

TEEP Architecture Draft

[draft-ietf-teep-architecture-02](#)

Mingliang Pei, [David Wheeler](#), Hannes Tschofenig

IETF#104, Prague










Agenda

- Document Status
 - Changes from the last version
- Issues Update

Draft Status Update

- v01 published [draft-ietf-teep-architecture-02](#)
 - Clarified Device Administrator further with added “Device User” and “Device Owner”
 - TA distribution variations updated
 - Multiple TEE illustration
 - Multiple TAM consideration
 - Attestation specification

All Closed Issues on GitHub

- | | |
|--------------------------|---|
| <input type="checkbox"/> |  TEEP Architecture Diagram
#39 by hannestschofenig was closed on Dec 10, 2018 |
| <input type="checkbox"/> |  Figure one arrow directionality correct?
#36 by ncamwing was closed 16 days ago |
| <input type="checkbox"/> |  Device administrator vs Device owner
#29 by nicopal was closed 16 days ago |
| <input type="checkbox"/> |  Terminology for "Agent"
#16 by hannestschofenig was closed on Nov 7, 2018 |
| <input type="checkbox"/> |  Applicability of TPMs to TEEP
#15 by hannestschofenig was closed on Oct 22, 2018 |
| <input type="checkbox"/> |  Every Rich App Talks to TAM?
#12 by hannestschofenig was closed on Nov 7, 2018 |
| <input type="checkbox"/> |  Attestation Agility
#6 by hannestschofenig was closed on Nov 7, 2018 |
| <input type="checkbox"/> |  Option to not use secure boot
#5 by hannestschofenig was closed on Nov 7, 2018 |
| <input type="checkbox"/> |  Algorithm Agility and Longer Key Sizes
#4 by hannestschofenig was closed on Nov 7, 2018 |

GitHub Open Issues

Issues Ready To Be Closed

Issue #	Description
#3	TA Packaging and Distribution
#8	Multiple vs Single TEE in Device
#13	Support for TA-to-TA dependency
#14	Multiple TAMs for Single Client App

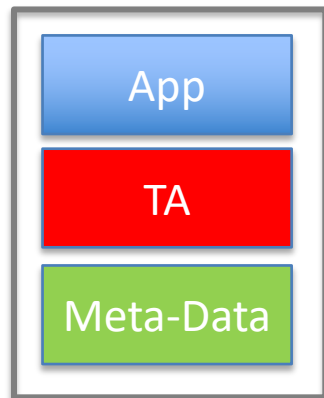
Issues Requiring More Work

Issue #	Description
#7	Security Domains
#9	Install TA in a single pass
#10	Connection Flow – Device to connect to TAM
#11	Role of Client App
#17	Capabilities of Attestation Mechanism
#30	Cardinality of Key Pair and Certificate
#31	SEED for TAM protocol
#32	Trust Anchor Lifecycle Management
#34	Dependencies between Client App & TA
#35	Coordinate TA updates with Client App
#37	Sample Device Setup flow
#38	Trust Anchor Fingerprint

#3: Trusted App Distribution

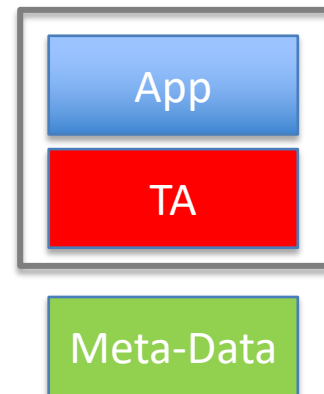
- Discussed in IETF 103
- Added a section to clarify different ways to combine:
 - Client Apps, TA, Personalization Data
- Three variations to be supported by TEEP

Case 1:



One Bundle

Case 2:



Two Bundles

Case 3:



Three Bundles

Primary issue is the encryption of the Meta-Data to preserve Privacy for the TA and Device/User.

Creating an encryption key while also providing proof-of-source from the SP issuing the personalized data is needed.

Such keys require the TEE to be engaged in creating the keys, and many TEEs cannot create such a key until the TA itself is instantiated on the device.

#8: Multiple TEEs vs. Single TEE

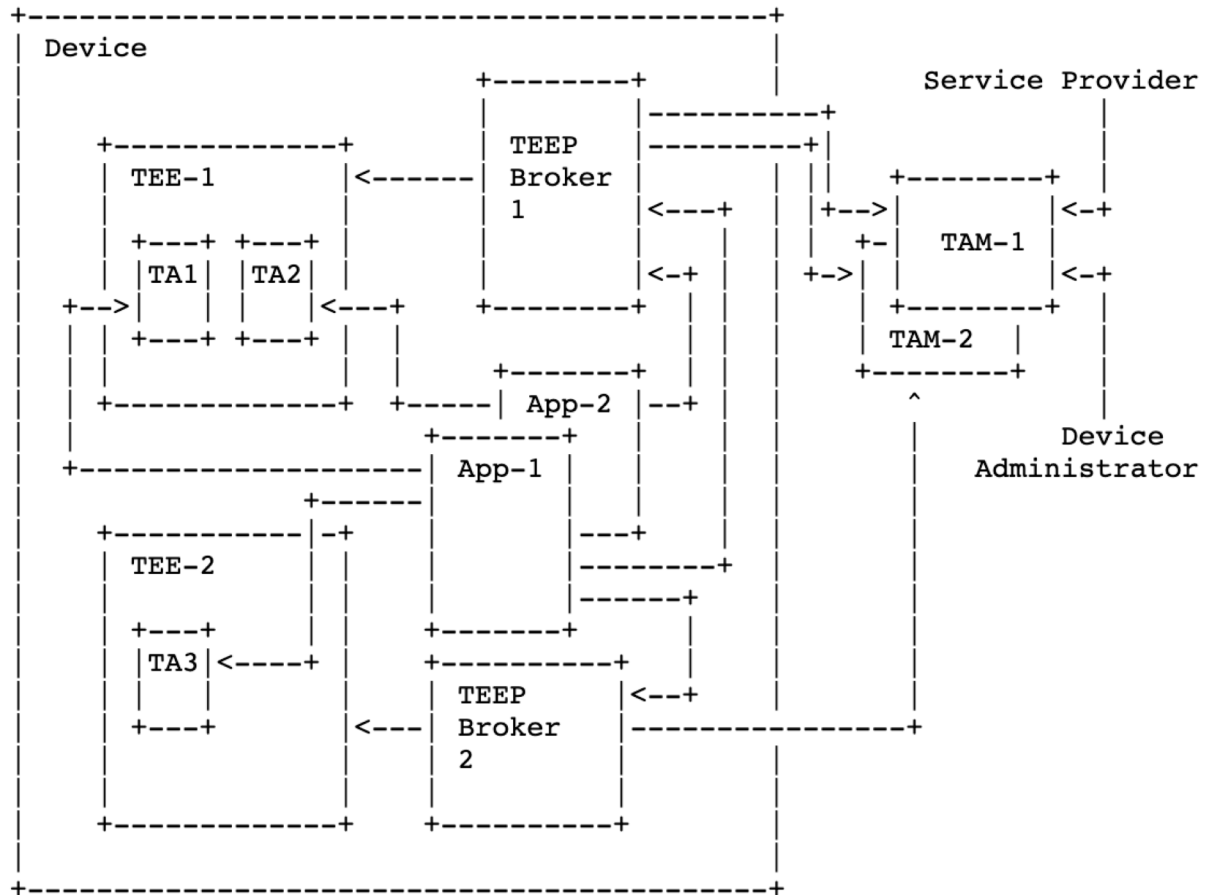


Figure 2: Notional Architecture of TEEP with multiple TEEs

- Multiple TEEs can be supported using multiple TEEP Brokers.
- Each TEE has their own TEEP Agent
- App selection of the proper TEEP broker selects the TEE that will be used. This does not seem to be a difficult or burdensome requirement on Apps.
- This does not prevent or restrict any one TEEP Broker from supporting multiple TEEs

#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 could be from different TAMs when multiple third party TAs are used by a Client App
 - However, a SP will typically provide the TAs of their own or work to acquire those third party TA binary to supply to a TAM on its choice.
- Resolution
 - **A Client App manifest file can contain all TAMs it may use to get TAs, normally just one**
 - Go with simple case that single TAM is contacted by a TEEP Broker for a Client App
 - A SP provides each TAM that it places in the Client App's manifest all the TAs that the app requires, so any TAM can provide all the TA's
 - For third party TAs that a Client App may depends on, a TAM can reach out to the original TAMs for those third party TAs that it is missing, but this would be a TAM implementation specialization
 - *We chose this over alternative approach that let TEEP Broker contact respective TAM per TA for simplification of majority case.*

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

- Discussed in interim work sessions
- Concerns
 - Complex: very deep dependency
 - Circular dependency
- Recommendation
 - Defer dependencies to a future version as this is an advanced case

#17 Capabilities of the Attestation Mechanism

- Changes made to:
 - Define attestation
 - Describe assumptions required for an attestation
 - Identify the need to support both proprietary and standard attestation signatures
 - Proposed format of attestation (may need more work & discussion)

TEEP Attestation

- **Attestation** is the process through which one entity (an attestor) **presents a series of claims to another entity** (a verifier), provides sufficient proof that the claims are true.
- Different verifiers may have different standards for attestation proofs and not all attestations are acceptable to every verifier.
- TEEP attestations are based upon the use of an asymmetric key pair under the control of the TEE to create digital signatures across a well-defined claim set.
- In TEEP, the **primary purpose of an attestation is to allow a device to prove to TAMs and SPs that a TEE in the device has particular properties**, was built by a particular manufacturer, or is executing a particular TA. Other claims are possible.

How will Attestation be Used?

- **Prove Identity of a Device or TA to authorize use of keys or secrets**
 - Attestation is used as the authorization vector for key usages
 - Includes use, migration, wrap-and-transport, delete, derive, etc.
- **Prove Identity of a Device or TA to prevent masquerading attacks on services**
 - Attestation is the authentication mechanism for TEEs
 - include TEE-to-Platform-or-Service, TEE-to-TEE, TEE-to-Remote-TAM/SP
- **Prevent unauthorized interactions between TAs**
 - Attestation is the authorization mechanism for TAs
- **Prevent unauthorized devices or TEEs from installing/using a TA**
 - Attestation is the authentication mechanism for environments

Attestation: Cryptographic Properties & Assumptions

- **Cryptographic Properties**

- **Non-repudiation, Unique Proof of Source** - the cryptographic digital signature across the attestation, and optionally along with information in the attestation itself SHALL uniquely identify a specific TEE in a specific device.
- **Integrity of claims** - the cryptographic digital signature across the attestation SHALL cover the entire attestation including all meta data and all the claims in the attestation, ensuring that the attestation has not be modified since the TEE signed the attestation.

- **Assumptions**

- Regarding **the quality of the attestation** and the quality and security provided by the TEE, the device, the manufacturer, or others involved in the device or TEE ecosystem. Assumptions include:
 - the security measures a manufacturer takes when provisioning keys into devices/TEEs;
 - what hardware and software components have access to the Attestation keys of the TEE;
 - to the source or local verification of claims within an attestation prior to a TEE signing a set of claims;
 - the level of protection afforded to attestation keys against exfiltration, modification, and side channel attacks;
 - the limitations of use applied to TEE Attestation keys;
 - the processes in place to discover or detect TEE breeches; and
 - the revocation and recovery process of TEE attestation keys.
- TAMs and SPs must be comfortable with the assumptions that are inherently part of any attestation they accept.
- Alternatively, any TAM or SP may choose not to accept an attestation generated from a particular manufacturer or device's TEE based on the inherent assumptions.

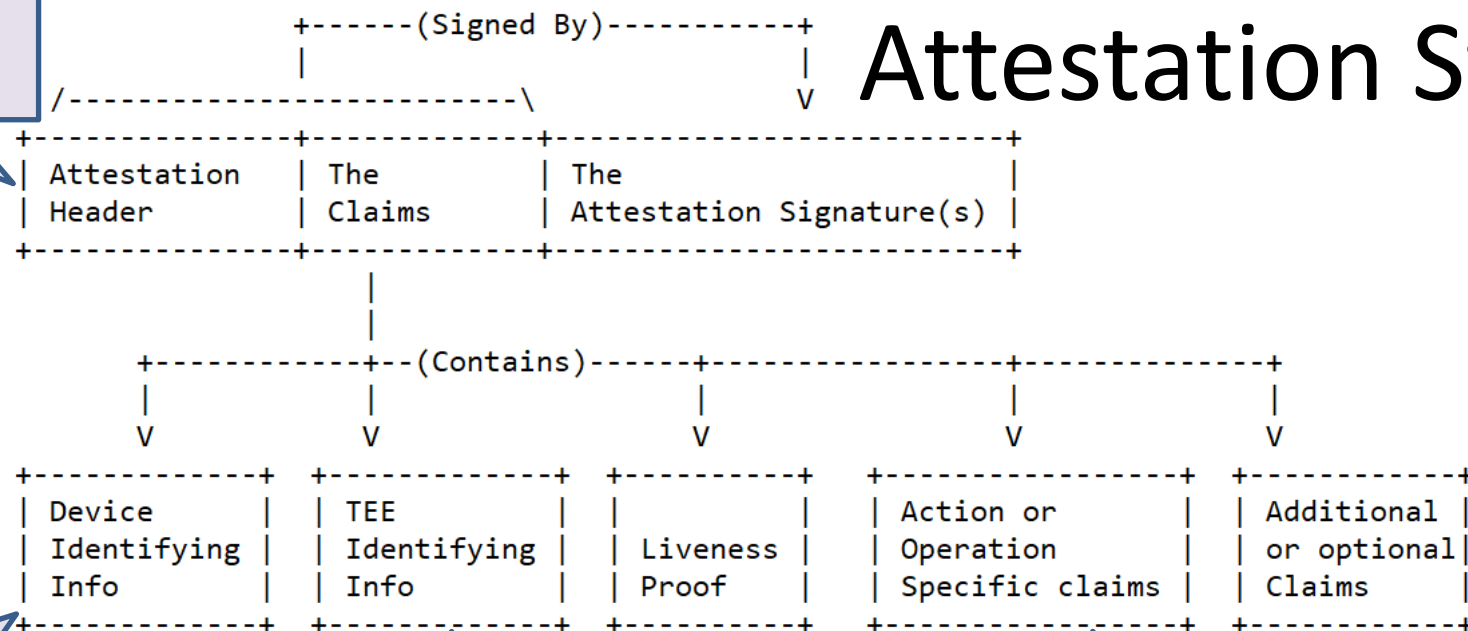
TEEP Standard Attestation Signatures

- Asymmetric Algorithms include larger sized keys to meet NIST recommendations for products that will exist beyond a 2030 horizon
 - RSA-2048 with SHA-256 or SHA-384 in RSASSA-PKCS1-v1_5 or PSS format
 - **RSA-3072 with SHA-256 or SHA-384 in RSASSA-PKCS1-v1_5 or PSS format**
 - **ECDSA-256 using NIST P256 curve using SHA-256**
 - ECDSA-384 using NIST P384 curve using SHA-384
 - **HashEdDSA using Ed25519 with SHA-512 (Ed25519ph in RFC8032) and context="TEEP Attestation"**
 - EdDSA using Ed448 with SHAK256 (Ed448ph in RFC8032) and context="TEEP Attestation"

Recommended

Attestation Structure

Attestation Type
Signature Type
Version Number



Manufacturer and
Device Unique
Identifiers

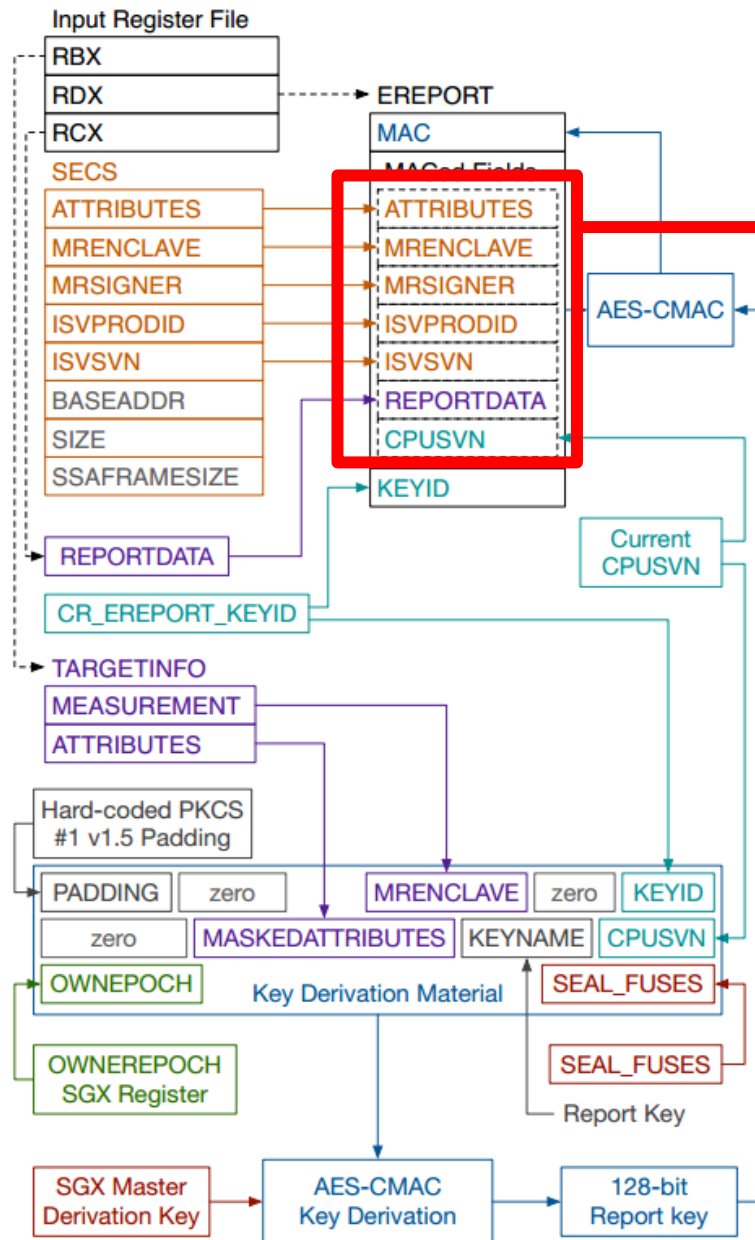
TEE Manufacturer and
TEE Type and
Version Numbers

Nonce
And/or
Timestamp

Claims based on
Attestation Type
In Header

Optional Claims
required by TEE Type
or required by
Requestor

SGX Attestation Elements



Required Attributes to validate an SGX TEE & Device DEVICE IDENTIFYING INFO

- **ATTRIBUTES** – system attributes affecting TEE (debug enable, architectural features & extensions like AVX instructions, FPU, etc.)
- **CPUSVN** – Device/TEE security version number identifies security release of SGX

TEE IDENTIFYING INFO

- **MRENCLAVE** – Hash of the TA code
- **MRSIGNER** – Hash of the TA signing key
- **ISVPRODID** – Independent SW Vendor's TA Product Identifier, uniquely identifies the TA
- **ISVSVN** – ISV's security version number increments only on security updates

LINK TO OTHER CLAIMS IN ATTESTATION

- **REPORTDATA** – other attestation elements

These elements must be in an SGX Attestation, in order to evaluate the code in the enclave and the SGX instance for trust.

Attestation Work still to Complete

- Update format based on feedback
- Provide clear direction for the mapping of Device, TEE, and TA attributes in the format
- Provide formats for TEEP standard claims
- Provide examples of real attestation (suggest SGX and ARM TZ)

WORK IN PROGRESS AND OPEN DISCUSSION

#7: Clarify meaning of Security Domain (SD)

- Problem
 - Concern of overhead to manage SDs
- Proposal
 - Proposed to allow implicit SD per TA
 - Alternate Proposal is to remove the SD altogether and require the TEEP Broker/Agent to implement SDs if the underlying TEE requires it
- Background
 - What is the SD actually needed for?
 - What is the SD's associated SP AIK (Anonymous Attestation Key) needed for?
- Issues
 - Do we really need separate SP AIK?
 - Do we really have a privacy concern?

Issue #7 – Security Domain

- From TEEP: “A Security Domain (SD) concept is used as the security boundary inside a TEE for trusted applications”.
- The introduction of the Security Domain concept has also side effects on the protocol specification:
 - First, it **requires separate messaging** for creating and deleting security domains.
 - Second, it **requires added protocol elements** to reference security domains.
 - Third, the **Anonymous Attestation Key (AAK) is established** and maintained for use with security domains.

Issue #7 – Security Domain

Since IETF 103, the following condensed summary of the issue was provided by Hannes:

The options for Security Domain inclusion in TEEP can be one of the following:

1. Make Security Domain **mandatory** for all exchanges (as it is in current protocol)
2. Make Security Domain **implicit**, using a default (new) security domain created for each app and the requirement to provision a public key per Security Domain.
3. Make Security Domain **optional**, and defaulting to a single TEE attestation key, which avoids the key provisioning round-trip.
4. **Remove concept** of Security Domain from the document

This boils down to defining what a Security Domain is really used for:

- **is it a management component**
- **Is it an isolation mechanism**
- **Is it a key provisioning mechanism**

#7: Clarify meaning of Security Domain (SD) Cont.

- Issue 1: why not single AIK per device instead of one per SP?
 - For SP initiated data encryption to a device, the same key provides confidentiality. Source of proof can use a signing etc.
 - Separate SP AIK for getTAInformation response provides privacy protection, mitigating a Client App from collecting TEE certificates.
- Issue 2: is TEE certificate as device identity a privacy concern?
 - Argument 1: a device already has unique hardware ID, a MAC address etc. Rich OS can hide.
 - Argument 2: this will require Rich OS involved to support hiding for TEE API calls. Also Rich OS was considered not at the same security level as TEE for this.
- Central question
 - Should TEE certificates be accessible and exposed to any Client App that's allowed to talk to the TEE?

Issue #7 – Argument Against Security Domain

Use a separate AIK may not be necessary. The TEE attestation key could be used for the key wrapping if we agreed that the separate SP AIK isn't necessary.

The install TA message should know the SP that is being linked, because:

- Client App requests install using a manifest that includes the SP's public key
- The TAM sends the indication to install the TA, and provides the SP's public key

RECOMMENDATION:

Remove Security Domain from the protocol, and create security mechanisms that provide the required security protections at the right place in the protocol and can work under any conceived scenario.

#18 Trust Anchor Update

- Trust Anchor update must be considered for the completeness of the Trust Anchor lifecycle management
- Two options
 - Part of architecture draft, synchronized with the SUIT definitions
 - A separate draft work for the full definition of the Trust Anchor lifecycle (creation/provisioning, use, update)
- Current preference
 - Defer complete definition for a separate draft document, but provide basic definitions aligned to SUIT and the use of the Trust Anchors in the architecture document.
 - A solution discussed was to use a system Manager TA pre-installed in a TEE for check and update of Trust Anchors
- A related question
 - Trust Anchor format: leave it to TEE implementation or define it in TEEP?
 - If defined in TEEP a very comprehensive document with many implementation options must be provided (including fuses, one-time-programmable bits (OTP), locked flash, battery backed RAM, PUFs, etc)

#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
- David T new draft content
 - To be merged back to the core protocol document

#10: Local TEE signing first

- ISSUE:
 - One proposal was put forward to make the TEE connect to the TAM using an attestation of the platform and include any “installTA” requests in the message
 - The objection was stated as: Local TEE signing first would leak the TEE signing key to potentially unknown TAM
- DISCUSSION
 - However, this can be solved by the TEEP Agent/Broker only sending such information to TAMs that are in the Trust Anchor list
 - How does the TEEP Agent know the TAM is trusted?
 - The Manifest must include TAM information and the TAM keys
 - The TAM info in the manifest is signed with the TAM’s private key and can be verified by the TEEP Agent in the TEE.

#12: Every Rich App talks to TAM?

- Not necessary
- Options Include:
 - Adopt support of metadata file and installer (as a service) in REE to initiate Broker contact with a TAM
 - System library that can be statically/dynamically linked to the Rich App to provide TAM communication and message parsing
 - Also discuss the use of an Installer Application.
- See also #9

Q&A

Thank you!