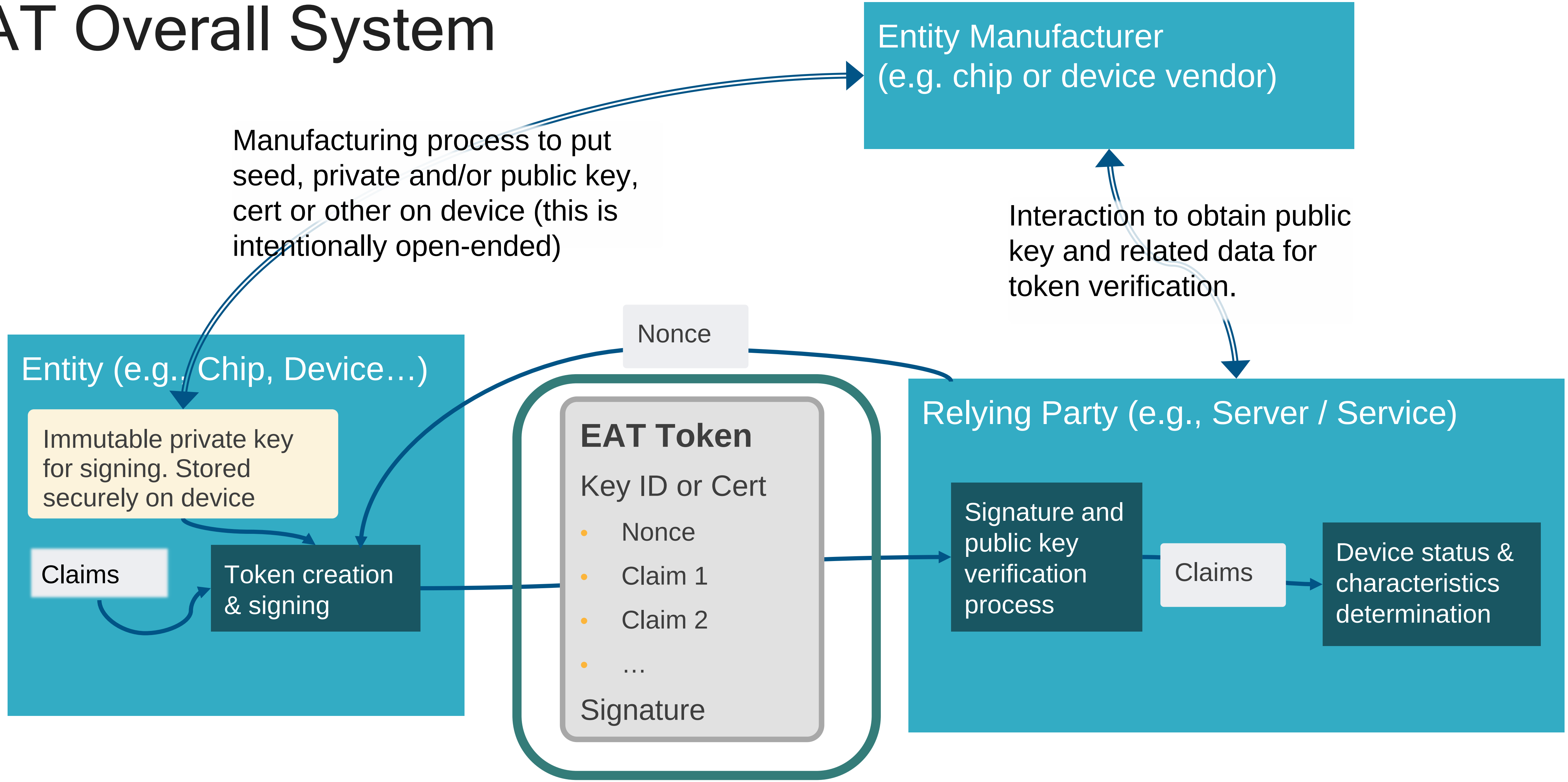
A Proposed Standard for Entity Attestation
draft-mandyam-eat-00

Laurence Lundblade

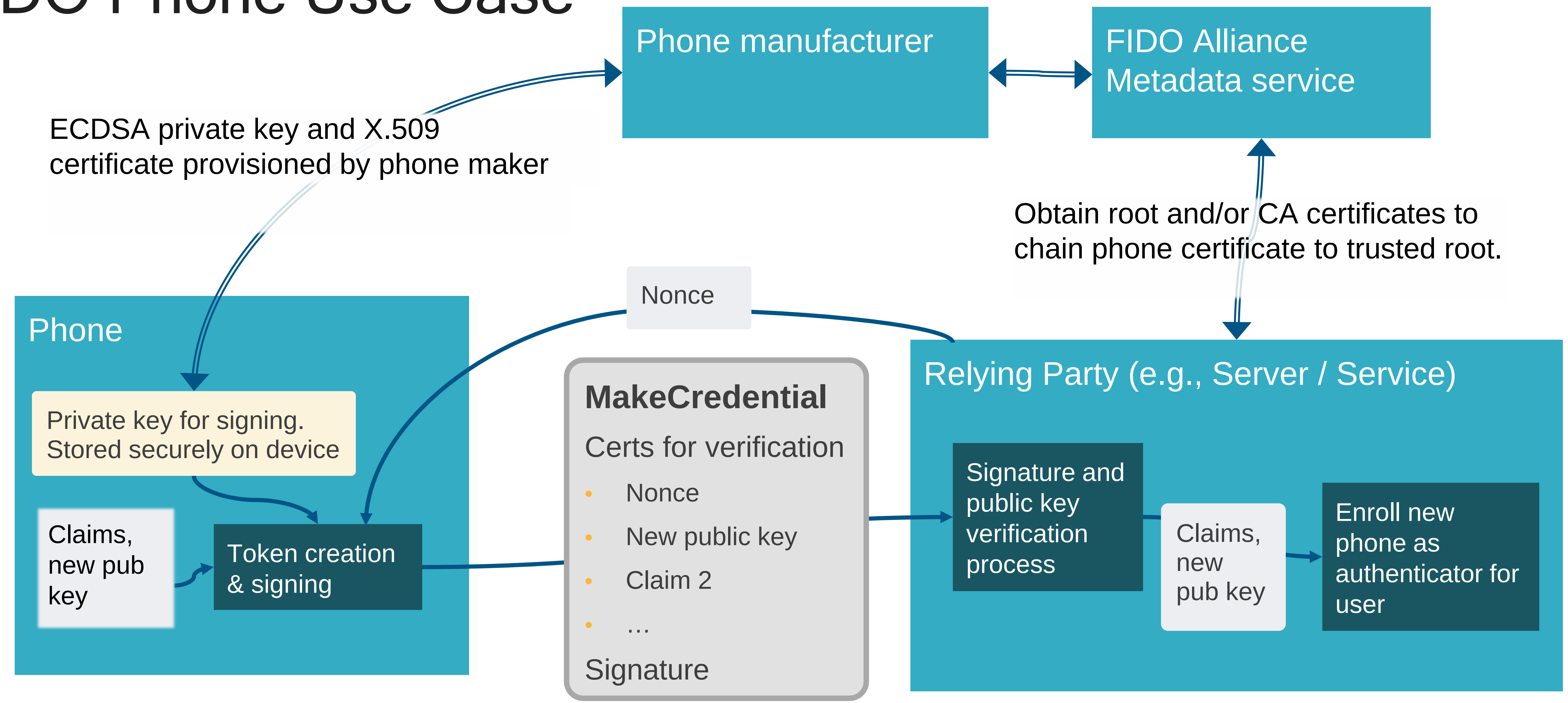
November 2018

EAT Overall System

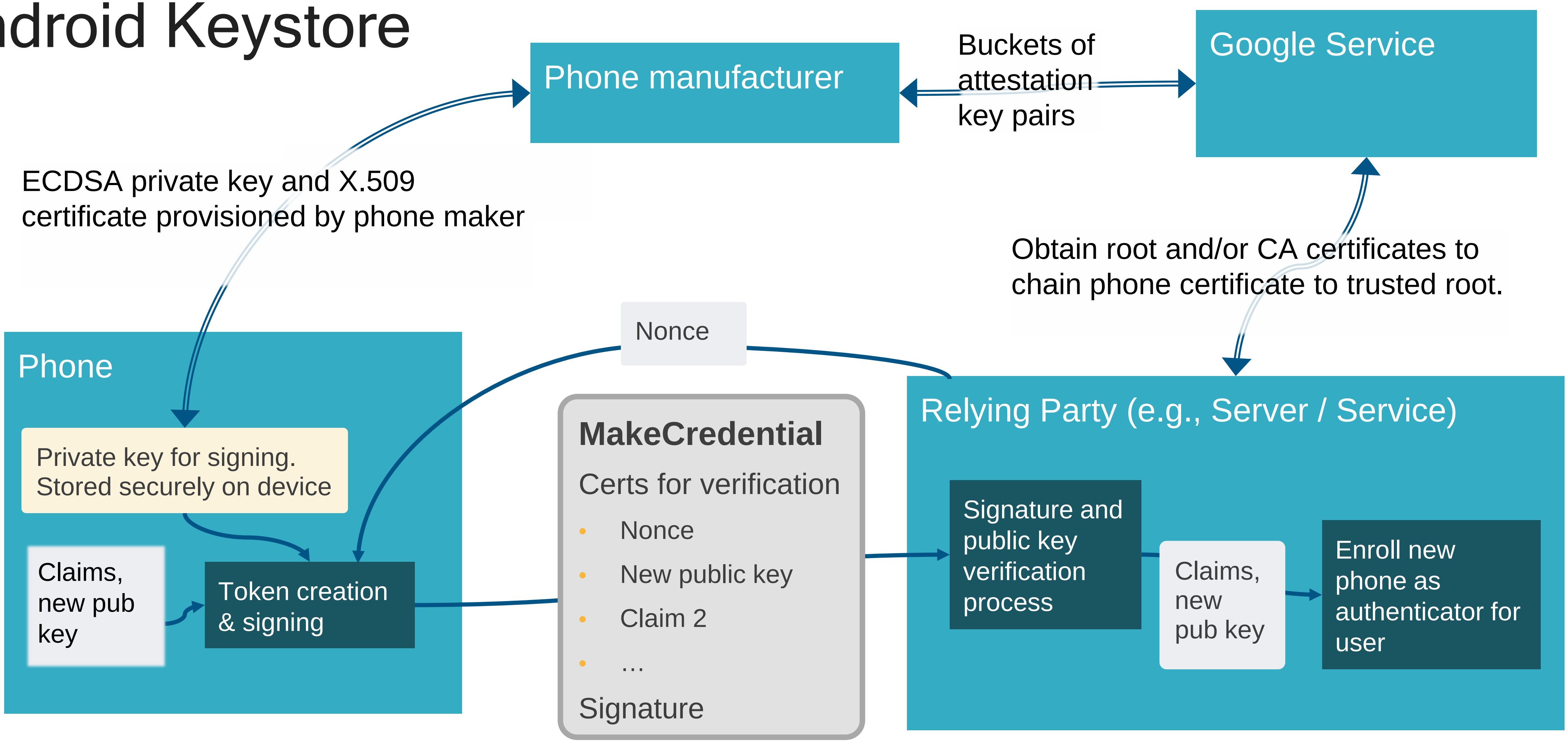


EAT Target for standardization

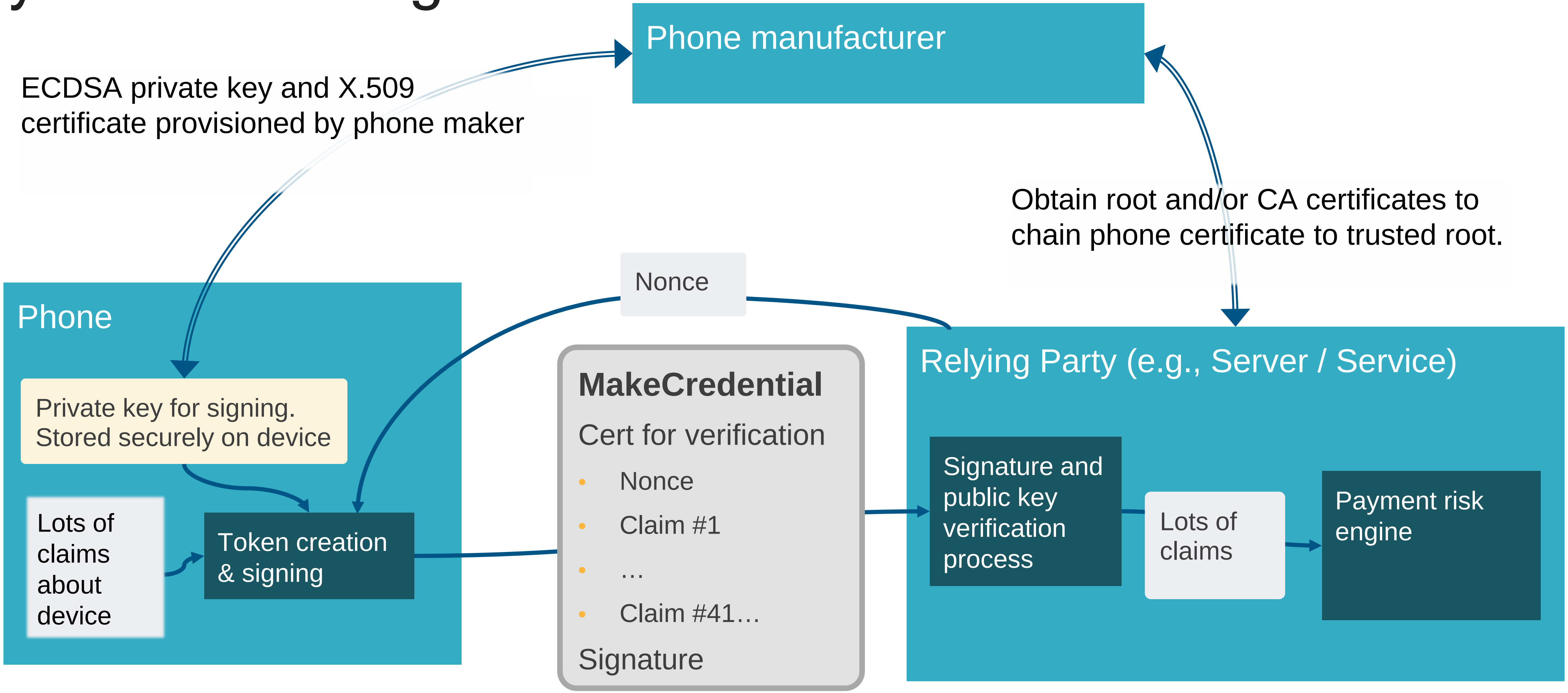
FIDO Phone Use Case



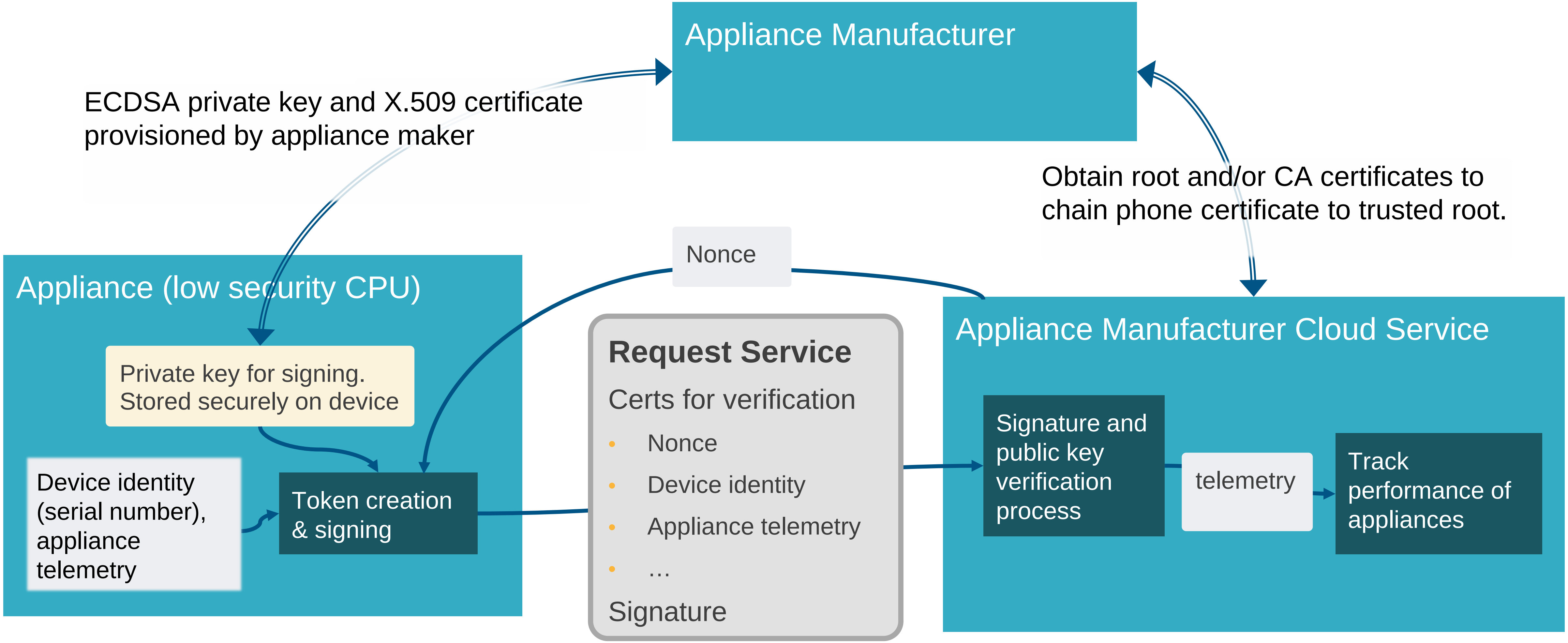
Android Keystore



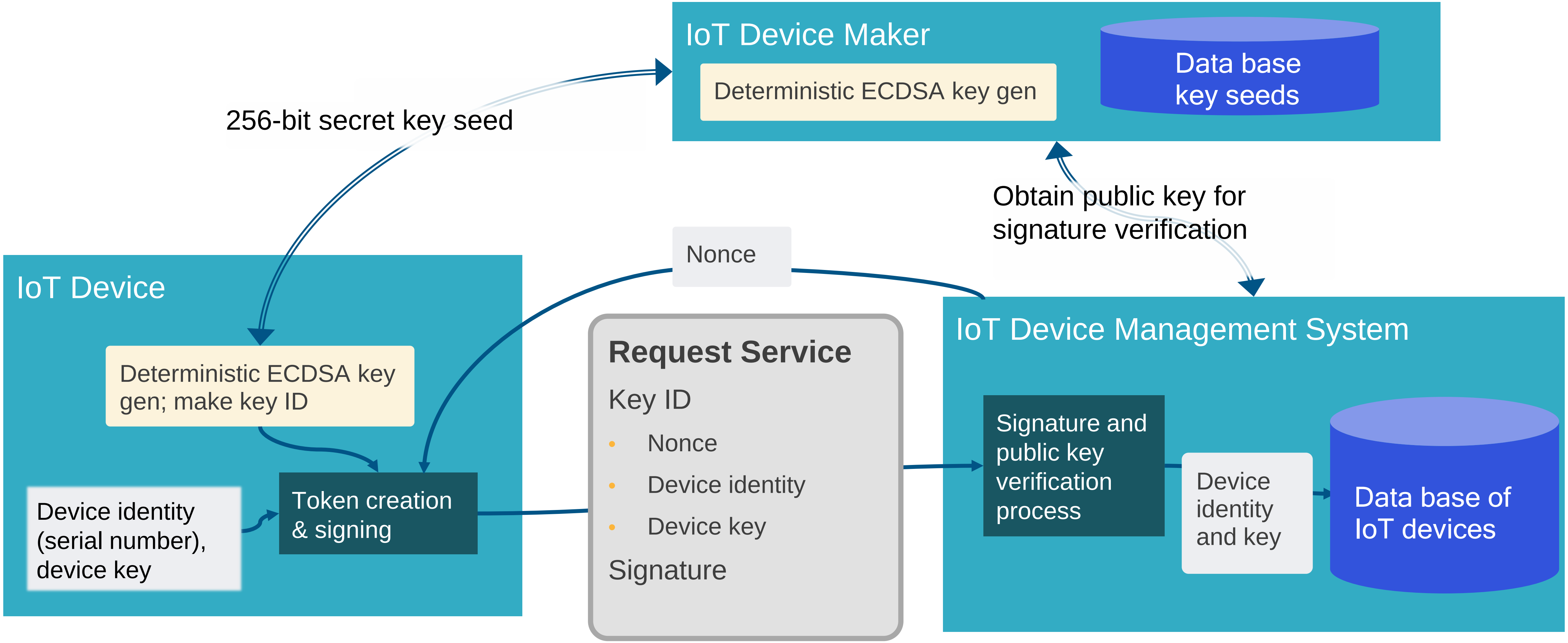
Payment Risk Engine



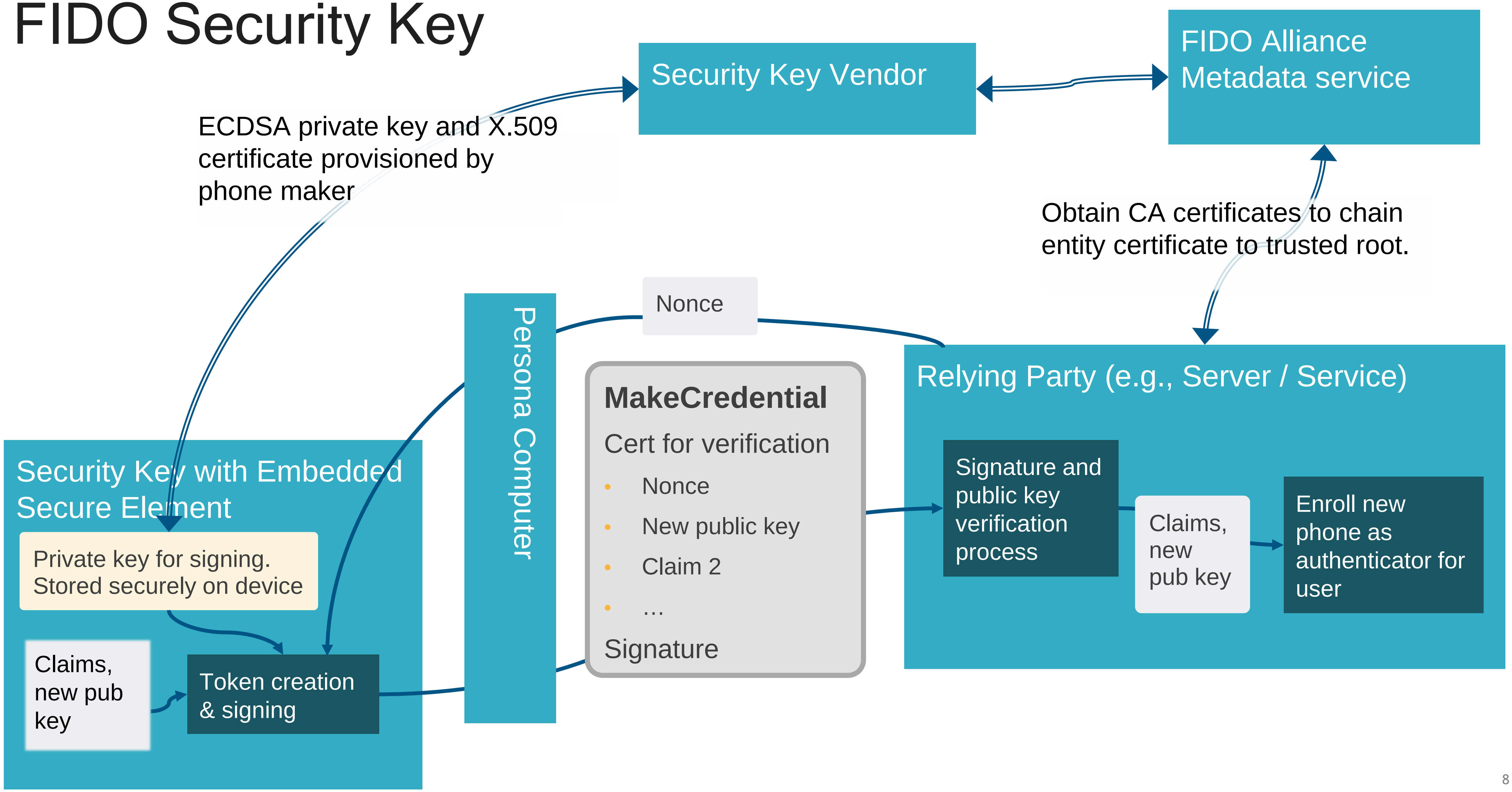
Home Appliance & Web Service



Enrollment of Low Cost IoT for Device Management



FIDO Security Key



Primary Standardization Goal is Semantic Interoperability of Claims

- Main types of claims to standardize:
  - Device Identity
  - Measurement
  - Device boot, debug and configuration state
  - Measurement and run time integrity checks
  - Geographic location
  - Device SW and HW versions
  - Public key created on the device – Keystore, IoT and FIDO use cases
- Claims should be generally applicable:
  - Not specific to TPM, TrustZone, SGX, Secure Element…
  - Not require any particular level of device security
    - Works with high-security device like Secure Elements and TPMs and low-security devices with nothing special at all.

EAT Format (basically CWT)

draft-mandyam-eat-00

Overall structure: COSE_Sign1

protected headers

Algorithm -- Examples: ECDSA 256, RSA 2048, ECDAA

Signing Scheme -- Examples: IEEE IDevID, EPID, X.509 Hierarchy

unprotected headers

Key ID -- identifies the key needed to verify signature

Certs (optional) -- to chain up to a root for some signing schemes

Signed payload

- CBOR formatted map of claims that describe device and its disposition
- Few and simple or many, complex, nested…
- All claims are optional -- no minimal set
- The format and meaning of a basic set of claims should be standardized for interoperability
- Should be adaptable to cover many different use cases from tiny IoT devices to complex mobile phones
- Privacy issues must be taken into account

sig

signature -- Examples: 64 byte ECDSA signature, 256 byte RSA signature

---

- 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

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

CBOR diagnostic representation of binary data of full signed token

```
[
  / protected / << {
    / alg / 1: -7 / ECDSA 256 /
  } >>,
  / unprotected / {
    / kid / 4: h'4173796d6d65747269634543445341323536’
  },
  / payload / << {
    / UEID / 8: h'5427c1ff28d23fbad1f29c4c7c6a55’,
    / secure boot enabled / 13: true
    / debug disabled / 15: true
    / integrity / -81000: {
       / status / -81001: true
       / timestamp / 21: 1444064944,
    },
    / location / 18: {
       / lat  / 19: 32.9024843386,
       / long / 20: -117.192956976
    },
 } >>,
  / signature / h'5427c1ff28d23fbad1f29c4c7c6a555e601d6fa29f9179bc3d7438bacaca5acd08c8
                d4d4f96131680c429a01f85951ecee743a52b9b63632c57209120e1c9e30'
]
```

Payload Translated to JSON

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

```
{
   “UEID” : “k8if9d98Mk979077L38Uw34kKFRHJgd18f==”,
   “secureBoot” : true,
   “debugDisable” : true,

   “integrity”: {
       “status”: true,
       “timestamp”: “2015-10-5T05:09:04Z”,
   },
   “location”: {
       “lat”: “32.9024843386”,
       “long”: “-117.192956976”,
   },
}
```

COSE Signing Scheme Flexibility

- 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)

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