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 April 26, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.
Table of Contents

1. Introduction ........................................... 3
2. Terminology ........................................... 5
3. Scope and Assumptions .................................... 7
4. Use Cases ............................................. 8
   4.1. Payment .......................................... 8
   4.2. Authentication .................................... 8
   4.3. Internet of Things ................................ 9
   4.4. Confidential Cloud Computing ....................... 9
5. Architecture ........................................... 9
   5.1. System Components ................................ 9
   5.2. Different Renditions of TEEP Architecture ........ 12
   5.3. Entity Relations .................................. 12
   5.4. Trust Anchors in TEE ............................. 15
   5.5. Trust Anchors in TAM ............................. 15
   5.6. Keys and Certificate Types ....................... 15
   5.7. Scalability ...................................... 18
   5.8. Message Security .................................. 18
   5.9. Security Domain Hierarchy and Ownership .......... 18
   5.10. SD Owner Identification and TAM Certificate Requirements 19
   5.11. Service Provider Container ....................... 20
6. TEEP Broker ............................................ 21
   6.1. Role of the Agent ................................ 22
   6.2. Agent Implementation Consideration ................ 22
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 recinded.

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.

- 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 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 owner 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. The trust anchor is normally stored in a location that resists unauthorized modification, insertion, or replacement. 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.

- 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
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. Scope and 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. This key pair is referred to as the ‘root of trust’ for remote attestation of the associated TEE in a device by an TAM.

New note: SD is for managing keys for TAs
A Security Domain (SD) concept is used as the security boundary inside a TEE for trusted applications. Each SD is typically associated with one TA provider as the owner, which is a logical space that contains an SP’s TAs. One TA provider may request to have multiple SDs in a TEE. One SD may contain multiple TAs. Each Security Domain requires the management operations of TAs in the form of installation, update and deletion.

Each TA binary and configuration data can be from either of two sources:

1. A TAM supplies the signed and encrypted TA binary and any required configuration data
2. A Client Application supplies the TA binary

The architecture covers the first case where the TA binary and configuration data are delivered from a TAM. The second case calls for an extension when a TAM is absent.

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.
Service Providers and Device Administrators 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 and SD’s on behalf of Service Providers and Device Administrators. This includes creation and deletion of TA’s and SD’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 and SD’s through an interaction with a Device’s TEEP Broker. As shown in Figure 1, 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 publically 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 public 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 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. An Agent 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 an Agent to the device manufacturer to be bundled in devices. Such a TEE must also include an Agent counterpart, namely, a processing module inside the TEE, to parse TAM messages sent through the Agent. An Agent is generally acting as a dummy relaying box with just the TEE interacting capability; it doesn’t need and shouldn’t parse protocol messages.
- 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

5.3. 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.

PKI   CA -- CA

Device   --- Agent / Client App ---

SW   --- TEE		TAM------

FW

Figure 2: Entities
Figure 3 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.
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 device SD and 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, an agent 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 agent, which is a software component that bridges communication between a TAM and a TEE. The agent does not need to know the actual content of messages except for the TEE routing information.

5.4. Trust Anchors in TEE

Each TEE comes with a trust store that contains a whitelist of root CA certificates 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 store 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 store live update or out-of-band update to devices.

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.

5.5. 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 decides what devices it will trust the TEE in.

5.6. Keys and Certificate Types

This architecture leverages the following credentials, which allow delivering end-to-end security without relying on any transport security.
<table>
<thead>
<tr>
<th>Key Entity Name</th>
<th>Location</th>
<th>Issuer</th>
<th>Checked Against</th>
<th>Cardinality</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. TFW key pair and certificate</td>
<td>Device secure storage</td>
<td>FW CA</td>
<td>A whitelist of FW root CA trusted by TAMs</td>
<td>1 per device</td>
</tr>
<tr>
<td>2. TEE key pair and certificate</td>
<td>Device TEE</td>
<td>TEE CA under a root CA</td>
<td>A whitelist of TEE root CA trusted by TAMs</td>
<td>1 per device</td>
</tr>
<tr>
<td>3. TAM key pair and certificate</td>
<td>TAM provider</td>
<td>TAM CA under a root CA</td>
<td>A whitelist of TAM root CA embedded in TEE</td>
<td>1 or multiple</td>
</tr>
<tr>
<td>4. SP key pair and certificate</td>
<td>SP</td>
<td>SP signer CA</td>
<td>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.</td>
<td>1 or multiple</td>
</tr>
</tbody>
</table>

Figure 5: 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.
   o 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.
   o Location: Device TEE
   o Supported Key Type: RSA and ECC
   o Issuer: A CA that chains to a TEE root CA
   o Checked Against: A whitelist of TEE root CAs trusted by TAMs
   o Cardinality: One per device

3. TAM key pair and certificate: A TAM provider acquires a certificate from a CA that a TEE trusts.
   o Location: TAM provider
   o Supported Key Type: RSA and ECC.
   o Supported Key Size: RSA 2048-bit, ECC P-256 and P-384. Other sizes should be anticipated in future.
   o Issuer: TAM CA that chains to a root CA
   o Checked Against: A whitelist of TAM root CAs embedded in a TEE
   o 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.
   o Location: SP
   o Supported Key Type: RSA and ECC
   o Supported Key Size: RSA 2048-bit, ECC P-256 and P-384. Other sizes should be anticipated in future.
   o Issuer: An SP signer CA that chains to a root CA
   o 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.
5.7. 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.

5.8. Message Security

Messages created by a TAM are used to deliver device SD and 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.

5.9. Security Domain Hierarchy and Ownership

The primary job of a TAM is to help an SP to manage its trusted applications. A TA is typically installed in an SD. An SD is commonly created for an SP.

When an SP delegates its SD and TA management to a TAM, an SD is created on behalf of a TAM in a TEE and the owner of the SD is assigned to the TAM. An SD may be associated with an SP but the TAM has full privilege to manage the SD for the SP.

Each SD for an SP is associated with only one TAM. When an SP changes TAM, a new SP SD must be created to associate with the new TAM. The TEE will maintain a registry of TAM ID and SP SD ID mapping.

From an SD ownership perspective, the SD tree is flat and there is only one level. An SD is associated with its owner. It is up to the
TEE implementation how it maintains SD binding information for a TAM and different SPs under the same TAM.

It is an important decision in this architecture that a TEE doesn’t need to know whether a TAM is authorized to manage the SD for an SP. This authorization is implicitly triggered by an SP Client Application, which instructs what TAM it wants to use. An SD is always associated with a TAM in addition to its SP ID. A rogue TAM isn’t able to do anything on an unauthorized SP’s SD managed by another TAM.

Since a TAM may support multiple SPs, sharing the same SD name for different SPs creates a dependency in deleting an SD. An SD can be deleted only after all TAs associated with the SD are deleted. An SP cannot delete a Security Domain on its own with a TAM if a TAM decides to introduce such sharing. There are cases where multiple virtual SPs belong to the same organization, and a TAM chooses to use the same SD name for those SPs. This is totally up to the TAM implementation and out of scope of this specification.

5.10. SD Owner Identification and TAM Certificate Requirements

There is a need of cryptographically binding proof about the owner of an SD in a device. When an SD is created on behalf of a TAM, a future request from the TAM must present itself as a way that the TEE can verify it is the true owner. The certificate itself cannot reliably used as the owner because TAM may change its certificate.

** need to handle the normal key roll-over case, as well as the less frequent key compromise case

To this end, each TAM will be associated with a trusted identifier defined as an attribute in the TAM certificate. This field is kept the same when the TAM renew its certificates. A TAM CA is responsible to vet the requested TAM attribute value.

This identifier value must not collide among different TAM providers, and one TAM shouldn’t be able to claim the identifier used by another TAM provider.

The certificate extension name to carry the identifier can initially use SubjectAltName:registeredID. A dedicated new extension name may be registered later.

One common choice of the identifier value is the TAM’s service URL. A CA can verify the domain ownership of the URL with the TAM in the certificate enrollment process.
A TEE can assign this certificate attribute value as the TAM owner ID for the SDs that are created for the TAM.

An alternative way to represent an SD ownership by a TAM is to have a unique secret key upon SD creation such that only the creator TAM is able to produce a proof-of-possession (PoP) data with the secret.

5.11. Service Provider Container

A sample Security Domain hierarchy for the TEE is shown in Figure 6.

```
| TEE |

--------

| SP1 SD1 |
--------

| SP1 SD2 |
--------

| SP2 SD1 |
--------
```

Figure 6: Security Domain Hierarchy

The architecture separates SDs and TAs such that a TAM can only manage or retrieve data for SDs and TAs that it previously created for the SPs it represents.

5.12. A Sample Device Setup Flow

Step 1: Prepare Images for Devices

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

- 1. [CA] Deliver root CA Whitelist

- 1. [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)

1. [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.

1. [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

1. [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 agent comes to meet this need. An agent 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 TAM SDK or Client Applications to query and trigger TA installation that the application needs to use.

This interface for Client Applications may be commonly an OS service call for an REE OS. A Client Application interacts with a TAM, and turns around to pass messages received from TAM to agent.

In all cases, a Client Application needs to be able to identify an agent that it can use.

6.1. Role of the Agent

An agent abstracts the message exchanges with the TEE in a device. The input data is originated from a TAM to which a Client Application connects. A Client Application may also directly call an Agent for some TA query functions.

The agent may internally process a message from a TAM. At least, it needs to know where to route a message, e.g., TEE instance. It does not need to process or verify message content.

The agent returns TEE / TFW generated response messages to the caller. The agent is not expected to handle any network connection with an application or TAM.

The agent only needs to return an agent 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. Agent Implementation Consideration

A Provider should consider methods of distribution, scope and concurrency on devices and runtime options when implementing an agent. 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. Agent Distribution

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

It is important to ensure a legitimate agent is installed and used. If an agent is compromised it may drop messages and thereby introduce a denial of service.
6.2.2.  Number of Agents

We anticipate only one shared agent instance in a device. The device’s TEE vendor will most probably supply one agent.

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

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

TAM providers are generally expected to provide an SDK for SP applications to interact with an agent for the TAM and TEE interaction.

7.  Attestation

7.1.  Attestation Hierarchy

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.

7.1.1.  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.

7.1.2. 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.

7.1.3. 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 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. An agent 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. A TA will be installed in a different SD for each respective SP.

9.3. Agent Trust Model

An agent 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 agent.

The architecture enables the TAM to communicate with the device’s TEE to manage SDs and TAs. All TAM messages are signed and sensitive data is encrypted such that the agent 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


12.2. Informative References


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