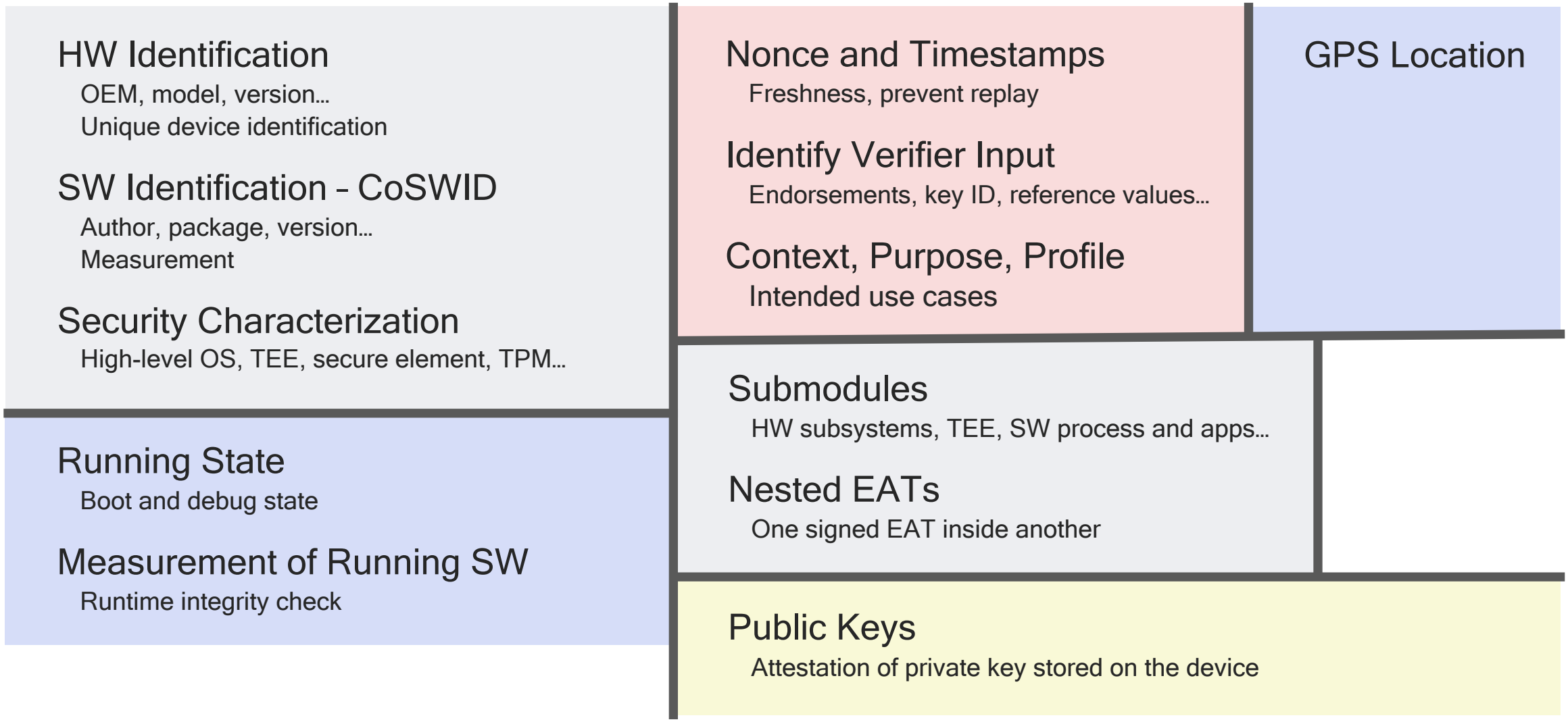
EAT Draft Status, Profiles and CoSWIDs

Laurence Lundblade

IETF 110 March 2021

Proposed Contents of an EAT – Claims



Level of Completion in EAT Draft

- Ready for last call, no open issues
- Near completion, reviewed
- Draft text
- Proposed, Interest in

Progress & change since IETF 109. Drafts -05 through -09

HW Identification

OEM, model, version…
Unique device identification

SW Identification – CoSWID

Author, package, version…
Measurement

Security Characterization

High-level OS, TEE, secure element, TPM…

Running State

Boot and debug state

Measurement of Running SW

Runtime integrity check

Nonce and Timestamps

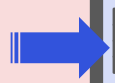
Freshness, prevent replay

Identify Verifier Input

Endorsements, key ID, reference values…

Context, Purpose, Profile

Intended use cases



GPS Location

Submodules

HW subsystems, TEE, SW process and apps…

Nested EATs

One signed EAT inside another

Public Keys

Attestation of private keys on the device (e.g., Android key store)

EAT work needed beyond claims

- Rework introduction and related with respect to RATS Architecture
  - Use Architecture terminology: “Attester”, “Verifier”…
  - Remove most of the architecture-related text currently in EAT

- Attestation Results

- More examples

- Should a verification procedure be included?

Changes since 109

- Change IMEI-based UEIDs to be 14-byte strings
- Submodules
  - Allow CWT in a JWT and vice versa; byte-string wrapping
  - CBOR tag usage in submodules
- Cryptographic keys in claims
- Added HW version claims
- Debug-related claims renamed
- Added intended use claim
- Improvements on location claim
- Added boot seed claim
- Rework CBOR interoperability section
- Added profiles section (details in following slides)

Discussed at 109, but no progress since

- Verifier Input
- Attestation Results
- Measurements

The Profile Claim

The Purpose of EAT Profiles

Achieving interoperability

- EAT describes a broad protocol with a lot of options
  - Options are necessary to accommodate use in many different environments
    - Constrained devices, reporting results between cloud services…
  - This optionality is partly inherited from:
    - COSE
    - CWT
- No round-trips in EAT so negotiation is not possible
- Two implementation of EAT will not necessarily interoperate
  - JSON vs CBOR
  - Key identification
  - Signing algorithm selection
  - CBOR encoding variants

What are EAT Profiles

- An EAT Profile is a human-readable text document
- It narrows the EAT, CWT, COSE and JWT options to result in an interoperable protocol
- Document may be:
  - IETF Standard, IETF Informational or such
  - Other standard (e.g., FIDO, GlobalPlatform)
  - Vendor proprietary (e.g., Android Attestation)
  - Private
- The Profile claim names the document
  - Either as an OID or a URI
  - The Profile claim is option (like all claims), but it is helpful for parties to know which profile is in use
  - Format of named document text and is not standardized for machine processing
- Profiles Addresses
  - Serialization format (JSON, CBOR…)
  - Protection (signing, encryption, algorithms)
  - Key identification
  - Required and prohibited claims

Serialization Requirements in a Profile

- JSON or CBOR or both
  - Also address format of nested tokens. A profile may say only CBOR tokens can be nested in a CBOR token.
- For CBOR, the following:
  - Definite / indefinite length for maps, arrays and strings
    - Suggested default is definite length
    - Constrained devices may use indefinite length
  - Whether CBOR tags are required or not

Token Protection Requirements in a Profile

- COSE and JOSE have many signing, encryption and MAC options
  - UCCS and Unsecured JWT are available

- Profile should indicate whether which is allowed/required:
  - Integrity protection: none, signing, MAC
  - Privacy protection: none pub key based encryption, symmetric key encryption

- Profile should indicate algorithms allowed / required:
  - List algorithms the Verifier must implement
  - The attester selects one

- Profile should be tight enough that interoperability is guaranteed when both Attester and Verifier implement it

Key and Endorsement Identification Requirements in a Profile

- A Verifier always requires a verification key
- A Verifier usually requires an Endorsement
- There are many ways to identify a verification key in EAT and COSE
  - COSE key ID
  - In an endorsement
  - By a claim like the UEID
  - Some other scheme

- The Profile document should make it clear how the Verifier obtains the inputs it needs to complete a verification, particularly any identifiers that come in the EAT itself.

Claims Requirements in a Profile

- All claims are optional in the EAT specification
- A Profile is likely to to require some claims to be present
  - Verification fails if they are not
- A Profile may prohibit some claims
  - Perhaps due to privacy reasons
- A Profile may describe new claims
- A Profile may allow many optional claims
  - Verification must not fail because of their presence

CoSWID Discussion

Goals for SW Description in EAT

Descriptions of SW created outside the device

- Likely signed by a SW manufacturer
- Put on the device during SW installation
- Sometimes called a manifest
- Relayed to the Verifier in an EAT claim (or an endorsement)
- May contain reference values for measurements

Descriptions of SW create on the device

- Created by code running on the device
- Typically signed as part of Attestation
- May contain measurements

- A CoSWID, possibly with extensions, can represent either
- Other formats exist too, like SUIT manifest, CoMID...
- Which should EAT support?
  - Seems like CoSWID is one
  - Perhaps others…

Proposal for CoSWID in EAT

- Must be able to carry many CoSWIDs in one EAT
- Individual CoSWID may or may not be signed and/or encrypted
- No XML SWIDs
- Signing / encryption format is COSE
- Whether they are payload of evidence is determined by examining the CoSWID

Option 1

- One claim called "coswids"
- Is an array of CoSWIDs
- Looks inside CoSWIDs to figure out that they are for

Option 2

- Describe how to include a CoSWID and let Profiles define specific claims containing CoSWIDs for specific purposes
- Similar to how public keys are handled

Option 3

- Single claim for evidence CoSWID plus single claim for payload CoSWID
- Multiple CoSWIDs via EAT submodules
  - One of each type of CoSWID per submodule

Issue 98: UEID permanence

- https://github.com/ietf-rats-wg/eat/issues/98
- FIDO IoT Onboarding spec uses GUID as device ID and maps to UEID
  - Manufacturer GUID replaced by device owner after onboarding
  - Manufacturer GUID can be restored through factory reset
- Sec. 3.4 of current text states states UEID ‘should be permanent’
- Since requirement is a ‘should’, FIDO spec may comply with spec as it stands
- Should this be clarified prior to LC?
  - Suggested text has been proposed in GH issue

Extra Slides

Discussion: EAT use for Attestation Results

- Clear interest and consensus that EATs can be used for attestation results
  - CWT, JWT and UCCS formats all useful
- EAT draft must discuss use as Attestation Results
  - Perhaps only briefly
- Many EAT claims will pass through the Verifier into Attestation Results
  - Reuse as many claims as possible
  - Don’t define new variants of EAT claims in Attestation Results
    - If existing EAT claims aren’t right for Attestation Results, let’s fix the EAT claims
- New “claims” for Attestation Results are needed
  - Overall success of verification
  - Results of checking claims against reference values
    - SW and HW version, measurements…
  - Certifications received by the Attester
  - Other?
- Should new Attestation Result claims be in EAT document or elsewhere?

Discussion: Work on Identifying Verifier Input

- Add discussion on key identification to EAT draft
  - By COSE kid
  - By COSE X509 draft (include certs, identify certs by thumbprint, URL for certs)
  - Using claims like UEID

- Add definition of COSE Header Parameters to identify Endorsements
  - Thumbprint / opaque bytes as identifier
  - URL
  - Will not define format or content type for Endorsements

- Add definition of COSE Header Parameters to identify Reference Values
  - Thumbprint / opaque bytes as identifier
  - URL
  - Will not define format or content type for Reference Values

Discussion: Measurement of Running State

- Example (e.g. Samsung TIMA)
  - TEE periodically measures high-level OS at run time
  - Results are evaluated:
    - In TEE and a claim just indicates success or failure
    - TEE sends measurements to Verifier that evaluates results

- More valuable than measurement only once at boot
  - Especially when devices run for months without a reboot in a place very far away

- Can CoSWID report measurements?

- Need new claims would be needed for reporting results evaluated by the device