TEEP
Internet-Draft
Intended status: Informational
Expires: January 3, 2019

M. Pei
Symantec
H. Tschofenig
Arm Ltd.
A. Atyeo
Intercede
D. Liu
Alibaba Group
July 2, 2018

Trusted Execution Environment Provisioning (TEEP) Architecture
draft-ietf-teep-architecture-00.txt

Abstract

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

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

Commented [DT1]: Suggested terminology fix to apply more generically to various TEE's including SGX. The term "application" alone refers to the combination of the client application + TAs that it uses.

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 http://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 3, 2019.

Copyright Notice

Table of Contents

1. Introduction

The Trusted Execution Environment (TEE) concept has been designed to separate a regular operating system, also referred as a Rich Execution Environment (REE), from security-~~-~~sensitive application components. A TEE provides hardware-enforcement ~~so~~ that any data inside the TEE cannot be read by code outside of the TEE. Compromising an REE and normal applications in the REE do not affect code running inside the TEE, which ~~is~~ are called ~~a~~ Trusted Applications (TAs)~~, running inside the TEE~~.

In a~~n~~ TEE ecosystem, a Trusted Application Manager (TAM) is commonly used to manage keys and TAs that run in a device. Different device vendors may use different TEE implementations. Different application providers or device administrators may choose to use different TAM providers.

To simplify the life of developers an interoperable protocol for managing TAs running in different TEEs of various devices is needed.

The protocol addresses the following problems.

1. A Device Administrator (DA) or Service Provider (SP) of the device users needs to determine security-relevant information of a device before provisioning the TA to the device with a TEE. Examples include the verification of the device 'root of trust' and the type of TEE included in a device.

2. A TEE in a device needs to determine whether a Device Administrator (DA) or a Service Provider (SP) that wants to manage a~~n~~ TA in the device is authorized to manage applications in the TEE.

3. Attestation must be able to ensure a TEE is genuine.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Client Application: An application running i~~o~~n a ~~rich OS~~Rich Execution Environment, such as an Android, Windows, or iOS application.

Commented [DT2]: (Plural since "normal applications" are plural)

Commented [DT3]: Don't use term before it is defined

Commented [DT4]: Can't parse this. Delete "users"?

Commented [DT5]: Attestation is not an actor. This should probably be combined into the previous two items.

Commented [DT6]: Undefined term. I'd suggest rephrasing to say "in a Rich Execution Environment" which is defined

Device: A physical piece of hardware that hosts a TEE along with a ~~rich OS~~Rich Execution Environment.

Agent: An application running in ~~the rich OS~~a Rich Execution Environment allowing the message protocol exchange between a TAM and a TEE in a device. A TEE is responsible ~~to~~ for processing relayed messages and for returning an appropriate response.

Rich Execution Environment (REE) An environment that is provided and governed by a typical OS (Linux, Windows, Android, iOS, etc.), 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.

Secure Boot Module (SBM): ~~A f~~Firmware in a device that delivers secure boot functionality. It is generally signed and can be verified whether it can be trusted.

Service Provider (SP): An entity that wishes to supply Trusted Applications to remote devices. A Service Provider requires the help of a TAM in order to provision the Trusted Applications to the devices.

Trust Anchor: A root certificate that can be used to validate its children certificates. It is usually embedded in a device or configured by a TAM for validating the trust of a remote entity's certificate.

Trusted Application (TA): An a~~A~~pplication 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

Commented [DT7]: So if the device administrator runs the TAM and manages the devices they own, and the device administrator wants to supply third-party TAs it has authorized, does that make the device administrator a Service Provider? If yes, then we should define it that way and section 1 would not need to say "Device Administrator or Service Provider" since a device administrator would just be one type of service provider. If no, then this definition is incorrect and needs work.

Commented [DT8]: Per email thread, the normal term in other standards bodies TEEP interfaces with is "Root of Trust". And since 'root of trust' is used in section 3, we should just use the same term throughout, whichever one we use, and mention here that it is also known as the other one.

Commented [DT9]: Probably not the right definition since "Application" is not a defined term. Maybe "An application component"?

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 of the TEE.

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

Trusted Firmware (TFW): A signed SBM ~~firmware~~ that can be verified and is trusted by a TEE in a device.

~~This document uses the following abbreviations:~~

~~CA Certificate Authority~~

~~REE Rich Execution Environment~~

~~SD Security Domain~~

~~SP Service Provider~~

~~SBM Secure Boot Module~~

~~TA Trusted Application~~

~~TEE Trusted Execution Environment~~

~~TFW Trusted Firmware~~

~~TAM Trusted Application Manager~~

Commented [DT10]: TA?

Commented [DT11]: My reading of others comments so far is that there is not consensus that firmware necessarily has to have SBM in order to use the TEEP architecture. So I'm not sure yet what the right definition is here.

Commented [DT12]: Recommended removing since redundant with the definitions immediately above

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.

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

1. A TAM supplies the signed and encrypted TA binary

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.

~~Messages exchange with a TAM require some transport and HTTPS is one commonly used transport.~~

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 ~~running inside TEE in the device~~ to provide strong identification and proof of transaction.

For a mobile payment application, some biometric identification information could also be stored in ~~the~~ 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.

Commented [DT13]: Can we use a specific RTx term here that matches a TCG/NIST/GP term?

Commented [DT14]: If the device owner is the SP, and the device owner has TAs that need to be isolated from each other, is there one SD or one SD per TA? For now I'm assuming the latter because of the next sentence.

Commented [DT15]: … which means that they are not isolated from each other and can share memory and other resources? If so, say so explicitly.

Commented [DT16]: What about the configuration data? Can the binary and config data come via different sources?

Commented [DT17]: I think it needs to cover both.

Commented [DT18]: I don't believe this is true. The use case discussed is where the TAM provides the authorization, it's just the raw binary bits that come from somewhere else (e.g., a client app). See the original BOF poster and slides.

Commented [DT19]: In my opinion, this sentence isn't needed in the arch doc, just in the protocol doc.

Commented [DT20]: (removed since redundant, as it's part of the definition of Trusted Application)

4.2. Authentication

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

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 a 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 to the locking mechanism. Bike-sharing is another example that shares a similar usage scenario.

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.

TAM: A TAM is responsible for originating and coordinating lifecycle management activity on a particular TEE on behalf of a Service Provider or a Device Administrator. For example, a payment application provider, which also provides payment service as a Service Provider using its payment TA, may choose to use a TAM that it runs or a third party TAM service to distribute and update its payment TA application in payment user devices. The payment SP isn't a device administrator of the user devices. A user who chooses to download the payment TA into its a devices acts as the device administrator, authorizing the TA installation via the downloading consent. The device manufacturer is typically

Commented [DT21]: This is often far more dangerous than simply stealing data.

Commented [DT22]: (The previous sentence is sufficient in my view, and "bike-sharing" would need an explanation to define for all readers, so isn't worth the text)

Commented [DT23]: So in this example, the TA installer component has an association with both the TAM and the device administrator, and only installs the TA once both approve? If so, say so explicitly.

responsible for embedding the TAM trust verification capability in its device TEE.

A TAM may be used by one SP or many SPs where a TAM may run as a Software-as-a-Service (SaaS). A TAM may provide Security Domain management and TA management in a device for the SD and TAs that an SP owns. In particular, a TAM typically offers over-the-air updates to keep an SP's TAs up-to-date and clean up when a version should be removed. A TEE administrator or device administrator may ~~decide~~ choose TAMs that it trusts to manage its devices.

Certificat~~eion~~ 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.

TEE: A TEE in a device is responsible for protecting applications from attack, enabling the application to perform secure operations.

REE: The REE in a device is responsible for enabling off-device communications to be established between a TEE and TAM. The architecture does not assume or require that the REE or Client Applications is secure.

Agent: 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 ~~the~~ Client Applications can use to pass messages to TEEs. An ~~Agent is this component to fill the role. In other words, an~~ Agent is thus an application in the REE or software library that can ~~simply~~ relay~~s~~ messages from a Client Application to a TEE in the device. A device usually comes with only one active TEE. A TEE ~~that supports~~ may provide such an Agent to the device manufacturer to be bundled in devices. Such a ~~compliant~~ 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.

Device Administrator: A device owner or administrator may want to manage what TAs allowed to run in its devices. A default list of allowed TA trust root CA certificates is included in a device by

Commented [DT24]: This only covers the non-DA use case. I think it should be its own paragraph (here or moved elsewhere I don't care which), along with another example where the DA uses the TAM.

Commented [DT25]: Does that mean that a device might have many TAM's, one per SP whose TAs can be used?

Commented [DT26]: Confusing. "When a TA should be removed"?

Commented [DT27]: Aren't these two the same thing? If they're different, when would they be different?

Commented [DT28]: No, a CA is not credentials. A CA issues credentials.

Commented [DT29]: Ambiguous. Does this mean that the REE needs to contain code for communicating with TAMs? Or can this be left to a Client Application to implement?

Commented [DT30]: This seems too limiting to me. If an REE includes such support, then in one example implementation, a client application would never need to communicate with a TAM directly, it would just express a dependency on a TA (which may or may not be included with the TA), and the agent in the rich OS might do it for the client app.

Commented [DT31]: This does not seem important to the definition of Agent (which admittedly, I am already confused by).

Commented [DT32]: No. People and organizations provide an agent. The TEE code does not.

Commented [DT33]: A specific defined term would be helpful here. I'd suggest "OTrP TA" as a strawman. I'd also be ok with other terms, but please defined one.

the device's manufacturer, which may be governed by the device carriers sometimes. ~~There may be needs to expose overriding capability for a~~A device owner ~~to~~ can decide the list of allowed TAMs by updating the list of trusted CA certificates.

Secure Boot: Secure boot ~~must~~ enables authenticity checking of TEEs by the TAM. Note that some TEE implementations do not require secure boot functionality.

5.2. Entity Relations

This architecture leverages asymmetric cryptography to authenticate a device to~~wards~~ a TAM. Additionally, a TEE in a device authenticates a TAM ~~provider~~ and TA signer. The provisioning of trust anchors to a device may different from one use case to the other. ~~The~~ 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.

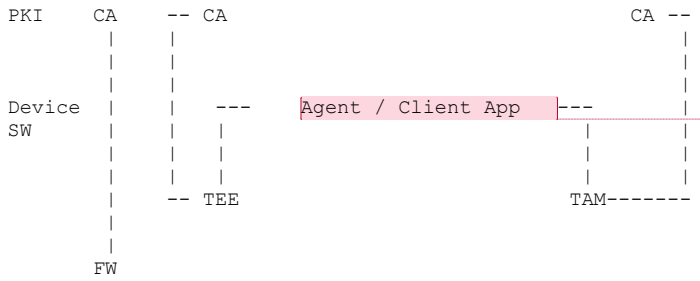

Figure 1: Entities

Commented [DT34]: Undefined term. If I have a Hue-like lightbulb with a TEE, and I carry the bulb upstairs to install it, am I the device carrier?

Commented [DT35]: No. It can decide the list of allowed TAMs. The TAM decides the list of allowed TAs.

Commented [DT36]: In this doc, try to avoid language that sounds RFC 2119-like if possible.

Commented [DT37]: "TAM provider" is not a defined term. TAM and SP are defined terms.

Commented [DT38]: Same problem here. Undefined term.

Commented [DT39]: Be careful… in an IoT device, there may be no notion of "downloading" (which term usually implies a human logged onto the device and initiating operations from it). So this sentence is a big confusing as worded.

Commented [DT40]: This figure looks wrong (or at least too limiting) to me, since it implies that the communication channel to the TAM must be in the same entity (i.e, the same Client App) as actually depends on the TA. I think that should be allowed, but certainly not required, as it just results in more bloated client apps. The term "Agent" would be fine, which may or may not be in the Client App.

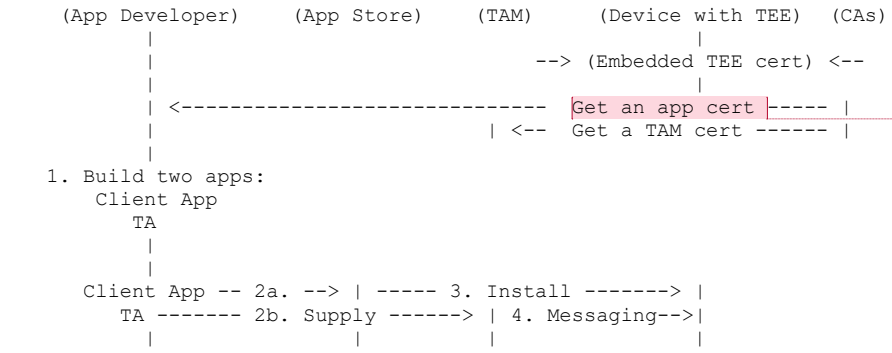

Figure 2: Developer Experience

Figure 2 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 ~~calling~~ 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 ~~will~~ shows a system diagram about the entity relationships between CAs, TAMs, SPs and devices.

Commented [DT41]: I think there would typically be a request from the app developer to the CA to request an app cert, and then a line back to supply it. It would be unusual for a CA to reach out to an app developer and just give them one without a request.

Commented [DT42]: … or metadata (like a manifest) about it?

Commented [DT43]: The diagram is more correct than this sentence, in my view. See my earlier comments.

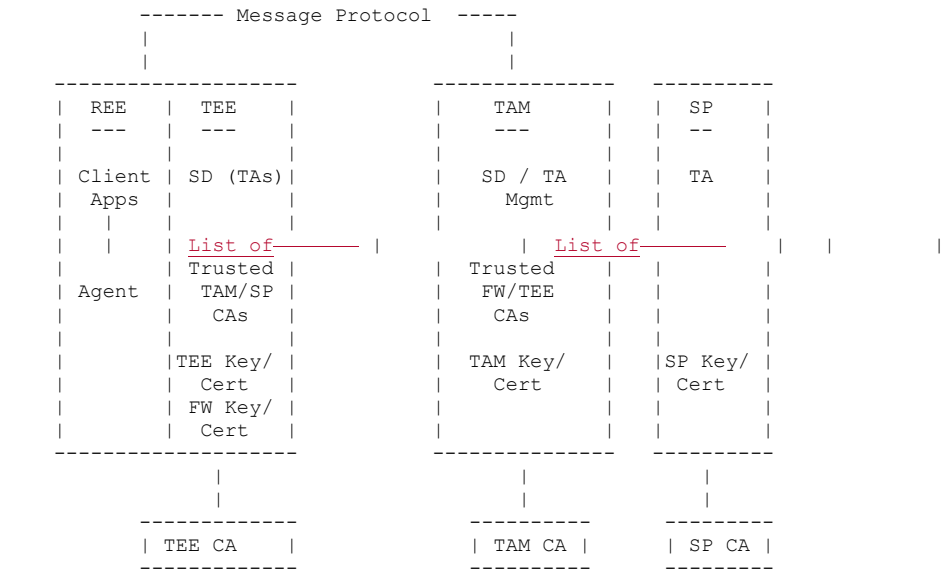

Figure 3: Keys

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 ~~a~~ 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~~s~~ 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~~internet~~.

Commented [DT44]: This might be ok, but offhand it sounds backwards to me. That is, I would expect the TAM to be passive and respond to requests from the TEE that wants to know whether it can install something requested by a client app. My main point though is that the arch doc should be more agnostic here and leave it to the protocol spec. E.g., just say “message” instead of “request” or “response”.

Commented [DT45]: Undefined term. I think devices are electrically powered, not TEE-powered. And the term “device” refers to the combination of REE+TEE and the REE does have it. I think you mean “in most TEE’s”.

Commented [DT46]: … which may or may not be the client application that depends on a TA that it wants installed.

It is imperative to have an interoperable protocol to communicate with different TAMs and different TEEs in different devices ~~that a Client Application needs to run and access a TA inside a TEE~~. 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.

Commented [DT47]: Not sure I understand this sentence. The agent should just be responsible for initiating a transport connection. I'm not sure what "interoperable protocol" is being referred to as the role of the agent. Are you referring to the transport protocol? OTrP? Something else?

5.3. 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.

Commented [DT48]: Can't parse grammar. Not sure what was meant.

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

Commented [DT49]: Undefined term. See suggested rewording.

5.4. 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.

5.5. 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 white list of FW root CA trusted by TAMs	1 per device
2. TEE key pair and certificate	Device TEE	TEE CA under a root CA	A white list of TEE root CA trusted by TAMs	1 per device
3. TAM key pair and certificate	TAM provider	TAM CA under a root CA	A white list of TAM root CA embedded in TEE	1 or multiple can be used by a TAM
4. SP key pair and certificate	SP	SP signer CA	A SP uses a TAM. TA is signed by a SP signer. TEE delegates trust of TA to TAM. SP signer is associated with a SD as the owner.	1 or multiple can be used by a TAM

Table 1: Key and Certificate Types

1. TFW key pair and certificate: A key pair and certificate for evidence of secure boot and trustworthy firmware in a device.

   Location: Device secure storage

   Supported Key Type: RSA and ECC

   Issuer: OEM CA

   Checked Against: A white list 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.

Commented [DT50]: Per TCG work, don't assume that the key pair is actually stored itself. They might be dynamically rederived from other things at boot time.

Commented [DT51]: Needs elaboration. I think you mean the SP signer is checked by the TAM. And the TEE only checks whether something is signed by a TAM it trusts. Correct?

This key pair is burned into the device ~~at~~ ~~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 white list 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 white list of TAM root CAs embedded in a TEE

   Cardinality: One or multiple can be used by a TAM

4. SP key pair and certificate: A~~a~~n 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: A~~a~~n 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 ~~from the TAM~~ is trustworthy.

   Cardinality: One or multiple can be used by an SP

Commented [DT52]: This seems too restrictive. If a manufacturer wants to have a way to renew certificates, why would we disallow that? Requiring long-lifetime (20-30 years) certs might be less secure.

Commented [DT53]: Don't assume that TA binaries come from a TAM. Just remove this phrase.

5.6. 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~~s~~ 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.

5.7. Message Security

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

Commented [DT54]: Some wording suggestions, but even with those I cannot understand this sentence. Grammar needs fixing.

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.

5.8. 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.

Commented [DT55]: (This is true if a device admin that uses a TAM to manage all TAs in the iot devices they own, can be called the SP. If not, this statement needs changing.)

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.

Commented [DT56]: Unclear what this phrase means

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.

Commented [DT57]: Ambiguous. Can you have a TAM that is distributed/replicated for high availability? Assuming so, changing which instance you're talking to should not result in any such SD change.

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 ~~protocol specification~~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 ~~this~~ the SD ~~is~~ 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.

5.9. 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.

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.

Commented [DT58]: I don't understand any of this text. Needs further elaboration or rewording.

Commented [DT59]: This is unclear/confusing. Why would an SD have to have a "name"? Can we keep this concept out of the arch doc? If not, then need to explain it more. Two paragraphs above here, it means "SD ID". Is that the same as "SD name"?

Commented [DT60]: I don't understand what "with a TAM" means here.

Commented [DT61]: Undefined term. What's a "virtual SP"?

Commented [DT62]: But the subject name can be used, no?

Commented [DT63]: Why would you need anything new?

5.10. Service Provider Container

A sample Security Domain hierarchy for the TEE is shown in Figure 4.

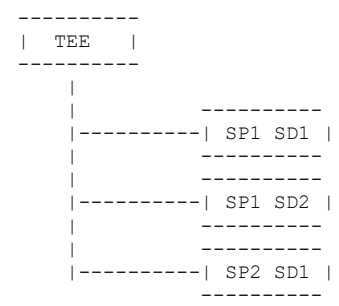

Figure 4: Security Domain Hierarchy

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.

Commented [DT64]: The relationship to a TAM is not evident in figure 4. How many TAMs are there in this example? One? One per SP? One per SD?

5.11. 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

Commented [DT65]: … to whom? "to device OEM"?

Commented [DT66]: … to device OEM?

Step 2: Inject Key Pairs and Images to Devices

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

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

Step 3: Set up attestation key pairs in devices

1. [OEM] Flash Secure Boot Public Key and eFuse Key (eFuse key is unique per device)

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 eFuse key

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

6. Agent

A TEE and TAs do not generally have the capability to communicate to the outside of the hosting device. For example, ~~the~~ Global ~~P~~Platform [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 ~~may~~ might carry this communication functionality but ~~it~~ 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~~d~~ 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 ~~Android~~OS service call for an ~~Android powered device~~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 ~~that~~ 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 ~~request~~ 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.

Commented [DT67]: If the agent is in the SDK, I don't know what it means for the agent to provide interfaces for the TAM SDK to communicate. Provide interfaces for itself? Seems like you shouldn't need to mention that. Maybe just say "for Client Applications"?

Commented [DT68]: No need to call out any single OS here.

Commented [DT69]: This sentence and the previous paragraph together imply that it's the same Client App that interacts with the TAM as wants a TA to be installed. That's certainly possible in some cases, but should not be the only model in the architecture.

As such, I think the term "Agent" is not well-defined enough. When an REE (whether in a OS or TAM SDK or whatever else) includes a client application that can talk to TAMs on behalf of other client apps that want to install TAs, I would call that component the Agent. Thus, the Agent would do the communication with the TAM, and the TEE interface as well. The client application that depends on a TA may or may not even be running at the time, e.g., if the app store installer invokes the Agent and passes it the client app manifest requirements and binaries, the client app itself may not be involved in the TAM exchange at all, except for the metadata pulled from its manifest.

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 introduc~~eing~~ 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 aent.

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.

Commented [DT70]: In my opinion, this statement is too restrictive as noted above. Maybe it's just a terminology point about what component is called "agent", and whether that term is misleading.

Commented [DT71]: Meaning what?

Commented [DT72]: Typo. Not sure what was meant

Commented [DT73]: Author? Distributor?

Commented [DT74]: I think this expectation is too restrictive, and rules out models that are more scalable in many cases. For example, if the OS provided an API independent of TAM provider, that's another model that allows a client app to scale to more TAM providers.

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 encryption that takes place may be the use of a so-called eFuse to release the SBM signing key and later during the protocol lifecycle management interchange with the TAM.

SBM attestation can be optional ~~in TEEP architecture~~ where the starting point of device attestation can be at TEE cert~~f~~ificates. A TAM can define its policies on what kinds of TEE it trusts if TFW attestation isn't 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, encrypted by eFuse. This key pair will be used for signing operations performed by the SBM.

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. Secure boot releases the TFW private key by decrypting it with eFuse.

2. The SBM 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 SBM leaves the TEE public key field blank.

3. The SBM signs the signing buffer with the TFW private key.

4. Each active TEE performs the same operation as the SBM, 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 ~~to support devices of a given TEE~~, it must obtain a TAM certificate from a CA that is registered in the trust store of devices ~~with that TEE~~. 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. Acknowledgements

The authors thank Dave Thaler for his very thorough review and many important suggestions. Most content of this document ~~are~~ 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.

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.

Commented [DT75]: Do not assume that all devices with the same type of TEE are equivalent. The TAM needs a cert that a device will trust. That's a statement independent of TEE. See suggested rewording.

Commented [DT76]: This is not the only way to do this, and should be more abstract than this in the arch doc.

Commented [DT77]: This is an implementation specific detail. Expecially "The GetTAInformation function" is implementation specific, not part of an architecture.

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 ~~Rich OS~~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.

Commented [DT78]: The next sentence is true, but I don't understand this one. Let's say each TA has an identifier that is common across all SPs. The system can simply use the tuple {TA ID, SD ID} to identify the instance in each SD. No separate identifier is needed. That's just an implementation detail. So I think this sentence should be removed.

Commented [DT79]: I can't parse/understand this phrase

Commented [DT80]: Either add [reference], or say it's up to the protocol spec to specify and remove mention of OCSP from this doc.

Commented [DT81]: Avoid RFC 2119 language in this doc if possible. (If not, we need RFC 2119 as a normative reference, but it would be better in my opinion if this doc were non-normative.)

Commented [DT82]: Same comments on this section

Commented [DT83]: Should elaborate on this. If the TEE does not have a hardware source of absolute time, it may have a secure relative clock but not a secure "current" time. To do that, you may need a secure time protocol implementation in the TEE, to get or compute absolute 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.

Commented [DT84]: FYI, some IOT devices have a lifetime of 20-30 years. Hence the desire to allow renewal of device certificates in some cases.

10. References

10.1. 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>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <https://www.rfc-editor.org/info/rfc7516>.

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <https://www.rfc-editor.org/info/rfc7517>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://www.rfc-editor.org/info/rfc7518>.

10.2. Informative References

[GPTEE] Global Platform, "Global Platform, GlobalPlatform Device Technology: TEE System Architecture, v1.0", 2013.

[GPTEECLAPI] Global Platform, "Global Platform, GlobalPlatform Device Technology: TEE Client API Specification, v1.0", 2013.

[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-00 (work in progress), May 2018.

Authors' Addresses

Mingliang Pei
Symantec
350 Ellis St
Mountain View, CA 94043
USA

Email: mingliang_pei@symantec.com

Hannes Tschofenig
Arm Ltd.
Absam, Tirol 6067
Austria

Email: Hannes.Tschofenig@arm.com

Andrew Atyeo
Intercede
St. Mary's Road, Lutterworth
Leicestershire, LE17 4PS
Great Britain

Email: andrew.atyeo@intercede.com

Dapeng
Alibaba Group
Wangjing East Garden 4th Area,Chaoyang District
Beijing 100102
China

Email: maxpassion@gmail.com