TEEP Architecture Draft

draft-ietf-teep-architecture-01

Mingliang Pei, Hannes Tschofenig

IETF#103, Bangkok

Agenda

- Interim work and meeting update
- Document Status
  - Changes from the last version
- Issues Update

Interim WG and Work Sessions

- Met three times
  - Work sessions (8/2, 9/17, 10/22) + interim WG meeting (10/22)
  - Authors (8/2), Authors + Chairs (9/17), Interim WG session (10/22)
- GitHub created for TEEP
  - https://github.com/ietf-teep
- Issues filed and tracked in GitHub
  - 18 issues
- Progress on several issues

Draft Status Update

- v01 published draft-ietf-teep-architecture-01
  - David Wheeler joined as a co-editor
  - Editorial changes
    - Introduction paragraph is rewritten with more context about TEE and REE
    - Expanded description about a SP's need in problem statements
    - A new conceptional component diagram (Section 5.1)
    - Some terminology update
      - Agent → Broker, Application → Application Component
      - Added "Device Administrator (DA)", "Root of Trust", "TFW" and "Bootloader Key". Removed "Secure Boot", updated "Trust Anchor" etc.
  - Incorporated changes for a few resolved issue
    - E.g. explicit section to specify requirement of offer algorithm and attestation agility (Section 8)

#5: Option to not use secure boot

- TFW and Secure Boot clarification
  - SGX doesn't depend on secure boot.
  - Comments on Secure Boot vs. Measured Boot (Henk)
  - Consensus in IETF 102 and follow up confirmed to make architecture agnostic to secure boot
- Document change has been made
  - Secure Boot is removed
  - Consistently use TFW verification, and use of TFW is optional
- Details at https://github.com/ietf-teep/architecture/issues/5

Revised TFW Definition

- **Trusted Firmware (TFW)**: A firmware in a device that can be verified with a trust anchor by RoT for Verification.

- **Bootloader key**: This symmetric key is protected by electronic fuse (eFUSE) technology. In this context it is used to decrypt a TFW private key, which belongs to a device-unique private/public key pair. Not every device is equipped with a bootloader key.

#4: Algorithm Agility

- Confirmed this is in agreement
  - Future algorithms can be negotiated and added

- An explicit paragraph added in v01
  – Section 8

#6: Attestation Agility

- An explicit paragraph added in v01
  - Section 8

- More to work on the protocol level design
  - We may add a header in the current OTrP protocol for attestation, say, "native" type.
  - This allows use of future other types of attestation data format (e.g. EAT, RAT etc.)

Root of Trust vs. Trust Anchor

- Updated in v01

- Trust Anchor: A public key in a device whose corresponding private key is held by an entity implicitly trusted by the device. The Trust Anchor may be a certificate or it may be a raw public key.

- Root-of-Trust (RoT): a *hardware or software component* in a device that is inherently trusted to perform a certain security-critical function. A RoT should be secure by design, small, and protected by hardware against modification or interference.

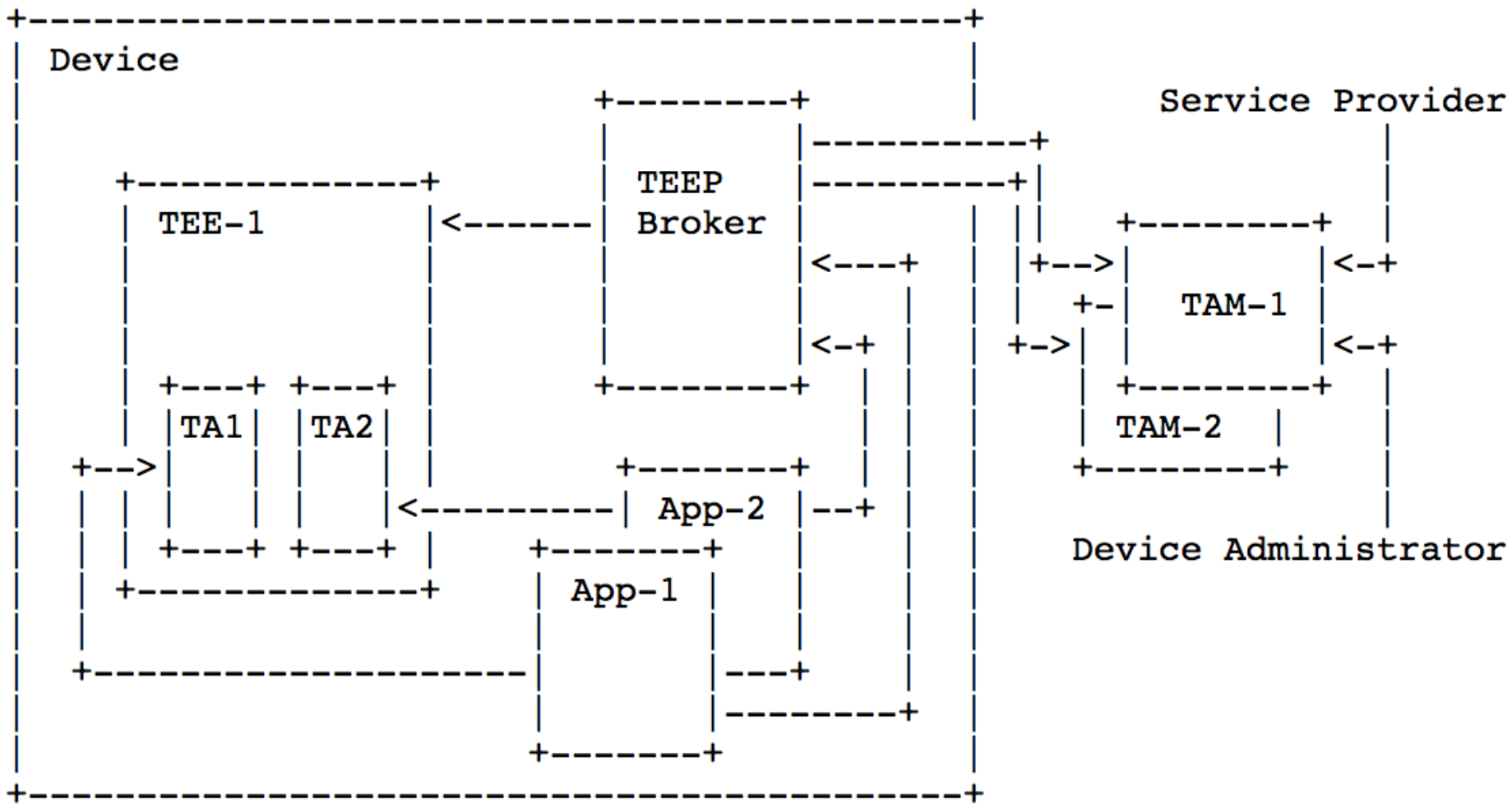
#16: Terminology for “Agent”

- Agreed to use word “Broker”
- Change done in v01

#29 Device Admin vs. Device Owner

- Device Administrator
  - "An entity that owns or is responsible for administration of a Device. A Device Administrator has privileges on the Device to install and remove applications and TAs, approve or reject Trust Anchors, and approve or reject Service Providers, among possibly other privileges on the Device. A Device Administrator may choose to remotely administrate a device through a TAM."
  - "A Device Owner can manage the list of allowed TAMs by modifying the list of Trust Anchors on the Device."

- Revised context diagram to include Administrators


Device
TEEP Broker
TEE-1
TA1
TA2
App-2
App-1
Service Provider
TAM-1
TAM-2
Device Administrator

ISSUES WORK IN PROGRESS

#18 Trust Anchor Update (New)

- This was assumed to be handled by device administrators or providers

- Introduce the use of a system Manager TA pre-installed in a TEE for check and update of Trust Anchors

#3: Trusted App Distribution

- Desirable Two Modes
  - TA binary bundled with the Client Application
  - TA distributed by TAM
- Challenges with first approach is
  - Passing TA instance specific data requires real-time interaction with a TAM
  - Client Application is not authorized to query TEE device state. Who is authorized to update a TA in the future?
- Resolution
  - TA binary + configuration support by TAM only
  - TA binary can go with either client app or TAM
  - TEEP protocol won't support TA binary encrypted delivery from a Client App

#7: Clarify meaning of Security Domain (SD)

- Concern of overhead to manage SDs
- Articulation of use of SD
  - Broadly used in GP, fitting the existing model in practice
    - Make it optional for other use cases?
  - Some TEEs allow shared resources or some level of trust among TAs
    - Represent ownership of multiple TAs to the same SP
    - There are other ways to represent this
  - Is it needed in IoT and SGX cases?
    - Implicit one SD per TA could work
  - Concern of overhead to manage SDs
    - Optimize by allowing use of one implicit SD per TA
- Details at https://github.com/ietf-teep/architecture/issues/7

#9: Install TA in Single Pass?

- Not always
- Flow update per Hackathon feedback
  - Initial TAM GET call is necessary
    - Only provide device signing key information to a trusted TAM, not others
  - Optimize to do this Single Pass for a device that has had cached TAM information

#10: Local TEE signing first

- See #9
- Local TEE signing first would leak the TEE signing key to potentially unknown TAM

#12: Every Rich App talks to TAM?

- Not necessary
- Adopt support of metadata file and installer in REE to initiate Broker contact with a TAM
- See also #9

OTHER ISSUES

#8: Multiple TEEs vs. Single TEE

- Impact on message routing
  - Multiple Broker use (To be added)

#13: Is it in scope: TA depends on another TA?

- Discussed in interim work sessions
- Concerns
  - Complex: very deep dependency
  - Circular dependency
- Scope
  - Should dependent TAs belong to the same SP or TAM?

#11: Role of Client Application

- Issue: clarify how much Client Application needs to participate in TAM interaction

- Discussion on use of “installer” in TEE to initiate the TAM call

#14: Multiple TAMs for single Client App?

- TAM is associated with a TA, not a Client App
- A Client App may depend on multiple TAs. Two different TAs may be from different TAMs
- Incline to keep it simple: single TAM
- To follow up later

Q&A

Thank you!