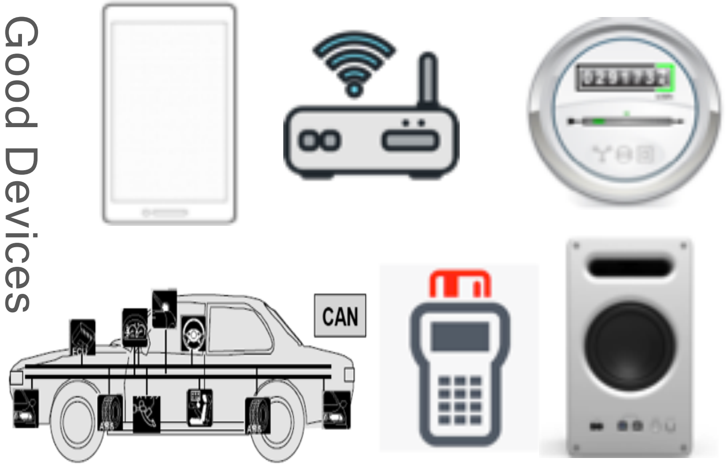
A Proposed Standard for Entity Attestation

Laurence Lundblade

July 2018

Good Devices



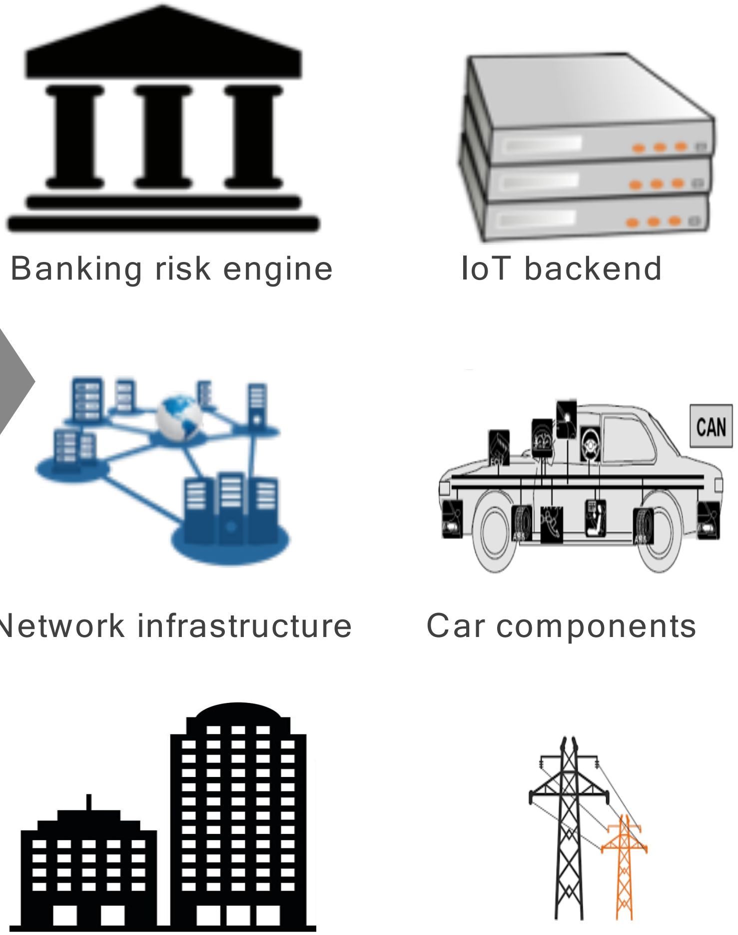
Bad Devices

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



Banking risk engine

IoT backend

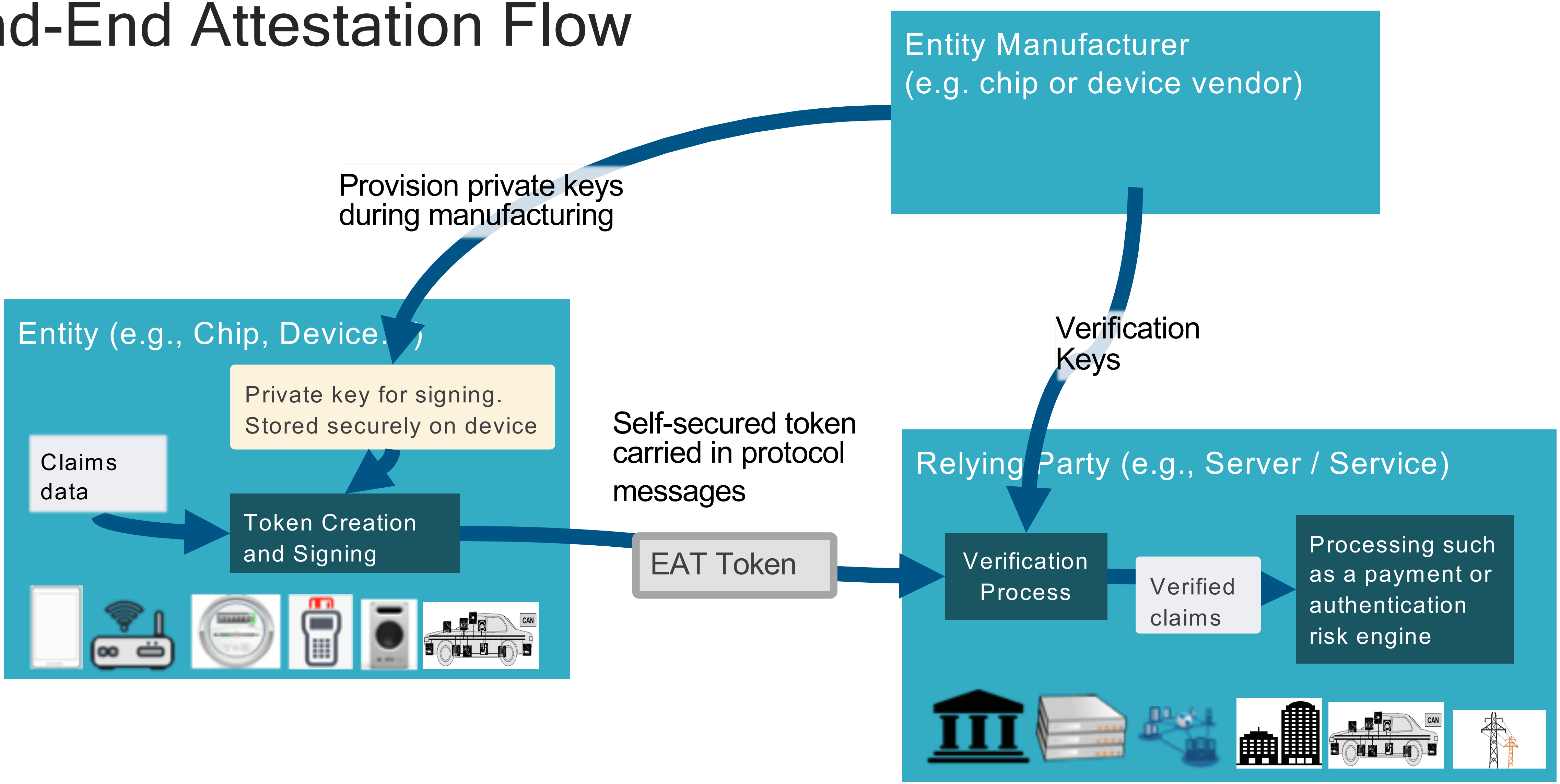
Network infrastructure

Car components

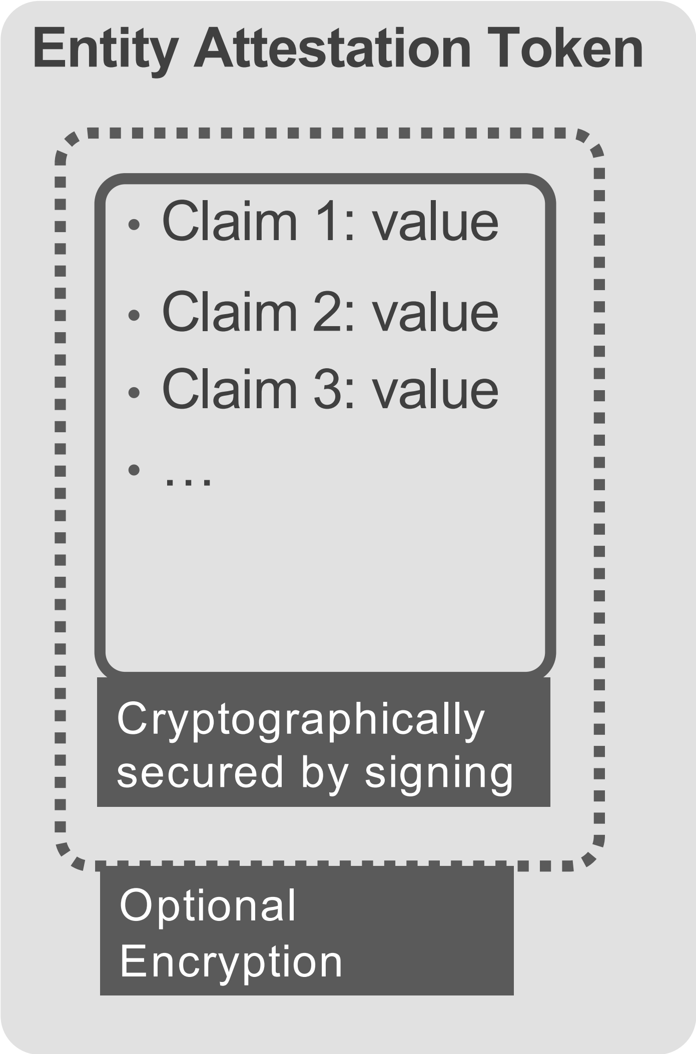
Enterprise auth risk engine

Electric company

End-End Attestation Flow



Other flows are possible where verification is done by a service or by the entity vendor.



Four Aspects of Standardization

1. General Structuring and Representation of Claims

- Labeling of claims
- Optionality of claims
- Flexible data representation – integers, strings, binary…

2. Meaning of Individual Claims

- Interoperability between devices and servers from different vendors

3. Signing Format

- Accommodate different schemes and algorithms

4. Encryption Format (optional)

- Accommodate different algorithms

EAT Format

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

Standardization / Extensibility of Claims

- Base standard describes how claims work and are formatted in general and
  - May include the most common, best agreed upon claims
- No claims will be mandatory in base standard
  - Verifiers can reject tokens missing claims required in specific use cases
  - Profile with minimum sets of claims can be defined, by industry / use case (automotive, power meter…)
- Non-standard and proprietary claims will be allowed
  - Verifiers can ignore claims they do not understand
- The bulk of standardization work will be defining claims well
  - Standardized meaning will allow verifiers to interpret claims from devices from different vendors
  - This will not always work perfectly and the meaning of some claims may be subjective
- IANA (Internet Assigned Names and Numbers) can be used to register claims to avoid collisions and duplications. Similar registries already exist (e.g. CWT and JWT registries).
- CBOR itself is extensible for new data types.

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'4173796d6d657472696345434453413235365'
  },
  / 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",
   },
}
```

Device and Submodules

- A top-level token is associated with a device – a finished commercial end product

- A device may have a set of submodules
  - Examples: WiFi subsystem, DSP subsystem
  - A submodule has a set of claims of its own
  - One level of submodules – keep it simple
  - The security of a submodule is either the same or less than that of the device

- Tokens may be nested
  - This allows submodules that have attestation keys to create their own attestations

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

Similar and Related Technologies

Technology	Use Case
FIDO Attestation	Attestation of FIDO Authenticator implementations
Android Key Store	Attestation key pairs in the key store
NEA	Collect and send endpoint security posture (e.g. anti-virus SW state and config) to enterprise collection / monitoring point
RATS / NSF	Attestation / Measurement of SW on Network Security Functions (e.g., firewalls)
TPM	Attestation / Measurement of SW running on a device
BRSKI / Zero Touch	Authenticates IoT devices for enrollment in IoT management system

More Info

- Non-WG mailing list: eat@ietf.org
- Mail list info: https://www.ietf.org/mailman/listinfo/EAT
- Draft document: https://tools.ietf.org/html/draft-mandyam-eat-00
- Github: https://github.com/eat-ietf-wg

Extra Slides Follow

Encryption Format

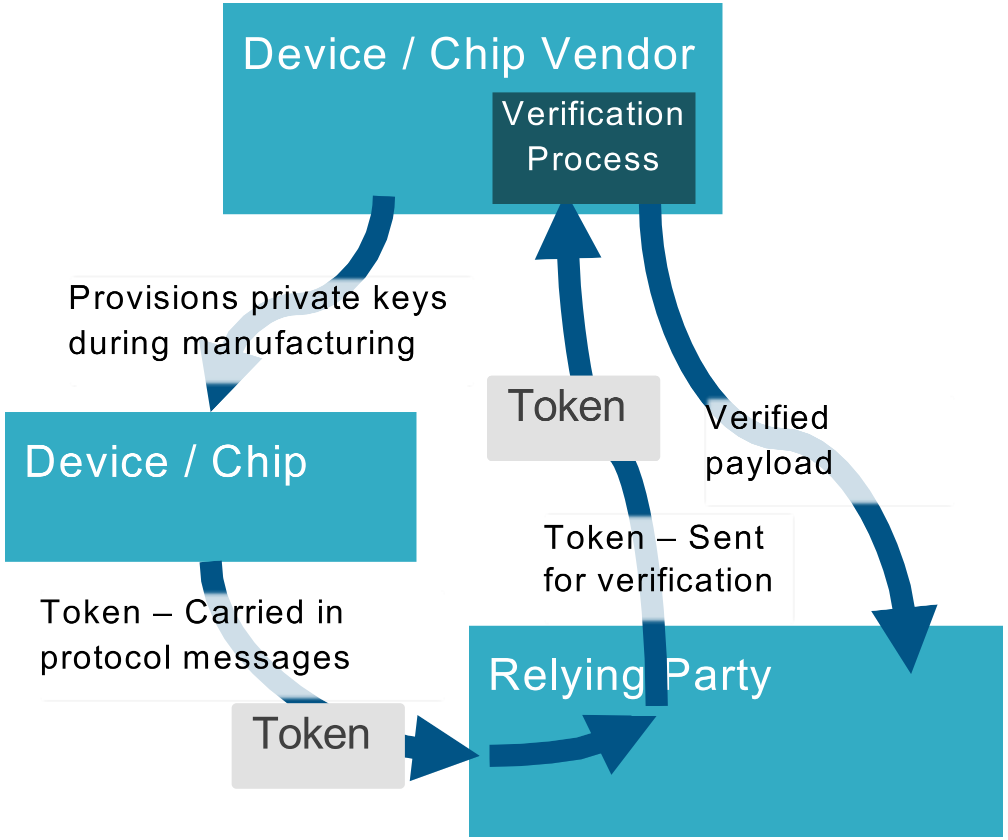
- COSE allows for signed data to be encrypted, vice versa (and even countersigning)

- CBOR encryption provides algorithm flexibility, structuring and so-on like it does for signing.

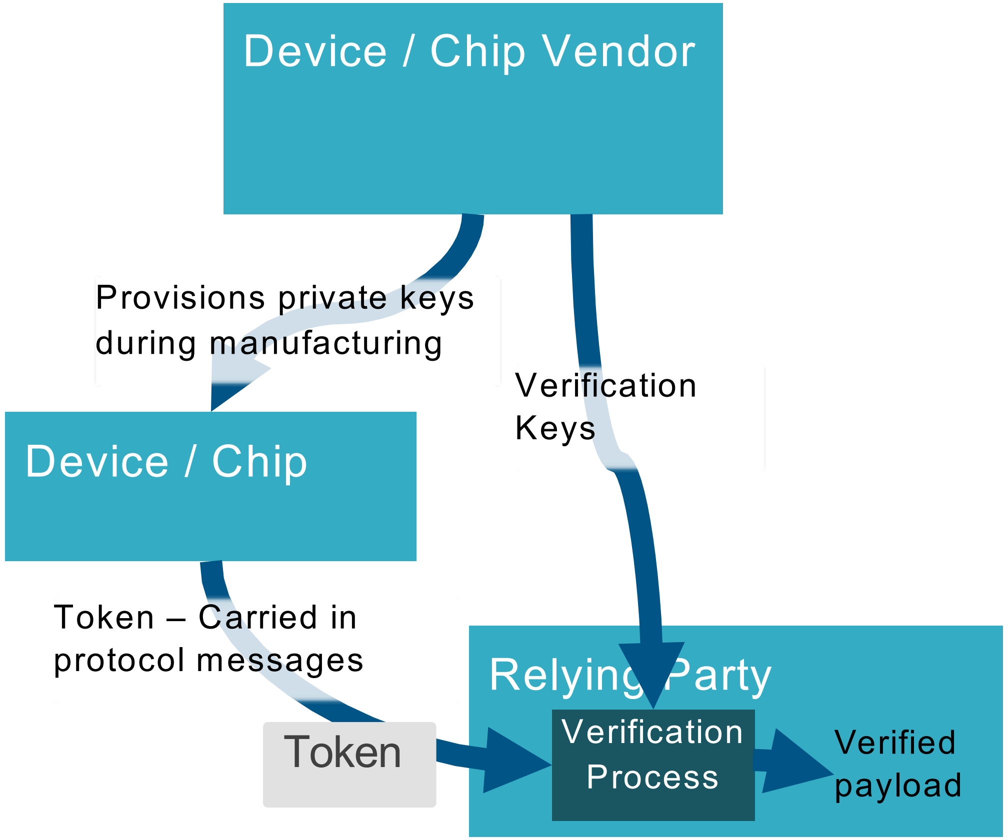
- Specifies how to combine AES symmetric encryption with EC or RSA public key

- Encryption of tokens is optional, but useful
  - Protect data that needs to be secret
  - Useful in implementing a privacy proxy
  - Monetization of an attestation service

End-end Attestation Flow - Two scenarios

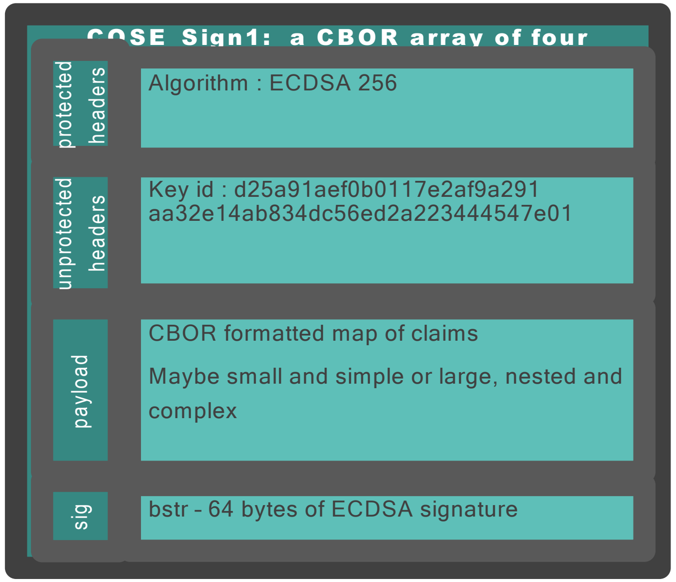


1. Device / Chip Vendor Provides a Service



2. Device / Chip Vendor Provides Keys

Signing Format



- COSE (CBOR Object Signing and Encryption) RFC 8152
- COSE is an IoT-oriented format for signing and/or encrypting a payload. It is similar to, but much simpler and more compact than PKCS #7, CMS and JOSE
- COSE signed tokens are small, self-secured data blobs that can be embedded in other protocols or written to disk…
- COSE provides structuring of payload (to-be-signed data), algorithm identification, key identification and signature
- Standard format allows use and development of standard / open source tools

General Structure & Representation of Claims

- CBOR (Concise Binary Object Representation) RFC 7049
  - Integers, text, binary, floating point numbers…
  - Aggregate types: arrays, maps of label-value pairs
  - Reasonably mature – On IETF Standards Track, RFC 7049 published in 2013
  - Open source implementations and tools available in many languages at http://cbor.io
  - Translatable to JSON by common tools
  - Compact code and data for IoT
  - Meets goal:
    - Top-level of token payload is a CBOR map of label-value pairs
    - CBOR maps easily allow for optional data
    - CBOR data types are simple & powerful – a top-level claim can be a simple integer or have a complex internal structure