TEEP
Internet-Draft
Intended status: Informational
Expires: January 9, 2020

M. Pei
Symantec
H. Tschofenig
Arm Limited
D. Wheeler
Intel
A. Atyeo
Intercede
L. Dapeng
Alibaba Group
July 08, 2019

Trusted Execution Environment Provisioning (TEEP) Architecture
draft-ietf-teep-architecture-03

Abstract

A Trusted Execution Environment (TEE) is designed to provide a hardware-isolation mechanism to separate a regular operating system from security-sensitive application components.

This architecture document motivates the design and standardization of a protocol for managing the lifecycle of trusted applications running inside a TEE.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 9, 2020.

Table of Contents

1. Introduction

Applications executing in a device are exposed to many different attacks intended to compromise the execution of the application, or reveal the data upon which those applications are operating. These attacks increase with the number of other applications on the device, with such other applications coming from potentially untrustworthy sources. The potential for attacks further increase with the complexity of features and applications on devices, and the unintended interactions among those features and applications. The danger of attacks on a system increases as the sensitivity of the applications or data on the device increases. As an example, exposure of emails from a mail client is likely to be of concern to its owner, but a compromise of a banking application raises even greater concerns.

The Trusted Execution Environment (TEE) concept is designed to execute applications in a protected environment that separates applications inside the TEE from the regular operating system and from other applications on the device. This separation reduces the possibility of a successful attack on application components and the data contained inside the TEE. Typically, application components are chosen to execute inside a TEE because those application components perform security sensitive operations or operate on sensitive data. An application component running inside a TEE is referred to as a Trusted Application (TA), while a normal application running in the regular operating system is referred to as an Untrusted Application (UA).

The TEE uses hardware to enforce protections on the TA and its data, but also presents a more limited set of services to applications inside the TEE than is normally available to UA's running in the normal operating system.

But not all TEEs are the same, and different vendors may have different implementations of TEEs with different security properties, different features, and different control mechanisms to operate on TAs. Some vendors may themselves market multiple different TEEs with different properties attuned to different markets. A device vendor may integrate one or more TEEs into their devices depending on market needs.

To simplify the life of developers and service providers interacting with TAs in a TEE, an interoperable protocol for managing TAs running in different TEEs of various devices is needed. In this TEE ecosystem, there often arises a need for an external trusted party to verify the identity, claims, and rights of Service Providers(SP), devices, and their TEEs. This trusted third party is the Trusted Application Manager (TAM).

This protocol addresses the following problems:

- A Service Provider (SP) intending to provide services through a TA to users of a device needs to determine security-relevant information of a device before provisioning their TA to the TEE within the device. Examples include the verification of the device 'root of trust' and the type of TEE included in a device.

- A TEE in a device needs to determine whether a Service Provider (SP) that wants to manage a TA in the device is authorized to manage TAs in the TEE, and what TAs the SP is permitted to manage.

- The parties involved in the protocol must be able to attest that a TEE is genuine and capable of providing the security protections required by a particular TA.

- A Service Provider (SP) must be able to determine if a TA exists (is installed) on a device (in the TEE), and if not, install the TA in the TEE.

- A Service Provider (SP) must be able to check whether a TA in a device's TEE is the most up-to-date version, and if not, update the TA in the TEE.

- A Service Provider (SP) must be able to remove a TA in a device's TEE if the SP is no longer offering such services or the services are being revoked from a particular user (or device). For example, if a subscription or contract for a particular service has expired, or a payment by the user has not been completed or has been rescinded.

- A Service Provider (SP) must be able to define the relationship between cooperating TAs under the SP's control, and specify whether the TAs can communicate, share data, and/or share key material.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are used:

- Client Application: An application running in a Rich Execution Environment, such as an Android, Windows, or iOS application. We sometimes refer to this as the 'Client App'.

- Device: A physical piece of hardware that hosts a TEE along with a Rich Execution Environment. A Device contains a default list of Trust Anchors that identify entities (e.g., TAMs) that are trusted by the Device. This list is normally set by the Device Manufacturer, and may be governed by the Device's network carrier. The list of Trust Anchors is normally modifiable by the Device's owner or Device Administrator. However the Device manufacturer and network carrier may restrict some modifications, for example, by not allowing the manufacturer or carrier's Trust Anchor to be removed or disabled.

- Rich Execution Environment (REE): An environment that is provided and governed by a typical OS (e.g., Linux, Windows, Android, iOS), potentially in conjunction with other supporting operating systems and hypervisors; it is outside of the TEE. This environment and applications running on it are considered un-trusted.

- Service Provider (SP): An entity that wishes to provide a service on Devices that requires the use of one or more Trusted Applications. A Service Provider requires the help of a TAM in order to provision the Trusted Applications to remote devices.

- Device User: A human being that uses a device. Many devices have a single device user. Some devices have a primary device user with other human beings as secondary device users (e.g., parent allowing children to use their tablet or laptop). Relates to Device Owner and Device Administrator.

- Device Owner: A device is always owned by someone. It is common for the (primary) device user to also own the device, making the device user/owner also the device administrator. In enterprise environments it is more common for the enterprise to own the device, and device users have no or limited administration rights. In this case, the enterprise appoints a device administrator that is not the device owner.

- Device Administrator (DA): An entity that is responsible for administration of a Device, which could be the device owner. 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 can manage the list of allowed TAMs by modifying the list of Trust Anchors on the Device. Although a Device Administrator may have privileges and Device-specific controls to locally administer a device, the Device Administrator may choose to remotely administrate a device through a TAM.

- 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 along with additional data if necessary such as its public key algorithm and parameters. The Trust Anchor is normally stored in a location that resists unauthorized modification, insertion, or replacement. The digital fingerprint of a Trust Anchor may be stored along with the Trust Anchor certificate or public key. A device can use the fingerprint to uniquely identify a Trust Anchor. The Trust Anchor private key owner can sign certificates of other public keys, which conveys trust about those keys to the

device. A certificate signed by the Trust Anchor communicates that the private key holder of the signed certificate is trusted by the Trust Anchor holder, and can therefore be trusted by the device. Trust Anchors in a device may be updated by an authorized party when a Trust Anchor should be deprecated or a new Trust Anchor should be added.

- Trusted Application (TA): An application component that runs in a TEE.

- Trusted Execution Environment (TEE): An execution environment that runs alongside of, but is isolated from, an REE. A TEE has security capabilities and meets certain security-related requirements. It protects TEE assets from general software attacks, defines rigid safeguards as to data and functions that a program can access, and resists a set of defined threats. It should have at least the following three properties:

  (a) A device unique credential that cannot be cloned;

  (b) Assurance that only authorized code can run in the TEE;

  (c) Memory that cannot be read by code outside the TEE.

  There are multiple technologies that can be used to implement a TEE, and the level of security achieved varies accordingly.

- 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. Examples of RoTs include software/firmware measurement and verification using a Trust Anchor (RoT for Verification), provide signed assertions using a protected attestation key (RoT for Reporting), or protect the storage and/or use of cryptographic keys (RoT for Storage). Other RoTs are possible, including RoT for Integrity, and RoT for Measurement. Reference: NIST SP800-164 (Draft).

- 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.

This document uses the following abbreviations:

- CA: Certificate Authority
- REE: Rich Execution Environment
- RoT: Root of Trust
- SD: Security Domain
- SP: Service Provider
- TA: Trusted Application
- TAM: Trusted Application Manager
- TEE: Trusted Execution Environment
- TFW: Trusted Firmware

3. Assumptions

This specification assumes that an applicable device is equipped with one or more TEEs and each TEE is pre-provisioned with a device-unique public/private key pair, which is securely stored.

A TEE uses an isolation mechanism between Trusted Applications to ensure that one TA cannot read, modify or delete the data and code of another TA.

4. Use Cases

4.1. Payment

A payment application in a mobile device requires high security and trust about the hosting device. Payments initiated from a mobile device can use a Trusted Application to provide strong identification and proof of transaction.

For a mobile payment application, some biometric identification information could also be stored in a TEE. The mobile payment application can use such information for authentication.

A secure user interface (UI) may be used in a mobile device to prevent malicious software from stealing sensitive user input data. Such an application implementation often relies on a TEE for user input protection.

4.2. Authentication

For better security of authentication, a device may store its sensitive authentication keys inside a TEE, providing hardware-protected security key strength and trusted code execution.

4.3. Internet of Things

The Internet of Things (IoT) has been posing threats to networks and national infrastructures because of existing weak security in devices. It is very desirable that IoT devices can prevent malware from manipulating actuators (e.g., unlocking a door), or stealing or modifying sensitive data such as authentication credentials in the device. A TEE can be the best way to implement such IoT security functions.

TEEs could be used to store variety of sensitive data for IoT devices. For example, a TEE could be used in smart door locks to store a user's biometric information for identification, and for protecting access the locking mechanism.

4.4. Confidential Cloud Computing

A tenant can store sensitive data in a TEE in a cloud computing server such that only the tenant can access the data, preventing the cloud hosting provider from accessing the data. A tenant can run TAs inside a server TEE for secure operation and enhanced data security. This provides benefits not only to tenants with better data security but also to cloud hosting provider for reduced liability and increased cloud adoption.

5. Architecture

5.1. System Components

The following are the main components in the system. Full descriptions of components not previously defined are provided below. Interactions of all components are further explained in the following paragraphs.

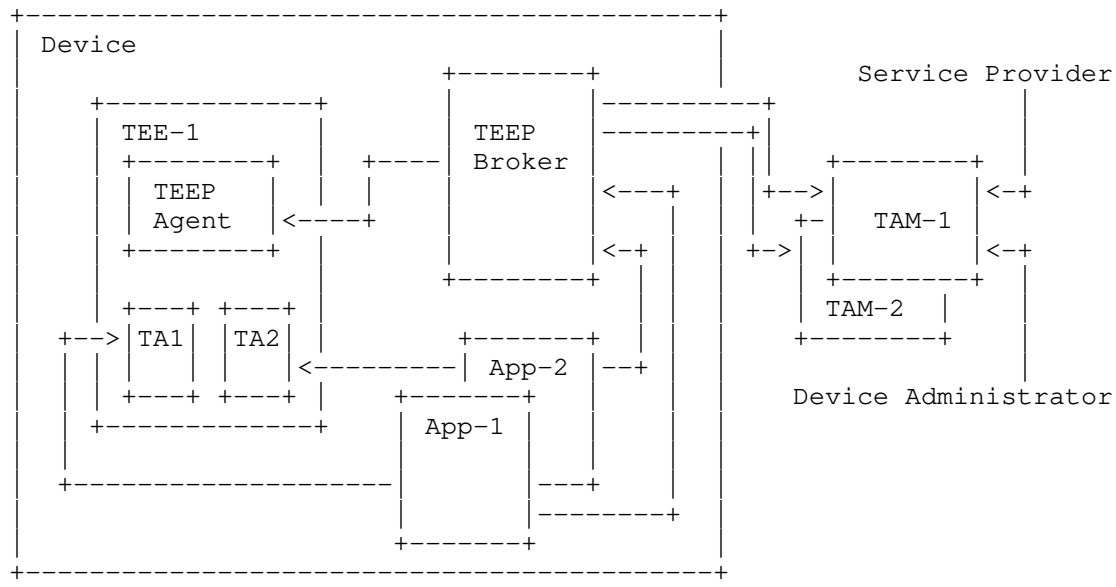

Figure 1: Notional Architecture of TEEP

- Service Providers (SP) and Device Administrators (DA) utilize the services of a TAM to manage TAs on Devices. SPs do not directly interact with devices. DAs may elect to use a TAM for remote administration of TAs instead of managing each device directly.

- TAM: A TAM is responsible for performing lifecycle management activity on TA's on behalf of Service Providers and Device Administrators. This includes creation and deletion of TA's, and may include, for example, over-the-air updates to keep an SP's TAs up-to-date and clean up when a version should be removed. TAMs may provide services that make it easier for SPs or DAs to use the TAM's service to manage multiple devices, although that is not required of a TAM.

  The TAM performs its management of TA's through an interaction with a Device's TEEP Broker. As shown in #notionalarch, the TAM cannot directly contact a Device, but must wait for a the TEEP Broker or a Client Application to contact the TAM requesting a particular service. This architecture is intentional in order to accommodate network and application firewalls that normally protect user and enterprise devices from arbitrary connections from external network entities.

  A TAM may be publicly available for use by many SPs, or a TAM may be private, and accessible by only one or a limited number of SPs.

It is expected that manufacturers and carriers will run their own private TAM. Another example of a private TAM is a TAM running as a Software-as-a-Service (SaaS) within an SP.

A SP or Device Administrator chooses a particular TAM based on whether the TAM is trusted by a Device or set of Devices. The TAM is trusted by a device if the TAM's public key is an authorized Trust Anchor in the Device. A SP or Device Administrator may run their own TAM, however the Devices they wish to manage must include this TAM's pubic key in the Trust Anchor list.

A SP or Device Administrator is free to utilize multiple TAMs. This may be required for a SP to manage multiple different types of devices from different manufacturers, or devices on different carriers, since the Trust Anchor list on these different devices may contain different TAMs. A Device Administrator may be able to add their own TAM's public key or certificate to the Trust Anchor list on all their devices, overcoming this limitation.

Any entity is free to operate a TAM. For a TAM to be successful, it must have its public key or certificate installed in Devices Trust Anchor list. A TAM may set up a relationship with device manufacturers or carriers to have them install the TAM's keys in their device's Trust Anchor list. Alternatively, a TAM may publish its certificate and allow Device Administrators to install the TAM's certificate in their devices as an after-market-action.

- TEEP Broker: The TEEP Broker is an application running in a Rich Execution Environment (REE) that enables the message protocol exchange between a TAM and a TEE in a device. The TEEP Broker does not process messages on behalf of a TEE, but merely is responsible for relaying messages from the TAM to the TEE, and for returning the TEE's responses to the TAM.

  A Client Application is expected to communicate with a TAM to request TAs that it needs to use. The Client Application needs to pass the messages from the TAM to TEEs in the device. This calls for a component in the REE that Client Applications can use to pass messages to TEEs. The TEEP Broker is thus an application in the REE or software library that can relay messages from a Client Application to a TEE in the device. A device usually comes with only one active TEE. A TEE may provide such a Broker to the device manufacturer to be bundled in devices. Such a TEE must also include a Broker counterpart, namely, a TEEP Agent inside the TEE, to parse TAM messages sent through the Broker. A TEEP Broker is generally acting as a dummy relaying box with just the TEE interacting capability; it doesn't need and shouldn't parse protocol messages.

- TEEP Agent: the TEEP Agent is a processing module running inside a TEE that receives TAM requests that are relayed via a TEEP Broker that runs in an REE. A TEEP Agent in the TEE may parse requests or forward requests to other processing modules in a TEE, which is up to a TEE provider's implementation. A response message corresponding to a TAM request is sent by a TEEP Agent back to a TEEP Broker.

- Certification Authority (CA): Certificate-based credentials used for authenticating a device, a TAM and an SP. A device embeds a list of root certificates (Trust Anchors), from trusted CAs that a TAM will be validated against. A TAM will remotely attest a device by checking whether a device comes with a certificate from a CA that the TAM trusts. The CAs do not need to be the same; different CAs can be chosen by each TAM, and different device CAs can be used by different device manufacturers.

5.2. Different Renditions of TEEP Architecture

There is nothing prohibiting a device from implementing multiple TEEs. In addition, some TEEs (for example, SGX) present themselves as separate containers within memory without a controlling manager within the TEE. In these cases, the rich operating system hosts multiple TEEP brokers, where each broker manages a particular TEE or set of TEEs. Enumeration and access to the appropriate broker is up to the rich OS and the applications. Verification that the correct TA has been reached then becomes a matter of properly verifying TA attestations, which are unforgeable. The multiple TEE approach is shown in the diagram below. For brevity, TEEP Broker 2 is shown interacting with only one TAM and UA, but no such limitation is intended to be implied in the architecture.

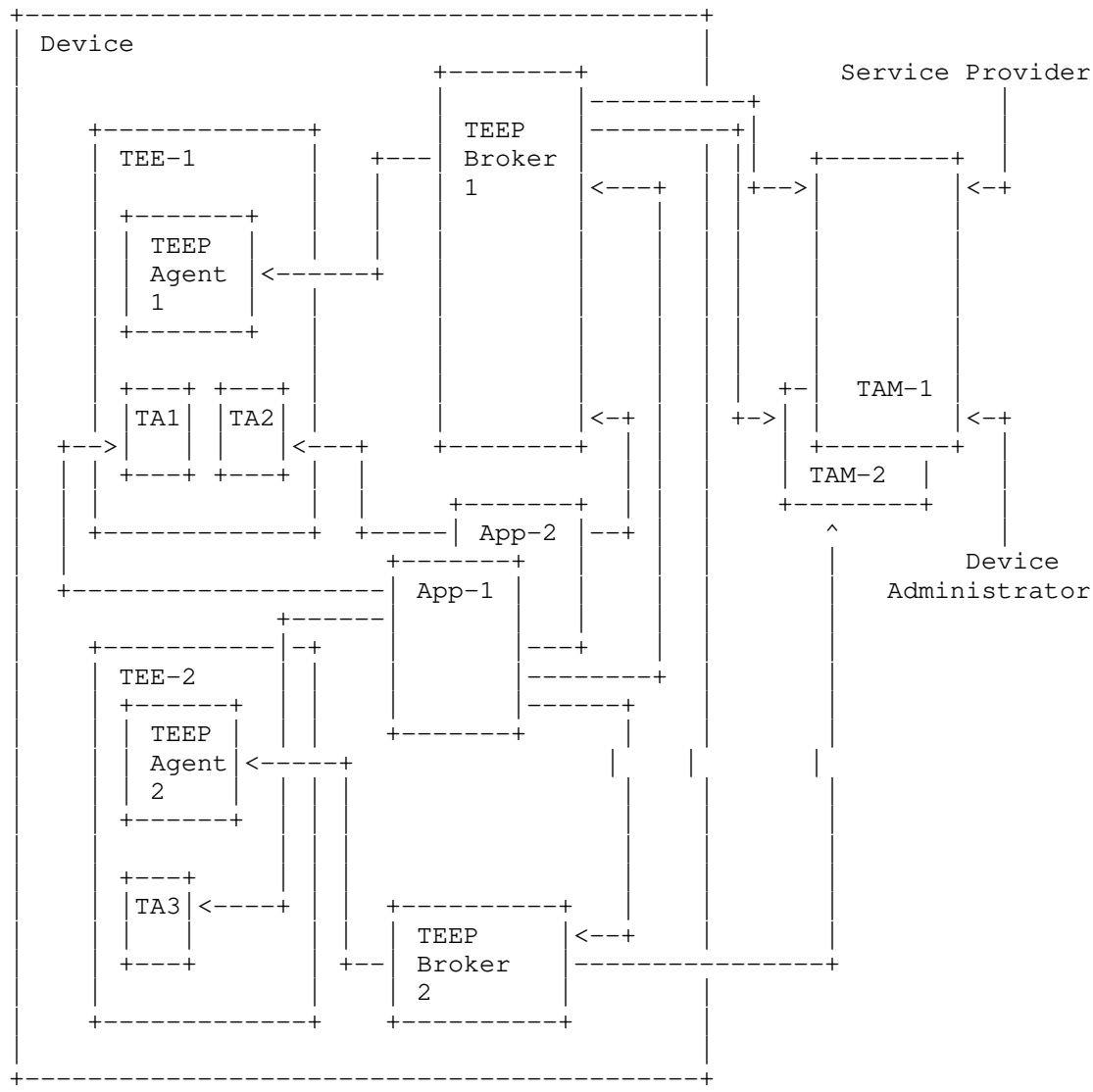

Figure 2: Notional Architecture of TEEP wtih multiple TEEs

In the diagram above, TEEP Broker 1 controls interactions with the TA's in TEE-1, and TEEP Broker 2 controls interactions with the TA's in TEE-2. This presents some challenges for a TAM in completely managing the device, since a TAM may not interact with all the TEEP Brokers on a particular platform. In addition, since TEE's may be physically separated, with wholly different resources, there may be no need for TEEP Brokers to share information on installed TAs or

resource usage. However, the architecture guarantees that the TAM will receive all the relevant information from the TEEP Broker to which it communicates.

5.3. Multiple TAMs and Relationship to TAs

As shown in Figure 2, the TEEP Broker provides connections from the TEE and the Client App to one or more TAMs. The selection of which TAM to communicate with is dependent on information from the Client App and is directly related to the TA.

When a SP offers a service which requires a TA, the SP associates that service with a specific TA. The TA itself is digitally signed, protecting its integrity, but the signature also links the TA back to the signer. The signer is usually the SP, but in some cases may be another party that the SP trusts. The SP selects one or more TAMs through which to offer their service, and communicates the information of the service and the specific client apps and TAs to the TAM.

The SP chooses TAMs based upon the markets into which the TAM can provide access. There may be TAMs that provide services to specific types of mobile devices, or mobile device operating systems, or specific geographical regions or network carriers. A SP may be motivated to utilize multiple TAMs for its service in order to maximize market penetration and availability on multiple types of devices. This likely means that the same service will be available through multiple TAMs.

When the SP publishes the Client App to an app store or other app repositories, the SP binds the Client App with a manifest that identifies what TAMs can be contacted for the TA. In some situations, an SP may use only a single TAM - this is likely the case for enterprise applications or SPs serving a closed community. For broad public apps, there will likely be multiple TAMs in the manifest - one servicing one brand of mobile device and another servicing a different manufacturer, etc. Because different devices and different manufacturers trust different TAMs, the manifest will include different TAMs that support this SP's client app and TA. Multiple TAMs allow the SP to provide thier service and this app (and TA) to multiple different devices.

When the TEEP Broker receives a request to contact the TAM for a Client App in order to install a TA, a list of TAMs may be provided. The TEEP Broker selects a single TAM that is consistent with the list of trusted TAMs (trust anchors) provisioned on the device. For any client app, there should be only a single TAM for the TEEP Broker to contact. This is also the case when a Client App uses multiple TAs,

or when one TA depends on anther TA in a software dependency (see section TBD). The reason is that the SP should provide each TAM that it places in the Client App's manifest all the TAs that the app requires. There is no benefit to going to multiple different TAMs, and there is no need for a special TAM to be contacted for a specific TA.

[Note: This should always be the case. When a particular device or TEE supports only a special proprietary attestation mechanism, then a specific TAM will be needed that supports that attestation scheme. The TAM should also support standard atttestation signatures as well. It is highly unlikely that a set of TAs would use different proprietary attestation mechanisms since a TEE is likley to support only one such proprietary scheme.]

[Note: This situation gets more complex in situations where a Client App expects another application or a device to already have a specific TA installed. This situation does not occur with SGX, but could occur in situations where the secure world maintains an trusted operating system and runs an entire trusted system with multiple TAs running. This requires more discussion.]

5.4. Client Apps, Trusted Apps, and Personalization Data

In TEEP, there is an explicit relationship and dependence between the client app in the REE and one or more TAs in the TEE, as shown in Figure 2. From the perspective of a device user, a client app that uses one or more TA's in a TEE appears no different from any other untrusted application in the REE. However, the way the client app and its corresponding TA's are packaged, delivered, and installed on the device can vary. The variations depend on whether the client app and TA are bundled together or are provided separately, and this has implications to the management of the TAs in the TEE. In addition to the client app and TA, the TA and/or TEE may require some additional data to personalize the TA to the service provider or the device user. This personalization data is dependent on the TEE, the TA and the SP; an example of personalization data might be username and password of the device user's account with the SP, or a secret symmetric key used to by the TA to communicate with the SP. The personalization data must be encrypted to preserve the confidentiality of potentially sensitive data contained within it. Other than this requirement to support confidentiality, TEEP place no limitations or requirements on the personalization data.

There are three possible cases for bundling of the Client App, TA, and personalization data:

1. The Client App, TA, and personalization data are all bundled together in a single package by the SP and provided to the TEEP Broker through the TAM.

2. The Client App and the TA are bundled together in a single binary, which the TAM or a publicly accessible app store maintains in repository, and the personalization data is separately provided by the SP. In this case, the personalization data is collected by the TAM and included in the InstallTA message to the TEEP Broker.

3. All components are independent. The device user installs the Client App through some independent or device-specific mechanism, and the TAM provides the TA and personalization data from the SP. Delivery of the TA and personalization data may be combined or separate.

5.5. Examples of Application Delivery Mechanisms in Existing TEEs

In order to better understand these cases, it is helpful to review actual implementations of TEEs and their application delivery mechanisms.

In Intel Software Guard Extensions (SGX), the Client App and TA are typically bound into the same binary (Case 2). The TA is compiled into the Client App binary using SGX tools, and exists in the binary as a shared library (.so or .dll). The Client App loads the TA into an SGX enclave when the client needs the TA. This organization makes it easy to maintain compatibility between the Client App and the TA, since they are updated together. It is entirely possible to create a Client App that loads an external TA into an SGX enclave and use that TA (Case 3). In this case, the Client App would require a reference to an external file or download such a file dynamically, place the contents of the file into memory, and load that as a TA. Obviously, such file or downloaded content must be properly formatted and signed for it to be accepted by the SGX TEE. In SGX, for Case 2 and Case 3, the personalization data is normally loaded into the SGX enclave (the TA) after the TA has started. Although Case 1 is possible with SGX, there are no instances of this known to be in use at this time, since such a construction would required a special installation program and SGX TA to recieve the encrypted binary, decrypt it, separate it into the three different elements, and then install all three. This installation is complex, because the Client App decrypted inside the TEE must be passed out of the TEE to an installer in the REE which would install the Client App; this assumes that the Client App binary includes the TA code also, otherwise there is a significant problem in getting the SGX encalve code (the TA) from the TEE, through the installer and into the Client App in a trusted fashion. Finally, the

personalization data would need to be sent out of the TEE (encrypted in an SGX encalve-to-enclave manner) to the REE's installation app, which would pass this data to the installed Client App, which would in turn send this data to the SGX enclave (TA). This complexity is due to the fact that each SGX enclave is separate and does not have direct communication to one another.

[NOTE: Need to add an equivalent discussion for an ARM/TZ implementation]

5.6. TEEP Architectural Support for Client App, TA, and Personalization Data Delivery

This section defines TEEP support for the three different cases for delivery of the Client App, TA, and personalization data.

[Note: discussion of format of this single binary, and who/what is responsible for splitting these things apart, and installing the client app into the REE, the TA into the TEE, and the personalization data into the TEE or TA. Obviously the decryption must be done by the TEE but this may not be supported by all TAs.]

5.7. Entity Relations

This architecture leverages asymmetric cryptography to authenticate a device to a TAM. Additionally, a TEE in a device authenticates a TAM and TA signer. The provisioning of Trust Anchors to a device may different from one use case to the other. A device administrator may want to have the capability to control what TAs are allowed. A device manufacturer enables verification of the TA signers and TAM providers; it may embed a list of default Trust Anchors that the signer of an allowed TA's signer certificate should chain to. A device administrator may choose to accept a subset of the allowed TAs via consent or action of downloading.

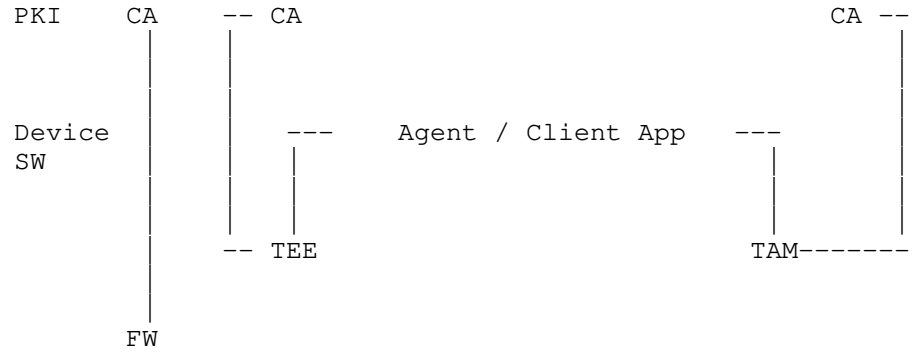

Figure 3: Entities

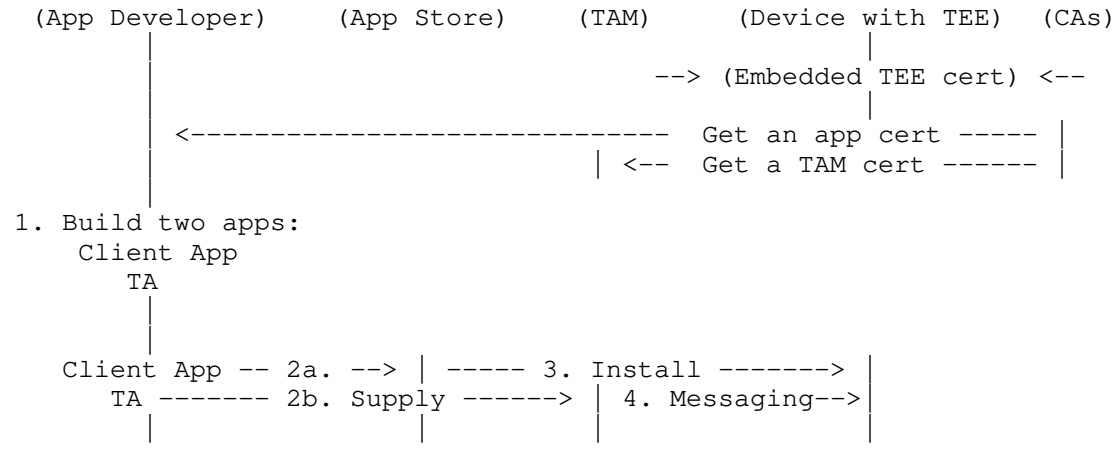

Figure 4: Developer Experience

Figure 4 shows an application developer building two applications: 1) a rich Client Application; 2) a TA that provides some security functions to be run inside a TEE. At step 2, the application developer uploads the Client Application (2a) to an Application Store. The Client Application may optionally bundle the TA binary. Meanwhile, the application developer may provide its TA to a TAM provider that will be managing the TA in various devices. 3. A user will go to an Application Store to download the Client Application. The Client Application will trigger TA installation by initiating communication with a TAM. This is the step 4. The Client Application will get messages from TAM, and interacts with device TEE via an Agent.

The following diagram shows a system diagram about the entity relationships between CAs, TAMs, SPs and devices.

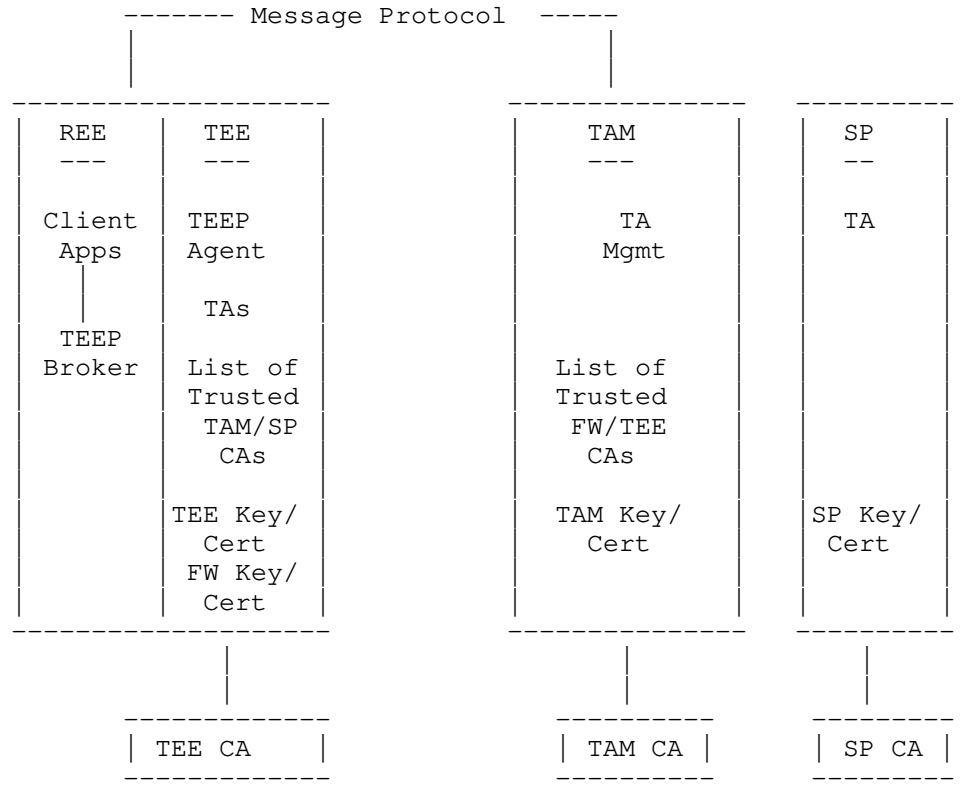

Figure 5: Keys

In the previous diagram, different CAs can be used for different types of certificates. Messages are always signed, where the signer key is the message originator's private key such as that of a TAM, the private key of trusted firmware (TFW), or a TEE's private key.

The main components consist of a set of standard messages created by a TAM to deliver TA management commands to a device, and device attestation and response messages created by a TEE that responds to a TAM's message.

It should be noted that network communication capability is generally not available in TAs in today's TEE-powered devices. The networking functionality must be delegated to a rich Client Application. Client Applications will need to rely on an agent in the REE to interact with a TEE for message exchanges. Consequently, a TAM generally communicates with a Client Application about how it gets messages that originate from a TEE inside a device. Similarly, a TA or TEE generally gets messages from a TAM via some Client Application,

namely, a TEEP Broker in this protocol architecture, not directly from the network.

It is imperative to have an interoperable protocol to communicate with different TAMs and different TEEs in different devices. This is the role of the Broker, which is a software component that bridges communication between a TAM and a TEE. Furthermore the Broker communicates with a Agent inside a TEE that is responsible to process TAM requests. The Broker in REE does not need to know the actual content of messages except for the TEE routing information.

5.8. Trust Anchors in TEE

Each TEE comes with a trust store that contains a whitelist of Trust Anchors that are used to validate a TAM's certificate. A TEE will accept a TAM to create new Security Domains and install new TAs on behalf of an SP only if the TAM's certificate is chained to one of the root CA certificates in the TEE's trust store.

A TEE's trust store is typically preloaded at manufacturing time. It is out of the scope in this document to specify how the trust anchors should be updated when a new root certificate should be added or existing one should be updated or removed. A device manufacturer is expected to provide its TEE trust anchors live update or out-of-band update to Device Administrators.

When trust anchor update is carried out, it is imperative that any update must maintain integrity where only authentic trust anchor list from a device manufacturer or a Device Administrator is accepted. This calls for a complete lifecycle flow in authorizing who can make trust anchor update and whether a given trust anchor list are non-tampered from the original provider. The signing of a trust anchor list for integrity check and update authorization methods are desirable to be developed. This can be addressed outside of this architecture document.

Before a TAM can begin operation in the marketplace to support a device with a particular TEE, it must obtain a TAM certificate from a CA that is listed in the trust store of the TEE.

5.9. Trust Anchors in TAM

The Trust Anchor store in a TAM consists of a list of CA certificates that sign various device TEE certificates. A TAM will accept a device for TA management if the TEE in the device uses a TEE certificate that is chained to a CA that the TAM trusts.

5.10. Keys and Certificate Types

This architecture leverages the following credentials, which allow delivering end-to-end security without relying on any transport security.

Key Entity Name	Location	Issuer	Checked Against	Cardinality
1. TFW key pair and certificate	Device secure storage	FW CA	A whitelist 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 whitelist 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 whitelist 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 TA as the owner.	1 or multiple can be used by a TAM

Figure 6: Key and Certificate Types

1. TFW key pair and certificate: A key pair and certificate for evidence of trustworthy firmware in a device. This key pair is optional for TEEP architecture. Some TEE may present its trusted attributes to a TAM using signed attestation with a TFW key. For example, a platform that uses a hardware based TEE can have attestation data signed by a hardware protected TFW key.

    - Location: Device secure storage
    - Supported Key Type: RSA and ECC
    - Issuer: OEM CA

- Checked Against: A whitelist of FW root CA trusted by TAMs
- Cardinality: One per device

2. TEE key pair and certificate: It is used for device attestation to a remote TAM and SP.
   - This key pair is burned into the device by the device manufacturer. The key pair and its certificate are valid for the expected lifetime of the device.
   - Location: Device TEE
   - Supported Key Type: RSA and ECC
   - Issuer: A CA that chains to a TEE root CA
   - Checked Against: A whitelist of TEE root CAs trusted by TAMs
   - Cardinality: One per device
3. TAM key pair and certificate: A TAM provider acquires a certificate from a CA that a TEE trusts.
   - Location: TAM provider
   - Supported Key Type: RSA and ECC.
   - Supported Key Size: RSA 2048-bit, ECC P-256 and P-384. Other sizes should be anticipated in future.
   - Issuer: TAM CA that chains to a root CA
   - Checked Against: A whitelist of TAM root CAs embedded in a TEE
   - Cardinality: One or multiple can be used by a TAM
4. SP key pair and certificate: An SP uses its own key pair and certificate to sign a TA.
   - Location: SP
   - Supported Key Type: RSA and ECC
   - Supported Key Size: RSA 2048-bit, ECC P-256 and P-384. Other sizes should be anticipated in future.
   - Issuer: An SP signer CA that chains to a root CA

- Checked Against: An SP uses a TAM. A TEE trusts an SP by validating trust against a TAM that the SP uses. A TEE trusts a TAM to ensure that a TA is trustworthy.

- Cardinality: One or multiple can be used by an SP

5.11. Scalability

This architecture uses a PKI. Trust Anchors exist on the devices to enable the TEE to authenticate TAMs, and TAMs use Trust Anchors to authenticate TEEs. Since a PKI is used, many intermediate CA certificates can chain to a root certificate, each of which can issue many certificates. This makes the protocol highly scalable. New factories that produce TEEs can join the ecosystem. In this case, such a factory can get an intermediate CA certificate from one of the existing roots without requiring that TAMs are updated with information about the new device factory. Likewise, new TAMs can join the ecosystem, providing they are issued a TAM certificate that chains to an existing root whereby existing TEEs will be allowed to be personalized by the TAM without requiring changes to the TEE itself. This enables the ecosystem to scale, and avoids the need for centralized databases of all TEEs produced or all TAMs that exist.

5.12. Message Security

Messages created by a TAM are used to deliver TA management commands to a device, and device attestation and messages created by the device TEE to respond to TAM messages.

These messages are signed end-to-end and are typically encrypted such that only the targeted device TEE or TAM is able to decrypt and view the actual content.

5.13. Security Domain

No security domain (SD) is explicitly assumed in a TEE for TA management. Some TEE, for example, some TEE compliant with Global Platform (GP), may continue to choose to use SD to organize resource partition and security boundaries. It is up to a TEE implementation to decide how a SD is attached to a TA installation, for example, one SD could be created per TA.

5.14. A Sample Device Setup Flow

Step 1: Prepare Images for Devices

1. [TEE vendor] Deliver TEE Image (CODE Binary) to device OEM

2. [CA] Deliver root CA Whitelist

3. [Soc] Deliver TFW Image

Step 2: Inject Key Pairs and Images to Devices

1. [OEM] Generate TFW Key Pair (May be shared among multiple devices)

2. [OEM] Flash signed TFW Image and signed TEE Image onto devices (signed by TFW Key)

Step 3: Set up attestation key pairs in devices

1. [OEM] Flash TFW Public Key and a bootloader key.

2. [TFW/TEE] Generate a unique attestation key pair and get a certificate for the device.

Step 4: Set up Trust Anchors in devices

1. [TFW/TEE] Store the key and certificate encrypted with the bootloader key

2. [TEE vendor or OEM] Store trusted CA certificate list into devices

6. TEEP Broker

A TEE and TAs do not generally have the capability to communicate to the outside of the hosting device. For example, GlobalPlatform [GPTEE] specifies one such architecture. This calls for a software module in the REE world to handle the network communication. Each Client Application in the REE might carry this communication functionality but such functionality must also interact with the TEE for the message exchange.
The TEE interaction will vary according to different TEEs. In order for a Client Application to transparently support different TEEs, it is imperative to have a common interface for a Client Application to invoke for exchanging messages with TEEs.

A shared module in REE comes to meet this need. A TEEP broker is an application running in the REE of the device or an SDK that facilitates communication between a TAM and a TEE. It also provides interfaces for Client Applications to query and trigger TA installation that the application needs to use.

It isn't always that a Client Application directly calls such a Broker to interact with a TEE. A REE Application Installer might carry out TEE and TAM interaction to install all required TAs that a Client Application depends. A Client Application may have a metadata file that describes the TAs it depends on and the associated TAM that each TA installation goes to use. The REE Application Installer can inspect the application metadata file and installs TAs on behalf of the Client Application without requiring the Client Application to run first.

This interface for Client Applications or Application Installers may be commonly in a form of an OS service call for an REE OS. A Client Application or an Application Installer interacts with the device TEE and the TAMs.

6.1. Role of the TEEP Broker

A TEEP Broker abstracts the message exchanges with a TEE in a device. The input data is originated from a TAM or the first initialization call to trigger a TA installation.

The Broker doesn't need to parse a message content received from a TAM that should be processed by a TEE. When a device has more than one TEE, one TEEP Broker per TEE could be present in REE. A TEEP Broker interacts with a TEEP Agent inside a TEE.

A TAM message may indicate the target TEE where a TA should be installed. A compliant TEEP protocol should include a target TEE identifier for a TEEP Broker when multiple TEEs are present.

The Broker relays the response messages generated from a TEEP Agent in a TEE to the TAM. The Broker is not expected to handle any network connection with an application or TAM.

The Broker only needs to return an error message if the TEE is not reachable for some reason. Other errors are represented as response messages returned from the TEE which will then be passed to the TAM.

6.2. TEEP Broker Implementation Consideration

A Provider should consider methods of distribution, scope and concurrency on devices and runtime options when implementing a TEEP Broker. Several non-exhaustive options are discussed below. Providers are encouraged to take advantage of the latest communication and platform capabilities to offer the best user experience.

6.2.1. TEEP Broker Distribution

The Broker installation is commonly carried out at OEM time. A user can dynamically download and install a Broker on-demand.

6.2.2. Number of TEEP Brokers

There should be generally only one shared TEEP Broker in a device. The device's TEE vendor will most probably supply one Broker. When multiple TEEs are present in a device, one TEEP Broker per TEE may be used.

When only one Broker is used per device, the Broker provider is responsible to allow multiple TAMs and TEE providers to achieve interoperability. With a standard Broker interface, each TAM can implement its own SDK for its SP Client Applications to work with this Broker.

Multiple independent Broker providers can be used as long as they have standard interface to a Client Application or TAM SDK. Only one Broker is generally expected in a device.

7. Attestation

Attestation is the process through which one entity (an attestor) presents a series of claims to another entity (a verifier), and 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; this architecture specification does not limit the attestation claims, but defines a minimal set of claims required for TEEP to operate properly. Extensions to these claims are possible, but are not defined in the TEEP specifications. Other standards or groups may define the format and semantics of extended claims. The TEEP specification defines the claims format such that these extended claims may be easily included in a TEEP attestation message.

As of the writing of this specification, device and TEE attestations have not been standardized across the market. Different devices, manufacturers, and TEEs support different attestation algorithms and mechanisms. In order for TEEP to be inclusive, the attestation

format shall allow for both proprietary attestation signatures, as well as a standardized form of attestation signature. Either form of attestation signature may be applied to a set of TEEP claims, and both forms of attestation shall be considered conformant with TEEP. However, it should be recognized that not all TAMs or SPs may be able to process all proprietary forms of attestations. All TAMs and SPs MUST be able to process the TEEP standard attestation format and attached signature.

The attestation formats and mechanisms described and mandated by TEEP shall convey a particular set of cryptographic properties based on minimal assumptions. The cryptographic properties are conveyed by the attestation; however the assumptions are not conveyed within the attestation itself.

The assumptions which may apply to an attestation have to do with 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. Some of the assumptions that might apply to an attestations include (this may not be a comprehensive list):

- Assumptions regarding the security measures a manufacturer takes when provisioning keys into devices/TEEs;

- Assumptions regarding what hardware and software components have access to the Attestation keys of the TEE;

- Assumptions related to the source or local verification of claims within an attestation prior to a TEE signing a set of claims;

- Assumptions regarding the level of protection afforded to attestation keys against exfiltration, modification, and side channel attacks;

- Assumptions regarding the limitations of use applied to TEE Attestation keys;

- Assumptions regarding the processes in place to discover or detect TEE breeches; and

- Assumptions regarding 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. The choice and policy decisions are left up to the particular TAM/SP.

Some TAMs or SPs may require additional claims in order to properly authorize a device or TEE. These additional claims may help clear up any assumptions for which the TAM/SP wants to alleviate. The specific format for these additional claims are outside the scope of this specification, but the OTrP protocol SHALL allow these additional claims to be included in the attestation messages.

The following sub-sections define the cryptographic properties conveyed by the TEEP attestation, the basic set of TEEP claims required in a TEEP attestation, the TEEP attestation flow between the TAM the device TEE, and some implementation examples of how an attestation key may be realized in a real TEEP device.

7.1. Attestation Cryptographic Properties

The attestation constructed by TEEP must convey certain cryptographic properties from the attestor to the verifier; in the case of TEEP, the attestation must convey properties from the device to the TAM and/or SP. The properties required by TEEP include:

- Non-repudiation, Unique Proof of Source - the cryptographic digital signature across the attestation, and optionally along with information in the attestion 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.

Standard public key algorithms such as RSA and ECDSA digital signatures convey these properties. Group public key algorithms such as EPID can also convey these properties, if the attestation includes a unique device identifier and an identifier for the TEE. Other cryptographic operations used in other attestation schemes may also convey these properties.

The TEEP standard attestation format SHALL use one of the following digital signature formats:

- 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"

All TAMs and SPs MUST be able to accept attestations using these algorithms, contingent on their acceptance of the assumptions implied by the attestations.

7.2. TEEP Attestation Structure

For a TEEP attestation to be useful, it must contain an information set allowing the TAM and/or SP to assess the attestation and make a related security policy decision. The structure of the TEEP attestation is shown in the diagram below.

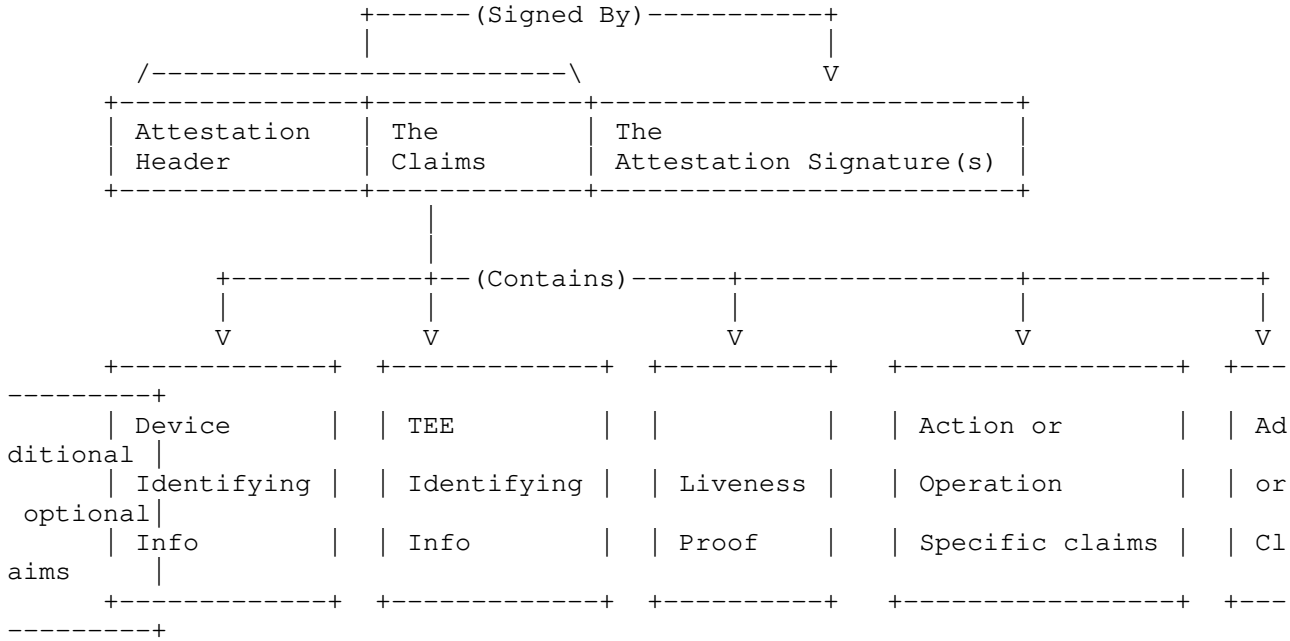

Figure 7: Structure of TEEP Attestation

The Attestation Header SHALL identify the "Attestation Type" and the "Attestation Signature Type" along with an "Attestation Format Version Number." The "Attestation Type" identifies the minimal set of claims that MUST be included in the attestation; this is an

identifier for a profile that defines the claims that should be included in the attestation as part of the "Action or Operation Specific Claims." The "Attestation Signature Type" identifies the type of attestation signature that is attached. The type of attestation signature SHALL be one of the standard signatures types identified by an IANA number, a proprietary signature type identified by an IANA number, or the generic "Proprietary Signature" with an accompanying proprietary identifier. Not all TAMs may be able to process proprietary signatures.

The claims in the attestation are set of mandatory and optional claims. The claims themselves SHALL be defined in an attestation claims dictionary. See the next section on TEEP Attestation Claims. Claims are grouped in profiles under an identifier (Attestation Type), however all attestations require a minimal set of claims which includes:

- Device Identifying Info: TEEP attestations must uniquely identify a device to the TAM and SP. This identifier allows the TAM/SP to provide services unique to the device, such as managing installed TAs, and providing subscriptions to services, and locating device-specific keying material to communicate wiht or authenticate the device. Additionally, device manufacturer information must be provided to provide better universal uniqueness qualities without requiring globally unique identifiers for all devices.

- TEE Identifying info: The type of TEE that generated this attestation must be identified. Standard TEE types are identified by an IANA number, but also must include version identification information such as the hardware, firmware, and software version of the TEE, as applicable by the TEE type. Structure to the version number is required.TEE manufacturer information for the TEE is required in order to disambiguate the same TEE type created by different manufacturers and resolve potential assumptions around manufacturer provisioning, keying and support for the TEE.

- Liveness Proof: a claim that includes liveness information SHALL be included which may be a large nonce or may be a timestamp and short nonce.

- Action Specific Claims: Certain attestation types shall include specific claims. For example an attestation from a specific TA shall include a measurement, version and signing public key for the TA.

- Additional Claims: (Optional - May be empty set) A TAM or SP may require specific additional claims in order to address potential assumptions, such as the requirement that a device’s REE performed

a secure boot, or that the device is not currenlty in a debug or non-productions state. A TAM may require a device to provide a device health attestation that may include some claims or measurements about the REE. These claims are TAM specific.

7.3. TEEP Attestation Claims

TEEP requires a set of attestation claims that provide sufficient evidence to the TAM and/or SP that the device and its TEE meet certain minimal requirements. Because attestation formats are not yet broadly standardized across the industry, standardization work is currently ongoing, and it is expected that extensions to the attestation claims will be required as new TEEs and devices are created, the set of attestation claims required by TEEP SHALL be defined in an IANA registry. That registry SHALL be defined in the OTrP protocol with sufficient elements to address basic TEEP claims, expected new standard claims (for example from https://www.ietf.org/id/draft-mandyam-eat-01.txt), and proprietary claim sets.

7.4. TEEP Attestation Flow

Attesations are required in TEEP under the following flows:

- When a TEE responds with device state information (dsi) to the TAM or SP, including a "GetDeviceState" response, "InstallTA" response, etc.

- When a new key pair is generated for a TA-to-TAM or TA-to-SP communication, the keypair must be covered by an attestation, including "CreateSecurityDomain" response, "UpdateSecurityDomain" response, etc.

7.5. Attestation Key Example

The attestation hierarchy and seed required for TAM protocol operation must be built into the device at manufacture. Additional TEEs can be added post-manufacture using the scheme proposed, but it is outside of the current scope of this document to detail that.

It should be noted that the attestation scheme described is based on signatures. The only decryption that may take place is through the use of a bootloader key.

A boot module generated attestation can be optional where the starting point of device attestation can be at TEE certificates. A TAM can define its policies on what kinds of TEE it trusts if TFW attestation is not included during the TEE attestation.

7.5.1. Attestation Hierarchy Establishment: Manufacture

During manufacture the following steps are required:

1. A device-specific TFW key pair and certificate are burnt into the device. This key pair will be used for signing operations performed by the boot module.

2. TEE images are loaded and include a TEE instance-specific key pair and certificate. The key pair and certificate are included in the image and covered by the code signing hash.

3. The process for TEE images is repeated for any subordinate TEEs, which are additional TEEs after the root TEE that some devices have.

7.5.2. Attestation Hierarchy Establishment: Device Boot

During device boot the following steps are required:

1. The boot module releases the TFW private key by decrypting it with the bootloader key.

2. The boot module verifies the code-signing signature of the active TEE and places its TEE public key into a signing buffer, along with its identifier for later access. For a TEE non-compliant to this architecture, the boot module leaves the TEE public key field blank.

3. The boot module signs the signing buffer with the TFW private key.

4. Each active TEE performs the same operation as the boot module, building up their own signed buffer containing subordinate TEE information.

7.5.3. Attestation Hierarchy Establishment: TAM

Before a TAM can begin operation in the marketplace, it must obtain a TAM certificate from a CA that is registered in the trust store of devices. In this way, the TEE can check the intermediate and root CA and verify that it trusts this TAM to perform operations on the TEE.

8. Algorithm and Attestation Agility

RFC 7696 [RFC7696] outlines the requirements to migrate from one mandatory-to-implement algorithm suite to another over time. This feature is also known as crypto agility. Protocol evolution is

greatly simplified when crypto agility is already considered during the design of the protocol. In the case of the Open Trust Protocol (OTrP) the diverse range of use cases, from trusted app updates for smart phones and tablets to updates of code on higher-end IoT devices, creates the need for different mandatory-to-implement algorithms already from the start.

Crypto agility in the OTrP concerns the use of symmetric as well as asymmetric algorithms. Symmetric algorithms are used for encryption of content whereas the asymmetric algorithms are mostly used for signing messages.

In addition to the use of cryptographic algorithms in OTrP there is also the need to make use of different attestation technologies. A Device must provide techniques to inform a TAM about the attestation technology it supports. For many deployment cases it is more likely for the TAM to support one or more attestation techniques whereas the Device may only support one.

9. Security Considerations

9.1. TA Trust Check at TEE

A TA binary is signed by a TA signer certificate. This TA signing certificate/private key belongs to the SP, and may be self-signed (i.e., it need not participate in a trust hierarchy). It is the responsibility of the TAM to only allow verified TAs from trusted SPs into the system. Delivery of that TA to the TEE is then the responsibility of the TEE, using the security mechanisms provided by the protocol.

We allow a way for an (untrusted) application to check the trustworthiness of a TA. A TEEP Broker has a function to allow an application to query the information about a TA.

An application in the Rich O/S may perform verification of the TA by verifying the signature of the TA. The GetTAInformation function is available to return the TEE supplied TA signer and TAM signer information to the application. An application can do additional trust checks on the certificate returned for this TA. It might trust the TAM, or require additional SP signer trust chaining.

9.2. One TA Multiple SP Case

A TA for multiple SPs must have a different identifier per SP. They should appear as different TAs when they are installed in the same device.

9.3. Broker Trust Model

A TEEP Broker could be malware in the vulnerable REE. A Client Application will connect its TAM provider for required TA installation. It gets command messages from the TAM, and passes the message to the Broker.

The architecture enables the TAM to communicate with the device's TEE to manage TAs. All TAM messages are signed and sensitive data is encrypted such that the TEEP Broker cannot modify or capture sensitive data.

9.4. Data Protection at TAM and TEE

The TEE implementation provides protection of data on the device. It is the responsibility of the TAM to protect data on its servers.

9.5. Compromised CA

A root CA for TAM certificates might get compromised. Some TEE trust anchor update mechanism is expected from device OEMs. A compromised intermediate CA is covered by OCSP stapling and OCSP validation check in the protocol. A TEE should validate certificate revocation about a TAM certificate chain.

If the root CA of some TEE device certificates is compromised, these devices might be rejected by a TAM, which is a decision of the TAM implementation and policy choice. Any intermediate CA for TEE device certificates SHOULD be validated by TAM with a Certificate Revocation List (CRL) or Online Certificate Status Protocol (OCSP) method.

9.6. Compromised TAM

The TEE SHOULD use validation of the supplied TAM certificates and OCSP stapled data to validate that the TAM is trustworthy.

Since PKI is used, the integrity of the clock within the TEE determines the ability of the TEE to reject an expired TAM certificate, or revoked TAM certificate. Since OCSP stapling includes signature generation time, certificate validity dates are compared to the current time.

9.7. Certificate Renewal

TFW and TEE device certificates are expected to be long lived, longer than the lifetime of a device. A TAM certificate usually has a moderate lifetime of 2 to 5 years. A TAM should get renewed or rekeyed certificates. The root CA certificates for a TAM, which are

embedded into the Trust Anchor store in a device, should have long lifetimes that don't require device Trust Anchor update. On the other hand, it is imperative that OEMs or device providers plan for support of Trust Anchor update in their shipped devices.

10. IANA Considerations

This document does not require actions by IANA.

11. Acknowledgements

The authors thank Dave Thaler for his very thorough review and many important suggestions. Most content of this document is split from a previously combined OTrP protocol document [I-D.ietf-teep-opentrustprotocol]. We thank the former co-authors Nick Cook and Minho Yoo for the initial document content, and contributors Brian Witten, Tyler Kim, and Alin Mutu.

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

12.2. Informative References

[GPTEE] Global Platform, "GlobalPlatform Device Technology: TEE System Architecture, v1.1", Global Platform GPD_SPE_009, January 2017, <https://globalplatform.org/specs-library/tee-system-architecture-v1-1/>.

[I-D.ietf-teep-opentrustprotocol] Pei, M., Atyeo, A., Cook, N., Yoo, M., and H. Tschofenig, "The Open Trust Protocol (OTrP)", draft-ietf-teep-opentrustprotocol-03 (work in progress), May 2019.

[RFC6024] Reddy, R. and C. Wallace, "Trust Anchor Management Requirements", RFC 6024, DOI 10.17487/RFC6024, October 2010, <https://www.rfc-editor.org/info/rfc6024>.

[RFC7696] Housley, R., "Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms", BCP 201, RFC 7696, DOI 10.17487/RFC7696, November 2015, <https://www.rfc-editor.org/info/rfc7696>.

Appendix A. History

RFC EDITOR: PLEASE REMOVE THIS SECTION

IETF Drafts

draft-00: - Initial working group document

Authors' Addresses

Mingliang Pei
Symantec

EMail: mingliang_pei@symantec.com

Hannes Tschofenig
Arm Limited

EMail: hannes.tschofenig@arm.com

David Wheeler
Intel

EMail: david.m.wheeler@intel.com

Andrew Atyeo
Intercede

EMail: andrew.atyeo@intercede.com

Liu Dapeng
Alibaba Group

EMail: maxpassion@gmail.com