TEEP Protocol

draft-ietf-teep-protocol-09

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

Fixed in draft-09, as agreed last meeting

- #217: Use media type as defined in draft-lundblade-rats-eat-media-type
- #214: Generalize Challenge in QueryRequest and Attestation-payload in QueryResponse
- #188: Remove IANA registry for data-item-requested
- #185: Clarify use of Attestation Results in TEEP
- #184: Evidence-format is also required for EAT
- #183: Better ciphersuite type definition
- #182: Ciphersuite default values
- #165: Use updated EAT draft instead of draft-birkholz-rats-suit-claims

New issues since last meeting

#226: IANA question about freshness mechanism registry

> IANA is requested to assign a media type for application/teep+cbor.

> …

Value	Freshness mechanism	Specification
1	Nonce	RFC TBD Section 9
2	Timestamp	RFC TBD Section 9
3	Epoch ID	RFC TBD Section 9

IANA: “Where should this new registry be located? Should it be added to an existing registry page? If it needs a new page, does it also need a new category at https://www.iana.org/protocols (and if so, should the page and the category have the same name)?”

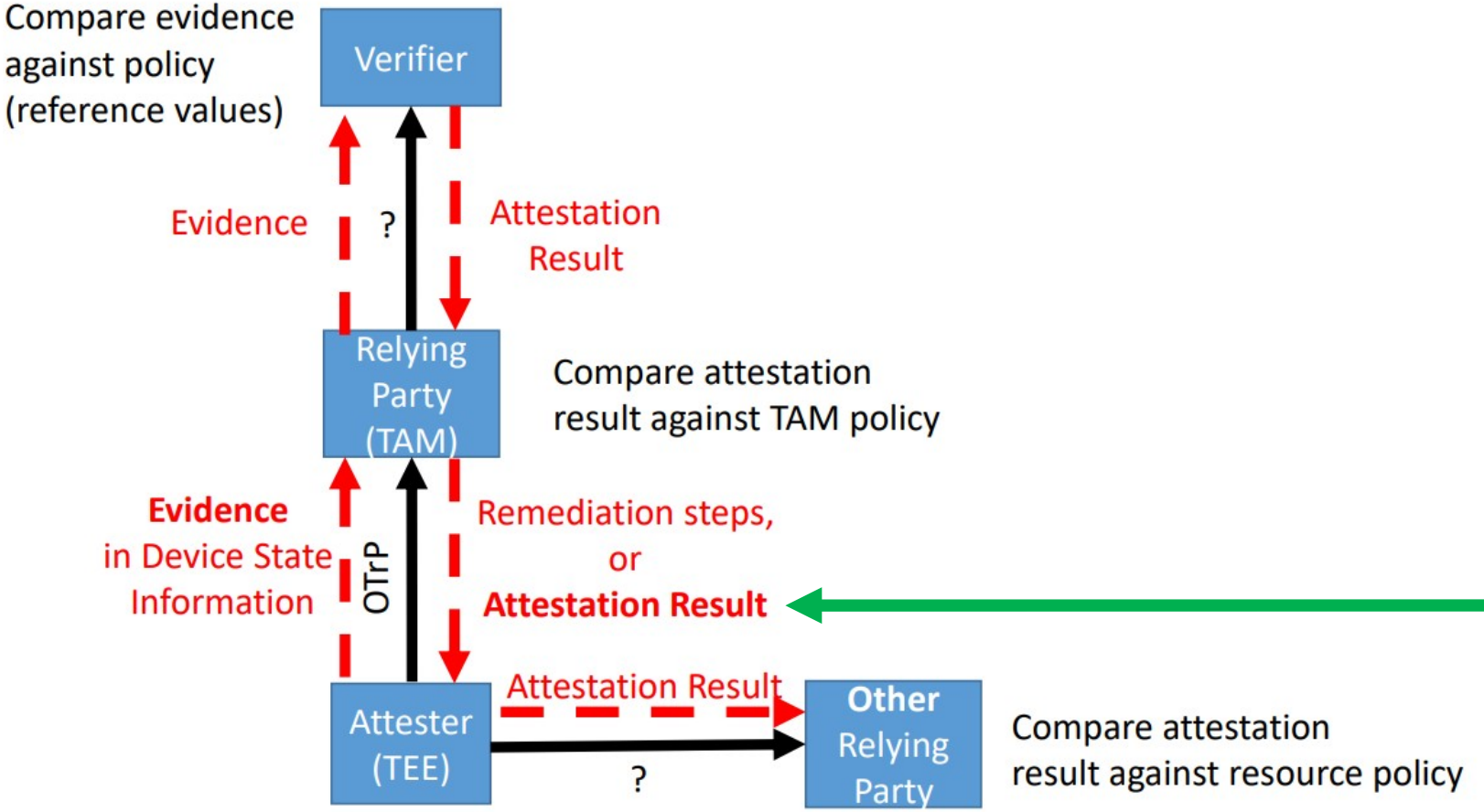
Option 1: New page for TEEP Parameters

Option 2: Move to draft-ietf-rats-reference-interaction-models

Attestation

#215: May require one more message for attestation (1/2)

Advanced use of OTrP in "Passport model"



IETF 105 slide

What TEEP message?

#215 (cont'd): the fix (2/2)

```
 update = [
   type: TEEP-TYPE-update,
    options: {
      ? token => bstr .size (8..64),
      ? manifest-list => [ + bstr .cbor SUIT_Envelope ],
+     ? attestation-payload-format => text,
+     ? attestation-payload => bstr,
      * $$update-extensions,
      * $$teep-option-extensions
    }
```

- Also added diagram from previous slide

#224: Clarify how TAM distinguishes Evidence from Attestation Results in QueryResponse

Updated TAM Behavior to clarify:

IF attestation-payload-format is recognized as Attestation Result

THEN it's Attestation Result

ELSE it's Evidence (send to Verifier to get Attestation Result)

Once have Attestation Result, three cases:

1. Attestation failed (can't trust QueryResponse) try to update TEEP Agent and any dependencies
2. Attestation succeeded, but TC list out of date try to update needed TCs
3. Attestation succeeded, TC list up to date no updates needed, but can pass Attestation Result in Update if Evidence received above

#227: SUIT Component Identifier isn't unique

- Consider:
  - Transfer new component via sneakernet, replacing a previous implementation
  - New implementation has same path and same version number as previous
  - TC List looks same as before since contains Component ID + version
- OLD: `? tc-list => [ + tc-info ],`
- NEW: `? tc-list => [ + system-property-claims ],`

draft-ietf-teep-protocol-09:

```
tc-info = {
  component-id => SUIT_Component_Identifier,
  ? tc-manifest-sequence-number => uint .size 8
}
```

draft-ietf-suit-report:

```
system-property-claims = {
     system-component-id => SUIT_Component_Identifier,
     + SUIT_Parameters, ▯ allows digest, etc.
   }
```

#189: Reliably getting TEE hardware properties

Four possible ways to get TEE properties in a QueryResponse:

- **Attestation Results**: Relies on Verifier copying TEE info into Attestation Result. May be too burdensome for some Verifiers.
- **SUIT Reports**: Relies on TEE generating SUIT Reports at boot time. May be too burdensome for some TEEs. Draft -09 used this
- **TC List**: Meant for components that can be updated via TEEP, not for hardware properties.
- **Some new field:** Github copy does this, using system-property-claims

#221: Reliably getting TEE **firmware** properties (1/3)

Four possible ways to get TEE properties in a QueryResponse:

Attestation Results: Relies on Verifier copying TEE info into Attestation Result. May be too burdensome for some Verifiers.

a) **sw-name + sw-version**: human readable so may not be reliable enough for machine processing draft-09 used this

b) **manifests**: great for machine processing but seems to require entire manifest rather than a reference to it

SUIT Reports: Relies on TEE generating SUIT Reports at boot time. May be too burdensome for some TEEs. draft -09 allowed this too

TC List: Unlike hardware properties, this could make sense, assuming the firmware can be updated via TEEP, but only if the TEEP Agent is healthy

Some new field: (no need)

#221 (cont'd): EAT claims (2/3)

Requirement from arch draft	draft-08	draft-09	github copy
Vendor of the device	oemid	oemid	oemid
Class of the device	class-identifier [Birkholz]	hwmodel	hardware-model
Device unique identifier	ueid	ueid	ueid
TEE hardware type	chip-version	chip-version	hardware-version
TEE hardware version	chip-version	chip-version	hardware-version
TEE firmware type	sw-name	sw-name	manifests
TEE firmware version	sw-version	sw-version	manifests
Freshness proof	nonce	nonce	nonce

#221 (cont'd): Use of EAT claims (3/3)

- `Required Claims: **None**.`
- `Prohibited Claims: None.`
- `Additional Claims: Optional claims are those listed in *[previous slide]*.`

  - Claims on previous slide are for efficiency:

    "A TAM implementation might simply accept a TEEP Agent as trustworthy based on a successful Attestation Result, and if not then attempt to update **the TEEP Agent and all of its dependencies**. This logic is simple but it might result in updating some components that do not need to be updated.

    An alternate TAM implementation might use any Additional Claims to determine whether the TEEP Agent or any of its dependencies are trustworthy, and **only update the specific components** that are out of date."

#220: Multiple problems with ciphersuite negotiation (1/3)

- Problems:
  - Unclear whether TEEP Agent must support COSE_Sign or only COSE_Sign1
  - Contradiction in ciphersuite definition between body vs appendix
  - Unclear what specific ciphersuite combinations are mandatory for TAM
    - it's only clear which algorithms are mandatory
  - Requiring MAC and encryption algorithm support seems overly burdensome
  - Incorrect CDDL mechanism for specifying ciphersuite extensibility
  - No defined way to specify order of operations
    - e.g., sign-then-encrypt vs encrypt-then-sign

#220 (cont'd): Summary of fixes (2/3)

- Change CDDL ciphersuite to $ciphersuite to be explicit that it supports extensibility
- Remove MAC and Encrypt algorithms and leave them to extensions
- Remove COSE_Sign and leave it to extensions (just use COSE_Sign1)
- Fix contradiction between CDDL in body vs appendix
- Specify the order of operations within a ciphersuite is meaningful

#220 (cont'd): Mandatory ciphersuites (3/3)

- `teep-ciphersuite-sign1-es256 = [ teep-operation-sign1-es256 ]`
  - `teep-operation-sign1-es256 = [cose-sign1, cose-alg-es256 ]`

- `teep-ciphersuite-sign1-eddsa = [ teep-operation-sign1-eddsa ]`
  - `teep-operation-sign1-eddsa = [cose-sign1, cose-alg-eddsa ]`

- TAM must support both, TEEP Agent can support either one
- Above have just 1 operation but a ciphersuite can have multiple ordered operations
- Above don't do encryption at the TEEP layer, but permit encryption of the SUIT payload (e.g., using [I-D.ietf-suit-firmware-encryption]).

#222: Make supported-ciphersuites be mandatory in QueryRequest (1/2)

- Any default now will be out of date in the future
  - Little value in specifying a default now
- Proposed fix:
  - TAM always includes its set

```
  query-request = [
      type: TEEP-TYPE-query-request,
      options: {
        ? token => bstr .size (8..64),
-       ? supported-ciphersuites => [ + $ciphersuite ],
        ? supported-freshness-mechanisms => [ + freshness-mechanism ],
        ? challenge => bstr .size (8..512),
        ? versions => [ + version ],
        * $$query-request-extensions
        * $$teep-option-extensions
      },
+     supported-ciphersuites: [ + $ciphersuite ],
      data-item-requested: data-item-requested
    ]
```

#222 (cont'd): Example (2/2)

```
/ query-request = /
[
  / type: / 1 / TEEP-TYPE-query-request /,
  / options: /
  {
    / token / 20 : h'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF',
    / versions / 3 : [ 0 ]  / 0 is current TEEP Protocol /
  },
  / supported-ciphersuites: / [ [ [ 18, -7 ] ], / Sign1 using ES256 /
                                [ [ 18, -8 ] ]  / Sign1 using EdDSA /
                              ],
  / data-item-requested: / 3 / attestation | trusted-components /
]
```

#234: Which TEEP messages are protected with which ciphersuites (1/3)

Section 8 (Ciphersuites) currently says:

- “After a QueryResponse is received, the selected ciphersuite is used in subsequent TEEP messages (Install, Success, and Error).”

Q1: must the same ciphersuite be used in both directions (TAM to Agent, Agent to TAM)?

- "yes" is probably ok for now
  - is implied in text quoted text, but not explicitly stated
- Leave it to a TEEP extension if separate mechanisms are needed later

#234: Which TEEP messages are protected with which ciphersuites (2/3)

Q2: does that mean that the QueryResponse cannot be protected?

> Might the attestation payload and the SUIT reports in the QueryResponse be considered sensitive information in some cases?

QueryResponse could be protected by the selected-ciphersuite

- "selected-ciphersuite" is inside the TEEP message
- Q: can a receiver figure it out from a COSE object and parse correctly?
- Q: can attestation payload be encrypted?
- Q: can SUIT reports be encrypted?

#234: Which TEEP messages are protected with which ciphersuites (3/3)

Q3: does that mean that an Error sent in response to a QueryRequest cannot be protected?

- Again the SUIT reports in the Error might be considered sensitive information in some cases.

Perhaps:

- If the TEEP Agent was able to select a ciphersuite from among the TAM's supported-ciphersuites, then use it to protect the Error message.
- Otherwise, protect the Error with a mandatory ciphersuite that the TEEP Agent supports.
- Don’t include sensitive information with a ciphersuite that doesn’t encrypt.

Ken: “Propose to add suit-uninstall in Multiple Trust Domains draft”

- To delete a component, TAM must send a new SUIT manifest with higher sequence number

```
update = [
  type: TEEP-TYPE-update,
  options: {
    ? token => bstr .size (8..64),
    ? manifest-list => [ + bstr .cbor SUIT_Envelope ],
    * $$update-extensions,
    * $$teep-option-extensions
  }
```

- What if the TAM has a manifest from a third party? Encapsulate/resign?
- What if device has a component the TAM doesn’t know about?
- Ken proposed adding uninstall directives into SUIT installation manifests
- If SUIT does that, we could add back in ability to delete by ID

Next steps

- Post updated draft
- Initiate WGLC?