

TEEP Protocol

draft-ietf-teep-protocol-11

Hannes Tschofenig, Ming Pei, David Wheeler, **Dave Thaler**, Akira
Tsukamoto

General TEEP protocol issues

#234, 251: Which TEEP messages are protected with which cipher suites

- Same negotiation cipher suite is used in both directions
 - Left to a TEEP extension if separate mechanisms are needed later
- QueryRequest now uses COSE_Sign with *both* MTI cipher suites (ES256 and EdDSA)
 - Other alternatives considered but not chosen:
 - a) Don't protect the QueryRequest
 - b) make TAM try multiple QueryRequests with different algorithms in Sign1
 - c) Agent specifies requested cipher suite in transport (e.g., HTTP headers)
 - d) use a separate TAM URI per cipher suite
- Clarified that QueryResponse is also signed using COSE_Sign1 like all later messages
- Error with ERR_UNSUPPORTED_CIPHER_SUITES is protected with one of the MTI cipher suites that the Agent supports

#245: Handling of unrecognized TEEP messages (1/2)

- **Issue:** draft-iab-protocol-maintenance argues that silently dropping invalid messages is harmful and instead one should reply with an error.
- TEEP Agent:
 - **OLD:** When the ProcessTeepMessage API is invoked, the Agent first does validation as specified in Section 4.1.2, and **drops the message if it is not valid.**
 - **Draft-11:** When the ProcessTeepMessage API is invoked, the Agent first does validation as specified in Section 4.1.2, and **if it is not valid then the Agent responds with an Error message.**

#245: Handling of unrecognized TEEP messages (2/2)

- TAM:
 - **Draft-11:** When the ProcessTeepMessage API is invoked, the TAM first does validation as specified in [Section 4.1.2](#), and **drops the message if it is not valid.**
 - Can't send an Error, but the TAM might be able to update the TEEP Agent
 - **Proposed:** "... It may also do additional implementation specific actions such as logging the results or attempting to update the TEEP Agent to a version that does not send invalid messages."

Minor updates in draft-11

- #242 (mcr): renamed bundling “examples” of relationships between manifests and binaries, to “scenarios”
- #243 (mcr): add security consideration about IP address being revealed to trusted binary server when using encrypted binaries
- #244 (mcr): suggest use of timestamp freshness mechanism if nonce storage might be for too long
- #258 (mcr): make EAT for Attestation Results (using TEEP profile) a SHOULD instead of just saying “when EAT is used”
- #269: Is Complete CDDL appendix normative?
 - Clarified that Appendix is informative, body of doc is normative
- #265 (fossati): profile in EAT was renamed to eat_profile

Use of SUIT

#238: Uninstalling trusted components

- Trusted Component Developer might generate a newer manifest that unlinks a component, with a higher sequence number
- But what if TAM or local admin wants to delete it and doesn't have a newer manifest?
- To delete a component, can specify manifests to unlink in Update message

```
options: {  
  ? token => bstr .size (8..64),  
  ? unneeded-manifest-list => [ + bstr .cbor SUIIT_Digest ],  
  ? manifest-list => [ + bstr .cbor SUIIT_Envelope_Tagged ],
```

- Works as long as installation manifest also includes unlink directives
- Recently added to draft-ietf-suit-trust-domains for this purpose
- Fixed in draft -11

#273, 262: SUIT_Envelope vs SUIT_Envelope_Tagged

- ? manifest-list => [+ bstr .cbor **SUIT_Envelope**],
- Tag only needed if multiple cbor types
 - e.g., when stored as a file in generic filesystem
- No other manifest format is currently permitted in TEEP messages
- Should we allow other manifest formats?
 - e.g., existing proprietary ones
- If so, would add a manifest content format optional param, e.g.:
 - ? manifest-list => [+ TEEP_Manifest],
 - TEEP_Manifest = { ? format => text, manifest => bstr }
- Propose: not now, leave for future extension if desired

#282: SUIT digest in unneeded-manifest-list

- ? unneeded-manifest-list => [+ bstr .cbor **SUIT_Digest**],
- Issue:
 - same component might be installed in multiple places
 - Results in multiple component IDs with same digest
 - Can't say which one is unneeded
- Proposal: use SUIT manifest component ID instead of SUIT digest
- ? unneeded-manifest-list => [+ **SUIT_Component_Identifier**],
- SUIT manifest previously had no component ID for a root manifest, only dependencies
 - Propose fixing in draft-ietf-suit-trust-domains

#286: SUIE reports can contain sensitive information

- TEEP protocol does not do encryption at message layer, but payloads (e.g., manifests) can be encrypted using COSE
- Draft-11 says nothing about privacy of SUIE reports
- Neither does draft-ietf-suit-reports
- TEEP isn't the only place SUIE Reports are likely to be used
- Proposal:
 - Leave it to draft-ietf-suit-reports to specify encryption details
 - Add privacy consideration text to teep-protocol draft and refer to draft-ietf-suit-reports for more detailed discussion

Use of EAT

#286: EAT tokens can contain sensitive information

- Draft-11:
 - To lower the privacy implications the TEEP Agent **MUST** present its attestation payload only to an authenticated and authorized TAM and when using an EAT, **it SHOULD use encryption as discussed in {{I-D.ietf-rats-eat}}**, since confidentiality is not provided by the TEEP protocol itself and the transport protocol under the TEEP protocol might be implemented outside of any TEE.
- But EAT profile section references TEEP message cipher suites which don't encrypt:
 - COSE/JOSE Protection: See {{ciphersuite}}.
- Questions:
 - Sign and then encrypt EAT?
 - Which cipher suite for encryption do we specify in the EAT profile?
 - Is it the same as for SUIT reports or might it be different since the sender is different?

#281: EAT profile: mandatory vs optional claims

- Currently all claims are listed as optional
- Thomas Fossati: “The only surprising bit in TEEP (for me) is the absence of mandatory claims: can it really contain *any* claims and still be called a TEEP token?”
- PR #284 proposes:
 - Required Claims: ueid, oemid, hwmodel, hwversion, and manifests.
 - Additional Claims: eat_nonce (present if using nonce freshness mechanism)

#285: EAT Manifests Claim in TEEP Profile

- EAT spec says: “A [SUIT.Manifest] may be used as a manifest.”

```
manifest-format = [  
    content-type:    coap-content-format,  
    content-format: JC< $manifest-body-json,  
                    $manifest-body-cbor >  
]
```

- What CBOR object in body? Propose: SUIT_Reference
- What coap content format? Propose: new value
 - More specific than just application/cbor
 - To be added in draft-ietf-suit-reports

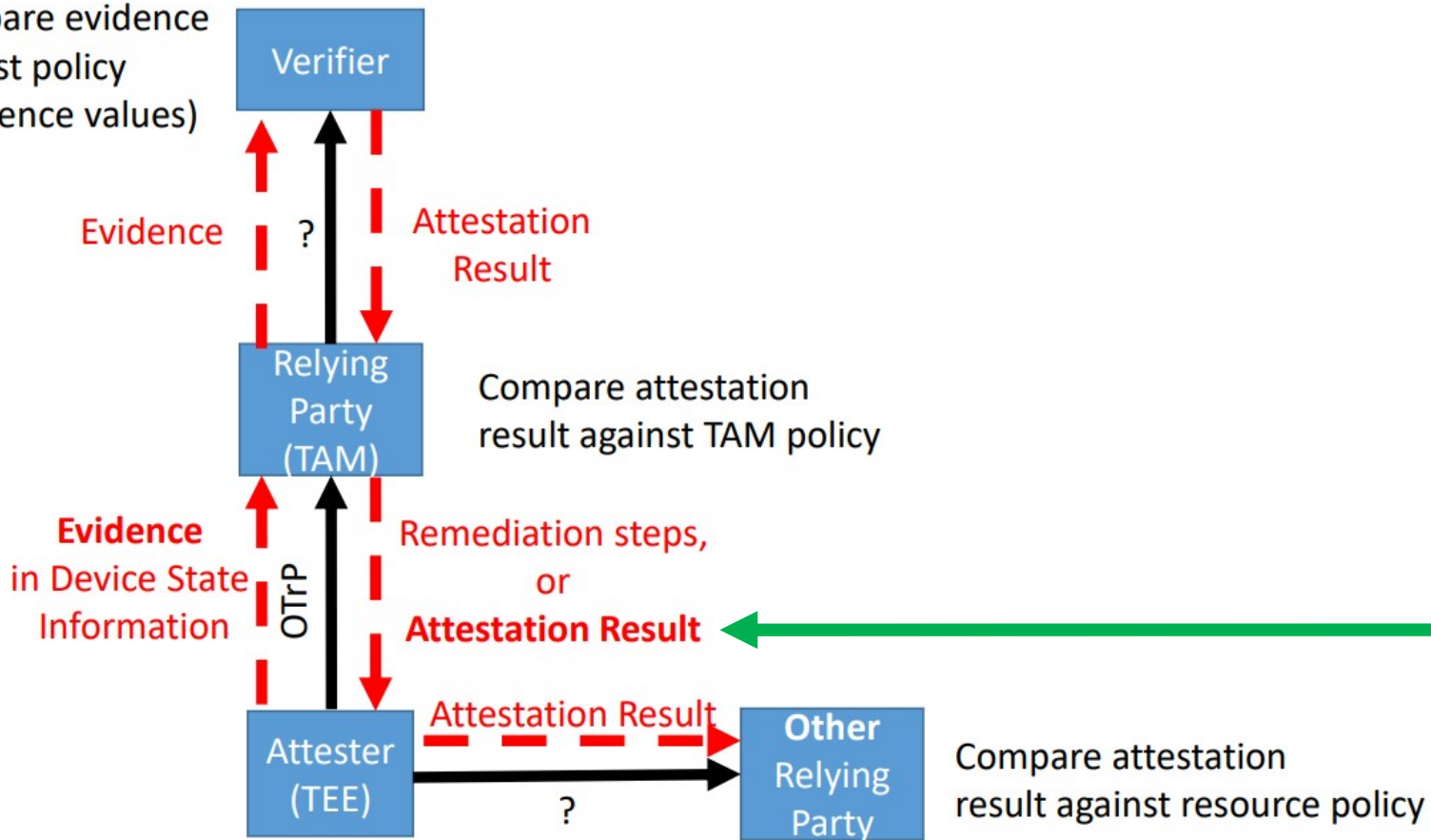
Sample EAT token in TEEP profile

```
/ eat-claim-set = / {  
  / eat_nonce / 10: h'948f8860d13a463e8e',  
  / ueid /      256: h'0198f50a4ff6c05861c8860d13a638ea',  
  / oemid /     258: h'894823', / IEEE OUI format OEM ID /  
  / hwmodel /   259: h'549dcecc8b987c737b44e40f7c635ce8'   / Hash of chip model name /,  
  / hwversion / 260: ["1.3.4", 1], / Multipartnumeric /  
  / manifests / 273: [  
    [ 60, / application/cbor, TO BE REPLACED with the format value for a SUIT_Reference once one is allocated /  
    { / SUIT_Reference /  
      / suit-report-manifest-uri / 1: "https://example.com/manifest.cbor",  
      / suit-report-manifest-digest / 0:[  
        / algorithm-id / -16 / "sha256" /,  
        / digest-bytes / h'a7fd6593eac32eb4be578278e6540c5c' h'09cfd7d4d234973054833b2b93030609'  
      ] } ] ] }  
  ] } ] ] }
```


#215 (IETF 114): Passing Attestation Result back to Attester

Advanced use of OTrP in “Passport model”

Compare evidence against policy (reference values)



□ IETF 105 slide

in TEEP Update message

#289: Relationship between TEEP EAT profile and AR4SI



- TEEP AR is a CWT, but Attester might need AR4SI as JWT or CWT
 - Freshness mechanisms might be different
 - Might even need media type / parameters allowing each combination
 - Option 1: Unify both into one claimset
 - Option 2: Encapsulate TEEP AR in AR4SI
 - Option 3: Encapsulate AR4SI in TEEP AR
- } Have above issues
- Option 4: Make TAM request both from verifier if chained
 - Requires ability to use same evidence nonce for getting both attestation results
 - Option 5: never put the TAM in the middle, always send Evidence separately to each
 - Requires more complex TEEP Agent configuration and behavior in Passport model to do both
 - Harder to initiate remediation when attestation fails since Attester might not parse Attestation Results in Passport model

#240: how does omitting eat profile work with bis documents

- Absence of attestation payload format parameter says it's the current EAT profile
- What if the EAT profile is rev'ed in the future, does this mean that we'll never be able to elide?
- Resolution (done in draft -11):
 - Default value when absent is specified by TEEP protocol version
 - Bumping the TEEP protocol number in the header means default is the attestation payload format value specified by that TEEP protocol version

Other questions?