# TEEP Architecture Draft

https://tools.ietf.org/html/draft-ietf-teep-architecture-00

Mingliang Pei, Hannes Tschofenig, Andrew Atyeo, Dapeng Liu

IETF#102

# Document Structure

- Introduction
- Terminology
- Scope and Assumptions
- Use Cases (Payment, Authentication, IoT, Confidential Cloud Computing)
- Architecture
- Agent
- Attestation
- Security Consideration

## Trusted Firmware

- Agreement in the group to make trusted firmware functionality optional since it is TrustZone-specific
- Still lots of left-over text in the document present
- Clean-up in next version

# Trusted App Distribution

- Desired feature
  - Signed TA binary bundled in a Client Application
  - TA obtained on demand by TAM
- Challenges with TA bundled with client app
  - TEE device state query and update authorization
    - TEE trusts a signer that represents a TAM.
    - A Client Application isn't trustworthy. A trust assertion must be from some entity separate from a Client Application.
  - Passing data that is device or TA instance specific when a TA starts requires online interaction with a TAM.  Functionality used today

## Multiple TEEs vs. Single TEE

- The original OTrP assumes that device is equipped only with a single TEE.
- One TEE per device deployment is common today.
- Multiple TEE support was asked. Use case unclear.
- Technical issue: How are messages routed to the correct TEE?
  - TEEP Agent is responsible to get a TEE identifier, and connects to the right TEE
  - It implies that TEEP Agent needs to parse the TAM messages or some header
  - The TEE identifier is better to be verifiable as the actual intent

## Every Rich App Talks to TAM?

- Dave Thaler: The document seems to assume that every rich app that needs a TA, also needs to have code to talk to a TAM.
  - I agree that's one possible implementation but the architecture should not require that.
  - An alternative implementation would be where the REE OS (e.g., the app store installer) contains code for communicating with TAM(s) using data in (for example) the rich app's manifest, and the rich app has no need to run until the TA is already installed. The installer/communicator is a rich app of a sort, but it's a different one from the client app that depends on the TA.
- In my opinion, the arch doc should not require that it's the same app, but should certainly allow it to be.

# Service Provider Terminology

- Dave Thaler: If the Device Admin is responsible for controlling what apps run in the TEE on an IoT class device (e.g., in a factory), is the Device Administrator just another type of SP, or are those terms disjoint?
- Discussion
  - How to identify an administrator that can control the device and TEE can authorize it?
  - The intent can be achieved via a TAM that the administrator trusts.
    - Local administration?

# Keys

(Not present in SGX)

Used for attestation

Used for Software signing

| Key Entity Name | Location | Issuer | Checked Against | Cardinality |
|---|---|---|---|---|
| 1. TFW key pair and certificate | Device secure storage | FW CA | A white list of FW root CA trusted by TAMs | 1 per device |
| 2. TEE key pair and certificate | Device TEE | TEE CA under a root CA | A white list of TEE root CA trusted by TAMs | 1 per device |
| 3. TAM key pair and certificate | TAM provider | TAM CA under a root CA | A white list of TAM root CA embedded in TEE | 1 or multiple can be used by a TAM |
| 4. SP key pair and certificate | SP | SP signer CA | A SP uses a TAM. TA is signed by a SP signer. TEE delegates trust of TA to TAM. SP signer is associated with a SD as the owner. | 1 or multiple can be used by a TAM |

## Root of Trust vs. Trust Anchor

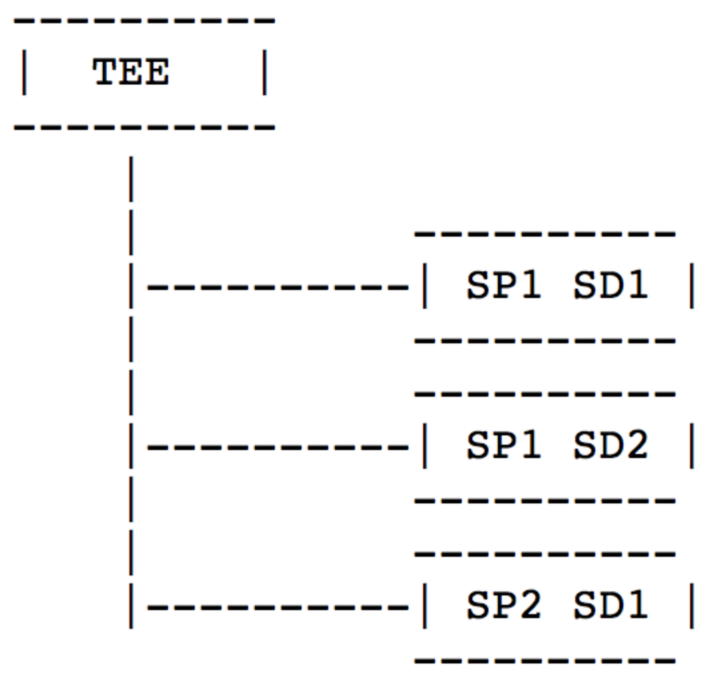
- Attempt to differentiate the different uses of the certificates.
- David Wheeler proposed terminology for the two terms.
- Andrew proposed to remove trust anchor term and to use terms like “TAM root CA certificate store”.

- **Trust Anchor**: A trust anchor is public asymmetric key, preferably contained in a certificate that represents a trusted entity to a device. This public key may be used by the holder of the corresponding private key to sign other certificates, thereby communicating the signed certificate may also be trusted. The trust anchor is usually embedded in a device or configured by a TAM and used by the device to validate the trust of a remote entity by verifying that entity's certificate is signed by a trust anchor. Trust anchors must be stored in a way that prevents or strongly resists modification by unauthorized software and hardware adversaries. An example of a trust anchor is the public key of a TAM or SP which is "pinned" or securely stored inside the device, and that trust anchor is used like a CA certificate to validate the trust in other keys/certificates. A trust anchor may be viewed by both trusted and untrusted entities on the device, But may only be modified or deleted by a trusted entity.

- **Root-of-Trust Key:** A device-unique key generated at device manufacturing or at TEE provisioning, which is securely stored and only accessible to the TEE. The Root-of-Trust Key is used for attestation signing which proves the signed message originated or was approved by the TEE, and may be trusted to the same degree as the TEE.

# Security Domain Concept

- Currently one level security domain hierarchy assumed.
- Purpose of the domain is for isolation of resources. TA in one SD cannot access resources of a TA in another SD.
- Up to TEE's implementation of isolation and access control.
- Definition of security domain not available and use cases unclear.
- Implication of SD concept is in the message exchange that requires messages to create and delete security domains.


TEE
SP1 SD1
SP1 SD2
SP2 SD1