IETF 111 TEEP Hackathon

July 30, 2021

Akira Tsukamoto (AIST)

IETF 111 SUIT TEEP Hackathon

- Date July 21, 17:00- in JST
- Participants:

Akira Tsukamoto, AIST

Kuniyasu Suzaki, TRASO/AIST

Kohei Isobe, SECOM

Ken Takayama, SECOM

Masashi Kikuchi, TRASIO

Takahiko Nagata, TRASIO

Brendan Moran, ARM

Hannes Tschofenig, ARM

Dave Thaler, Microsoft (was midnight)

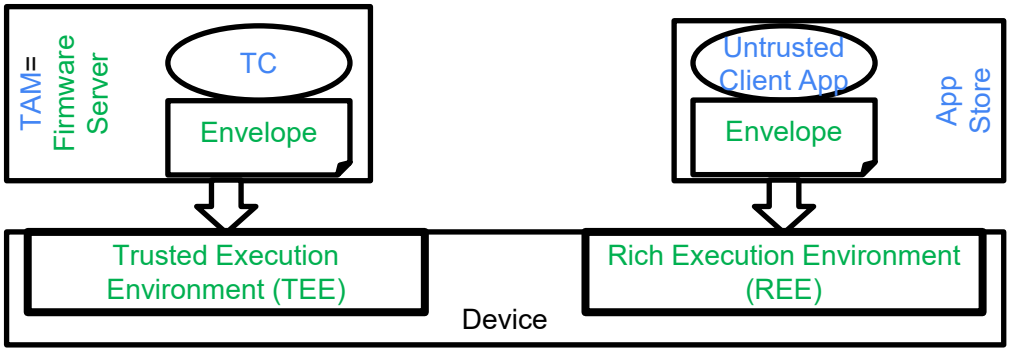
Background and activities

- TEEP requires SUIT and RATS
- Started implementing SUIT manifest format in TEEP protocol
  - Our implementations had only TEEP protocol portion and did not have SUIT manifest in the message
  - Started adding from SUIT manifest, without COSE for TEEP message

- Matching terminology between TEEP and SUIT draft
- The challenge of generating SUIT manifest for TEEP use cases
- Three candidates of SUIT manifests
- Creating format of SUIT manifests

Terminology: TEEP (in blue) SUIT (in green)

1. App Developer = Author generates SUIT manifest
   - Untrusted/Client App executed in REE (untrusted area) and its SUIT manifest
   - Trusted App (=Trusted Component, TC) , Firmware and its SUIT manifest
     - TC is signed by TC-signer (may or may not be App/TC Developer)
2. Upload TC to App Store, TAM, Firmware Server
   - Untrusted/Client App and its SUIT_Envelope to App Store
   - Trusted Component (TC) = Firmware and its SUIT_Envelope to Trusted Application Manager (TAM) = Firmware Server
3. TAM and App Store distribute TC and Client APP respectively

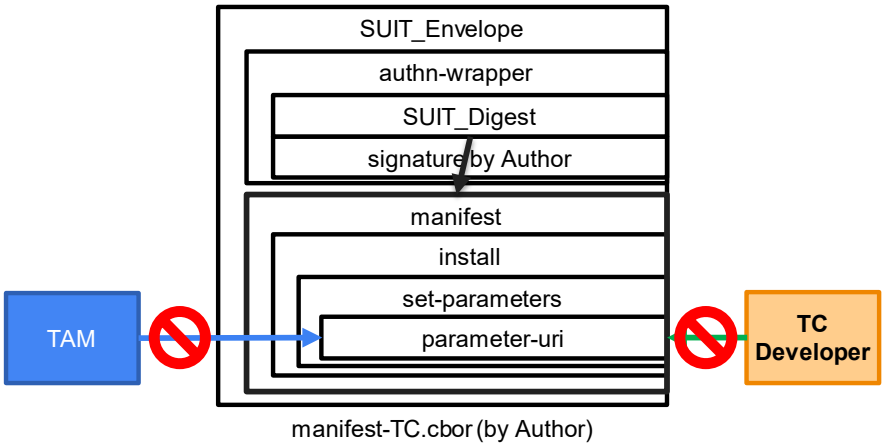


Challenge: Two URIs by TC Developer and TAM

- TC Developer = Author create manifest with URI
  - It is in manifest signed by Author
- TAM = Firmware Server would like to have different URI in some cases
  - TAM is hosting TC binary instead of TC Developer
  - Using third party hosting server, CDN, etc.

github #104 TAM to have a way to choose uri for TC

github #105 Construction of SUIT_Envelope of URI outside the digest region



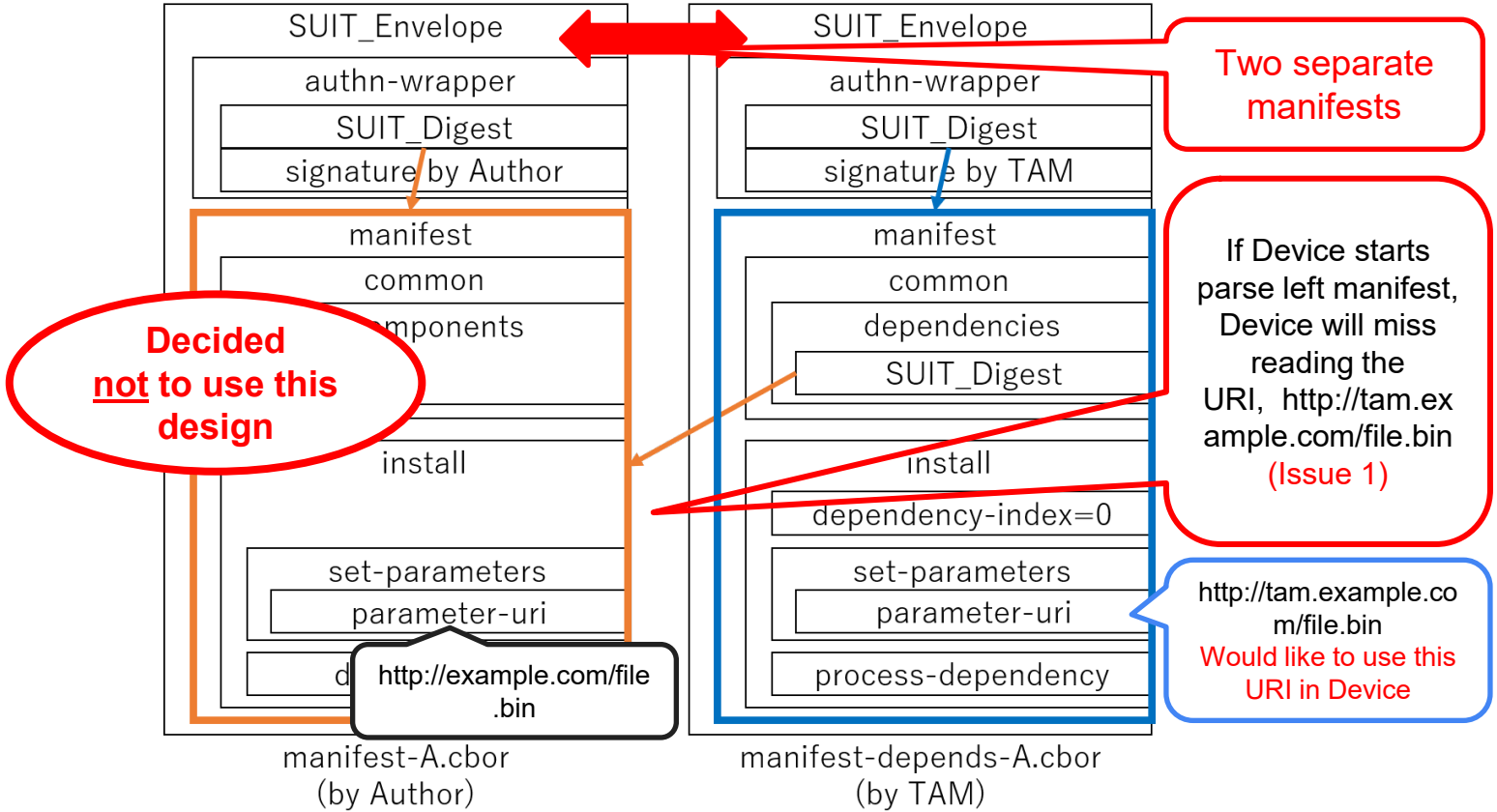
Three candidates of manifest design

- From discussion at IETF110 TEEP WG session, Concluded to use “Dependency” for having different URI from TC Developer and TAM.
  - https://datatracker.ietf.org/doc/html/draft-ietf-suit-information-model-12#section-3.19
  - **"Dependencies**
  - A list of other manifests that are required by the current manifest.“

- Considered three manifests #1, #2, #3
- manifest #1 and #2 – Using Dependency
  - Pros: Device can verify signatures both Author and TAM
  - Cons: Issue1 Device could not find which to parse first among two manifests.
  - Cons: Issue2-1 how to override URI with integrated dependency
  - Cons: Issue2-2 Require order of parsing Set & override in Devise

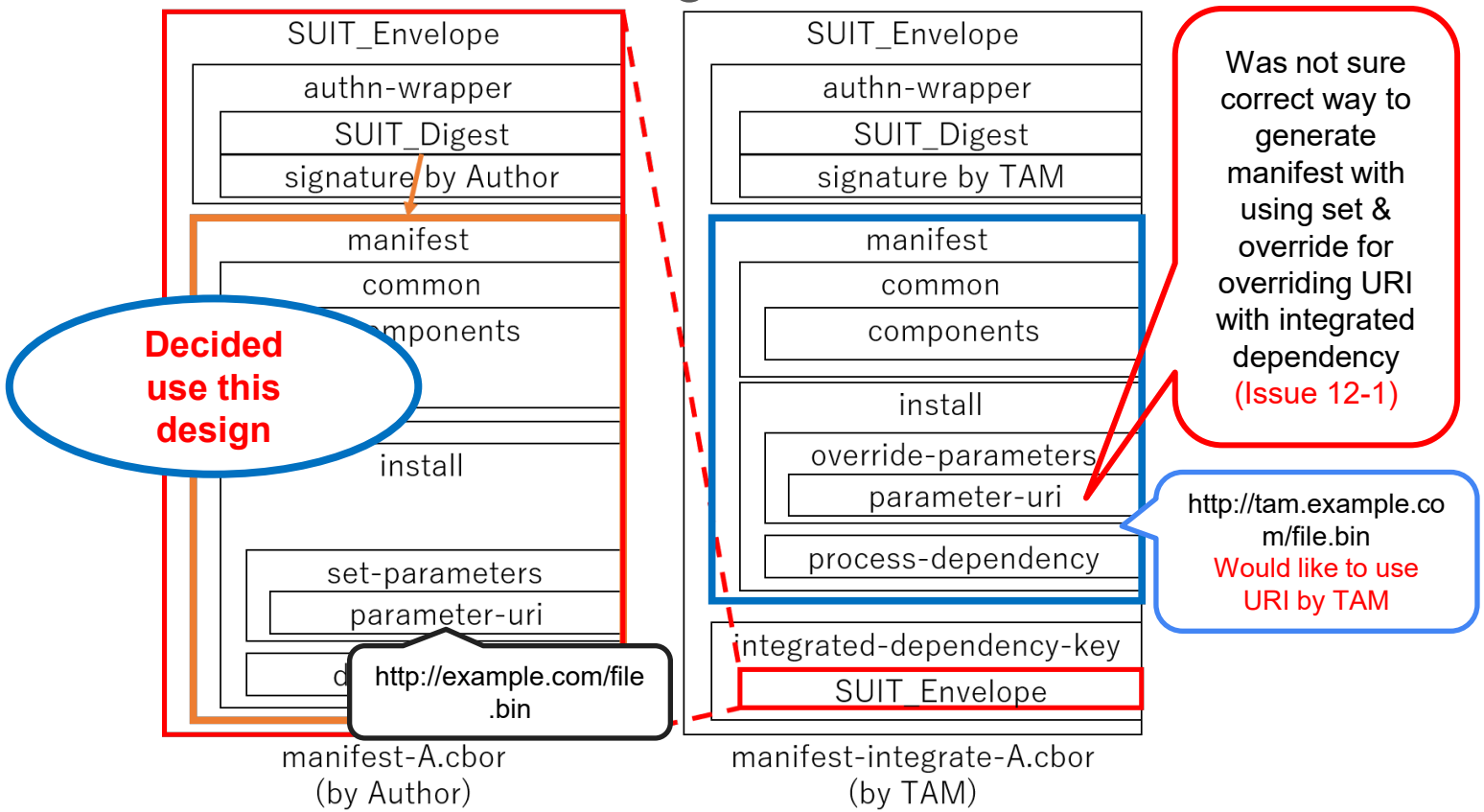
manifest #1: Using Dependency

- TAM override the URI using dependency



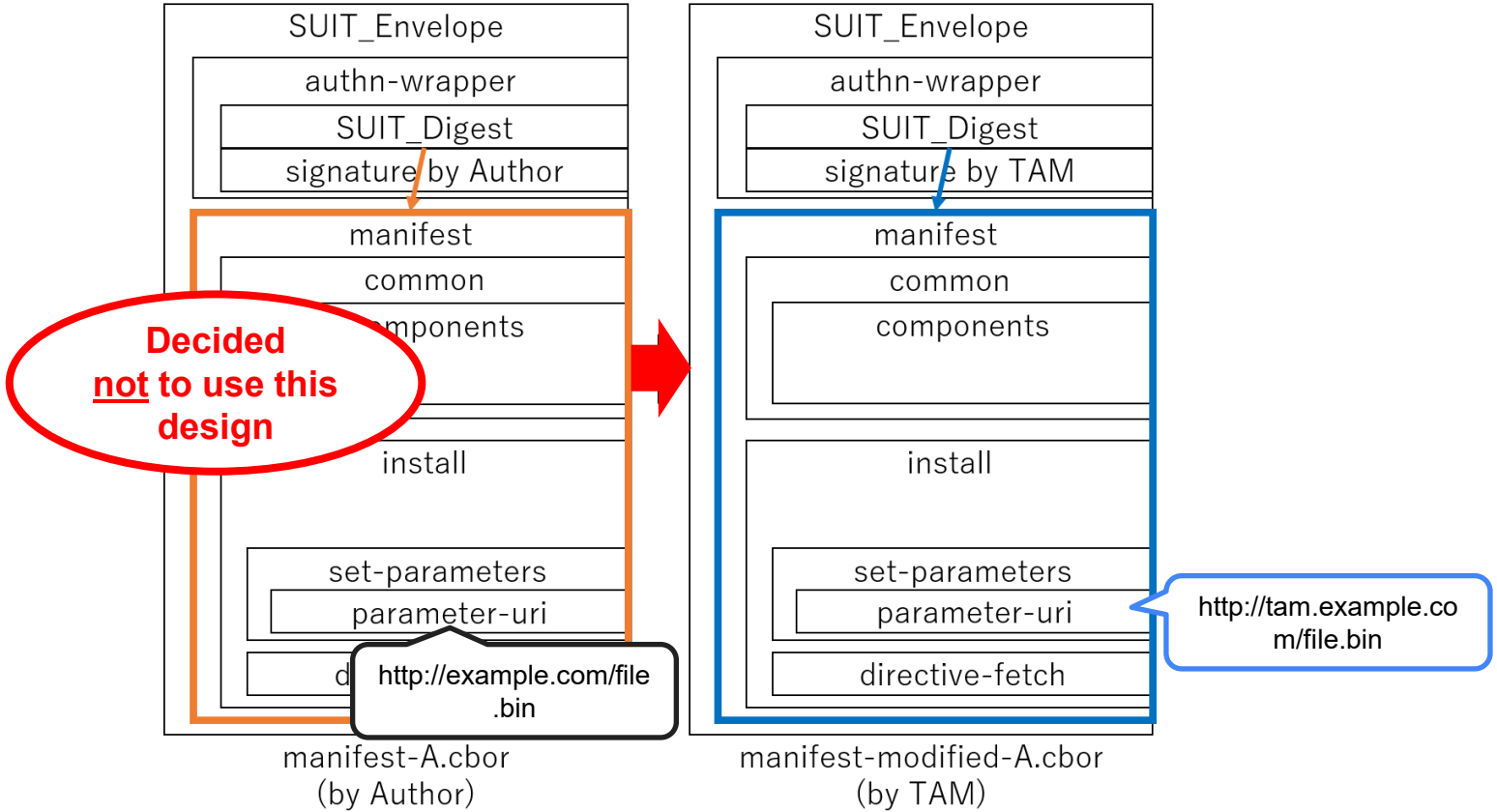
manifest #2: Using Integrated-Dependency

- TAM embedding SUIT manifest from TC Dev



manifest #3: TAM modifying both URI and signature

- TAM re-signing it with TAM's URI



Requirements of SUIT manifest

- SUIT manifest generated by TC Developer
  - Have one entry of pointing TC binary (SUIT component) inside SUIT manifest
  - Have Install command sequence
    - Specifing URI and conduct directive-fetch
- SUIT manifest generated by TAM
  - Have manifest generated by TC Developer inside with Integrated Dependencies
  - Have Install command sequence
    - Have process dependency to follow install command sequence in manifest created by TC Developer, but override URI.

Considered SUIT_Envelope format

```
00  SUIT_Envelope {
01    authn-wrapper : {
02      SUIT_Digest = `digest-of-manifest-TAM`
03    }
04    integrated-dependency-key : {
05      SUIT_Envelope {
06        authn-wrapper : {
07          SUIT_Digest = `digest-of-manifest-TC`
08        }
09        manifest : { // manifest-TC (by Author)
10          common : {
11            components [ [ h'00' ] ]
12          }
13          install : {
14            directive-set-parameters : {
15              uri = "http://example.com/hello.txt"
16            }
17            directive-fetch
18          }
19        }
20      }
21    }
22    manifest : { // manifest-TAM
23      common : {
24        dependencies : [
25          SUIT_Digest = `digest-of-manifest-TC`
26        ]
27        components : [ [ h'00' ] ]
28      }
29      install : {
30        directive-set-component-index = 0
31        directive-override-parameters : {
32          uri = "http://tam.example.com/hello.txt"
33        }
34        directive-set-dependency-index = 0
35        directive-process-dependency
36      }
37    }
38  }
```

Having digest-of-manifest-TC ahead of the manifest-TAM, to make it easier when parsing common entry in manifest-TAM would already know manifest-TC.

(Did not include validating image digest to make this example simple)

Override the URI, and then fetch it with process-dependency

Summary

- Mapped terminologies between TEEP and SUIT draft
- Adding SUIT manifest for TEEP use cases
  - TAM having different URI based on manifest generated by TC Developer
- Considered three designs of SUIT manifests
  - Decided to use Integrated Dependency for TEEP Protocol
- Creating format of SUIT manifests
  - Discussion of details of is ongoing
- After IETF 111 to IETF 112
  - Finish SUIT manifest implementation
  - Add reference formats in TEEP drafts, start supporting COSE

A part of this hackathon presentation is based on results obtained from a project, JPNP16007,commissioned by the New Energy and Industrial Technology Development Organization (NEDO).