Trusted Execution Environment Provisioning (TEEP) WG

IETF 101, Tuesday, July 17, 2018

Chairs:

Dave Thaler

Nancy Cam-Winget

Note Well

This is a reminder of IETF policies in effect on various topics such as patents or code of conduct. It is only meant to point you in the right direction. Exceptions may apply. The IETF's patent policy and the definition of an IETF "contribution" and "participation" are set forth in BCP 79; please read it carefully.

As a reminder:

- By participating in the IETF, you agree to follow IETF processes and policies.
- If you are aware that any IETF contribution is covered by patents or patent applications that are owned or controlled by you or your sponsor, you must disclose that fact, or not participate in the discussion.
- As a participant in or attendee to any IETF activity you acknowledge that written, audio, video, and photographic records of meetings may be made public.
- Personal information that you provide to IETF will be handled in accordance with the IETF Privacy Statement.
- As a participant or attendee, you agree to work respectfully with other participants; please contact the ombudsteam (https://www.ietf.org/contact/ombudsteam/) if you have questions or concerns about this.

Definitive information is in the documents listed below and other IETF BCPs. For advice, please talk to WG chairs or ADs:

- BCP 9 (Internet Standards Process)
- BCP 25 (Working Group processes)
- BCP 25 (Anti-Harassment Procedures)
- BCP 54 (Code of Conduct)
- BCP 78 (Copyright)
- BCP 79 (Patents, Participation)
- https://www.ietf.org/privacy-policy/ (Privacy Policy)

Administrative Tasks

Bluesheets

We need volunteers to be:

- Two note takers
- One jabber scribe

Jabber: xmpp:teep@jabber.ietf.org?join

MeetEcho: https://www.meetecho.com/ietf101/teep

Etherpad: https://etherpad.tools.ietf.org/p/notes-ietf-101-teep

Agenda

1. Agenda bashing, Logistics -- Chairs (5 mins)
2. Review of charter milestones -- Chairs (5 mins)
3. Architecture (draft-ietf-teep-architecture-00) – Ming & Hannes (30 mins)
4. Open Trust Protocol (draft-ietf-teep-opentrustprotocol-01) – Ming (30 mins)
5. TEEP Hackathon report – Dave Thaler (20 mins)
6. SGX Overview and Impact on TEEP – David Wheeler (30 mins)
7. AOB

Milestones (are we on target?)

Date	Milestone
Mar 2018	**Adopt an Architecture document**
Mar 2018	**Adopt a solution document**
Dec 2018	Begin WGLC for Architecture document
Apr 2019	Progress Architecture document to the IESG for publication
Jul 2019	Begin WGLC for Solution document
Dec 2019	Progress Solution document to the IESG for publication

TEEP Architecture Draft

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

Mingliang Pei, Hannes Tschofenig, Andrew Atyeo, Dapeng Liu

IETF#102

Document Status

- Submitted WG draft on 07/02/2018
  - Received a good list of comments in mailing list
- It took largely the architecture extracted from the WG approved OTrP draft
- Addressed some comments from the mailing list on OTrP into this draft
- Started to address the feedback from the last WG

Current Document Structure

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

Changes in v0

- Terminology Unification
  - Secure Boot Module vs. Trusted Firmware
  - Service Provider vs. App Developer
- Added a diagram about “User Experience”
- Started to make Trusted Firmware verification optional
- Made transport support as a requirement
- Made assumption to support multiple TEEs
- Made TA binary distribution by a Client Application a goal

OPEN ISSUES

Trusted Firmware

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

Trusted App Distribution

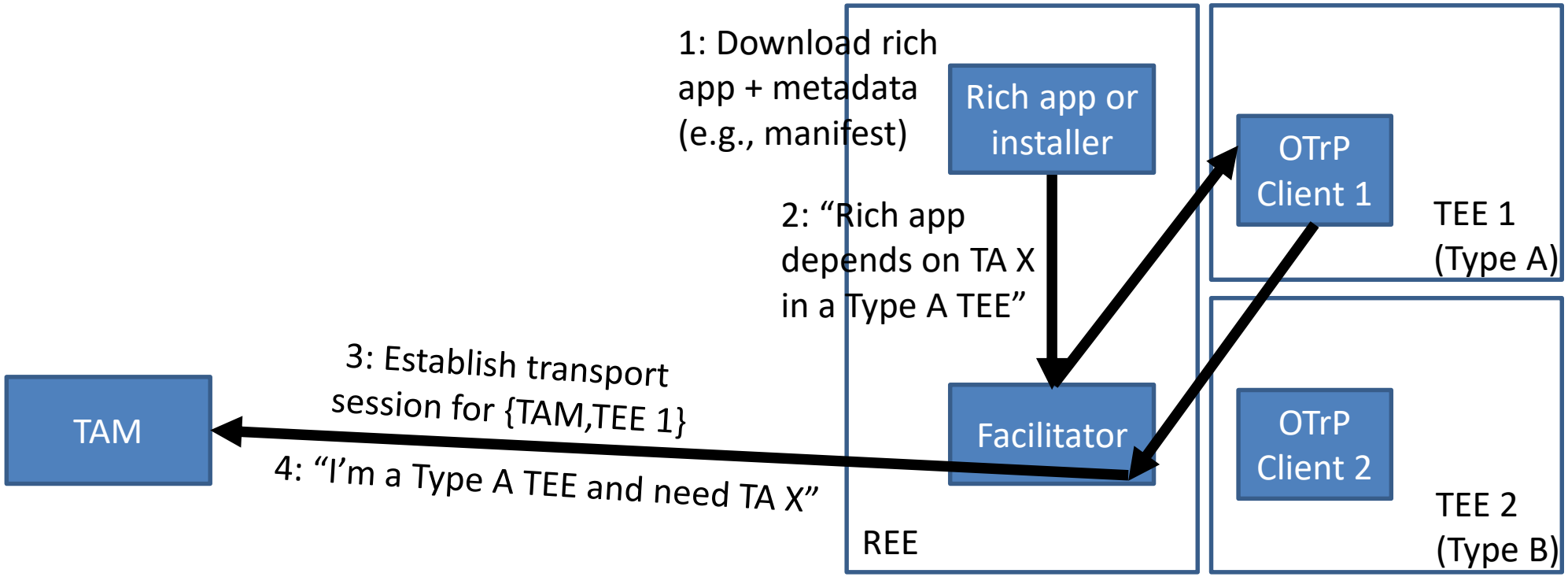
- Two modes:
  - TA binary bundled with the Client Application
  - TA distributed by TAM
- Challenges with first approach is
  - Passing device or TA instance specific data requires real-time interaction with a TAM. This functionality is in use today.
  - Client Application is not authorized to query TEE device state. Who is authorized to update a TA in the future? What would be the Security Domain?

Multiple TEEs vs. Single TEE

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

How are messages routed to the correct TEE?

Example below for heterogeneous TEE types in same device



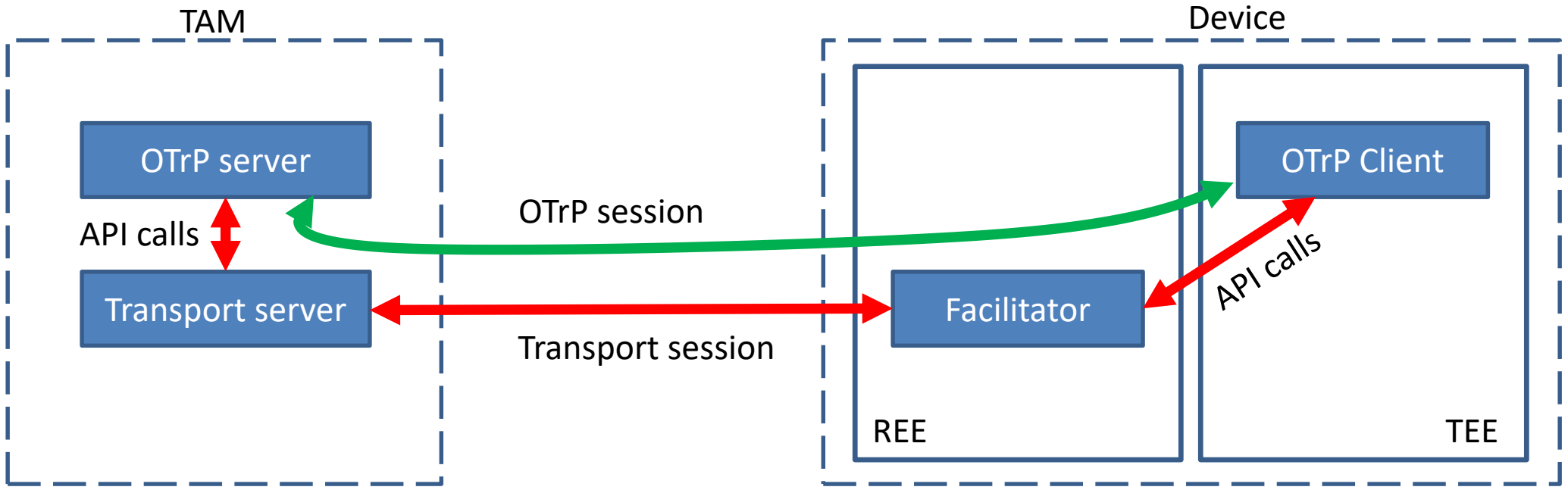
Subsequent messages from TAM are sent in transport session associated with the correct TEE

Every Rich App Talks to TAM?

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

OTrP Roles & Terminology

- The term “Agent” is confusing since as used in the doc, the Agent does not understand OTrP, just transport: “facilitator”? “connector”?
- Facilitator might be in rich app or app installer or whatever else



Service Provider Terminology

- Service Provider: An entity that wishes to supply Trusted Applications to remote devices.
- The architecture document also says:
  - A Device Administrator or Service Provider of the device needs to determine security-relevant information of a device before provisioning the TA to the device with a TEE.
  - A TEE in a device needs to determine whether a Device Administrator or a Service Provider that wants to manage an TA in the device is authorized to manage applications in the TEE.
- Dave Thaler: If the Device Admin is responsible for controlling what apps run in the TEE on an IoT class device (e.g., in a factory), is the Device Administrator just another type of SP, or are those terms disjoint?

Keys

(Not present in SGX)

Used for attestation

Used for Software signing

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

Root of Trust vs. Trust Anchor

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

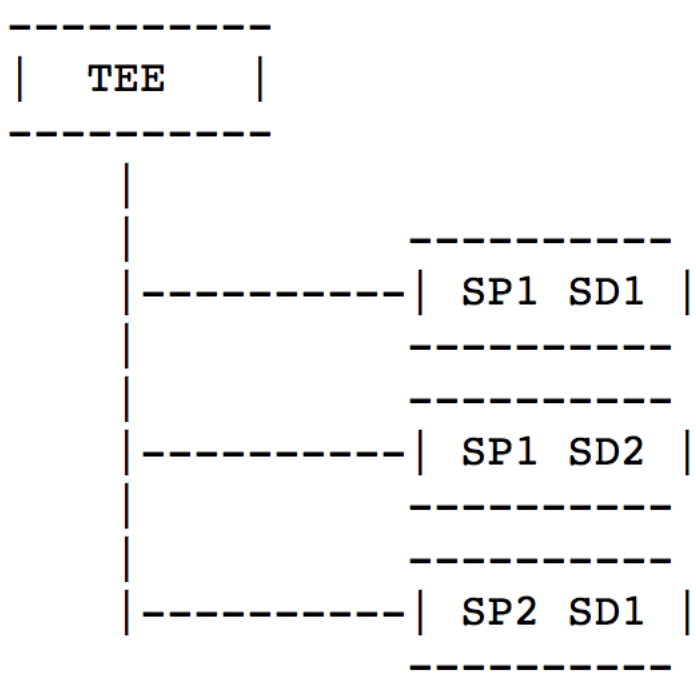
Root of Trust vs. Trust Anchor (cont.)

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

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

Security Domain Concept

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



TEEP Open Trust Protocol (OTrP) Draft

draft-ietf-teep-opentrustprotocol-01.txt

Mingliang Pei (*mingliang_pei@symantec.com*)

Andrew Atyeo (*andrew.atyeo@intercede.com*)

Nick Cook (*nicholas.cook@arm.com*)

Minho Yoo (*paromix@sola-cia.com*)

Hannes Tschofenig (*hannes.tschofenig@arm.com*)

IETF 102th, Montreal

Agenda

- Draft status update
- Main changes in the last version
- TEEP architecture and protocol implementation mapping
- Gap discussion and future work

Status Update

- WG draft approved 4/26/2018
  - Draft name change to *draft-ietf-teep-opentrustprotocol v00*
  - Minor changes from the previously draft discussed in IETF 101 WG

- Updated version v01
  - Split the draft into a architecture draft and the updated protocol draft
  - Architecture draft v00 was made more general, incorporating discussions in IETF 101 and mailing list

OTrP Design Quick Refresh

- Original TEEP architecture and protocol foundation before split
- Covers protocol part that implements TEEP architecture
- A message protocol
  - JSON-based messaging between TAM and TEE
- Use asymmetric keys and certificates for device and TAM attestation
- An OTrP Agent in REE is used to facilitate communication between a device TEE and a TAM
- Support a transport binding

OTrP Operations and Messages

✓ Remote Device Attestation

Command	Descriptions
GetDeviceState	• Retrieve information of TEE device state including SD and TA associated to a TAM

✓ Security Domain Management

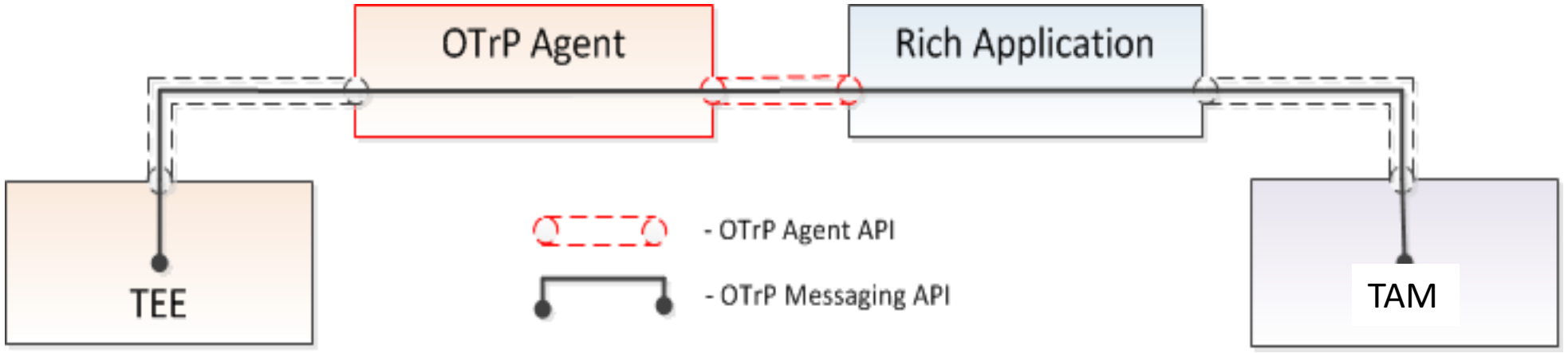
Command	Descriptions
CreateSD	• Create a SD in the TEE associated with a TAM
UpdateSD	• Update a SD or associated SP information
DeleteSD	• Delete a SD or SD related information in the TEE associated with a TAM

✓ Trusted Application Management

Command	Descriptions
InstallTA	• Install a TA in a SD associated with a TAM
UpdateTA	• Update a TA in a SD associated with a TAM
DeleteTA	• Delete a TA in a SD associated with a TAM

OTrP Message Exchange via an OTrP Agent

- An OTrP Agent handles how to interact with a TEE from a REE
- Most commonly developed and distributed by TEE vendor



OTrP JSON Message Format and Convention

```
{
   “<name>[Request | Response]”: {
      "payload": "<payload contents of <name>TBS[Request | Response]>",
      "protected":"<integrity-protected header contents>",
      "header":  <non-integrity-protected header contents>,
      "signature":"<signature contents>"
   }
}
```

For example:

- CreateSDRequest
- CreateSDResponse

Changes from the prior version

- Moved general architecture specification into the architecture draft
  - Adjusted introduction part to link with the architecture draft
  - Referred to Architecture draft to definitions and terminologies
  - Referred to Architecture doc for general architecture requirements
  - Retained the most part of entity relationship, certificate types, and OTrP Agent as part of Architecture to OTrP mapping reference
- No changes in API and messages
- Changed to make Trusted Firmware (TFW) check optional
  - TAM will decide whether a TEE acceptable in the absence of TFW signature
- Terminology update
  - Use TFW in all occurrences of Secure Boot Module (SBM)

TEEP Architecture to Implementation Mapping

- Mostly mapped implementation except a few new architecture expansion requests from mailing list
- Multiple TEE support
  - TEEP architecture proposes to expand single active TEE in a device to allow multiple full TEEs
- TA binary distribution by a Client Application
  - OTrP currently requires TA binary be distributed by a TAM and sent in an encrypted form
  - Issue in authorizing a Client Application and TA personalization data
- Use of an Agent for communication between a TEE and a TAM
  - Discussion around making it optional

Gap Discussion and Future Work

- Multiple TEE support
  - TEE identifier needs to be made visible to an OTrP Agent
  - OTrP Agent isn't just relaying anymore; add routing capability to a target TEE
  - Other options
- TA binary distribution by a Client Application
  - Installation can be addressed
    - The signer of TA is trusted by a TEE
  - Issues with SD update and TA update in future
  - Issues to send device specific data that a TA needs to use
- Communication between a TEE and TAM might be facilitated by OS
  - A Rich App may not need to call OTrP Agent itself

Q&A

Thank you!

Message Format Negotiation

- A Client Application may query a device for its preferred message format
- A Client Application triggers TAM to send messages in a preferred format
- Use a default message format

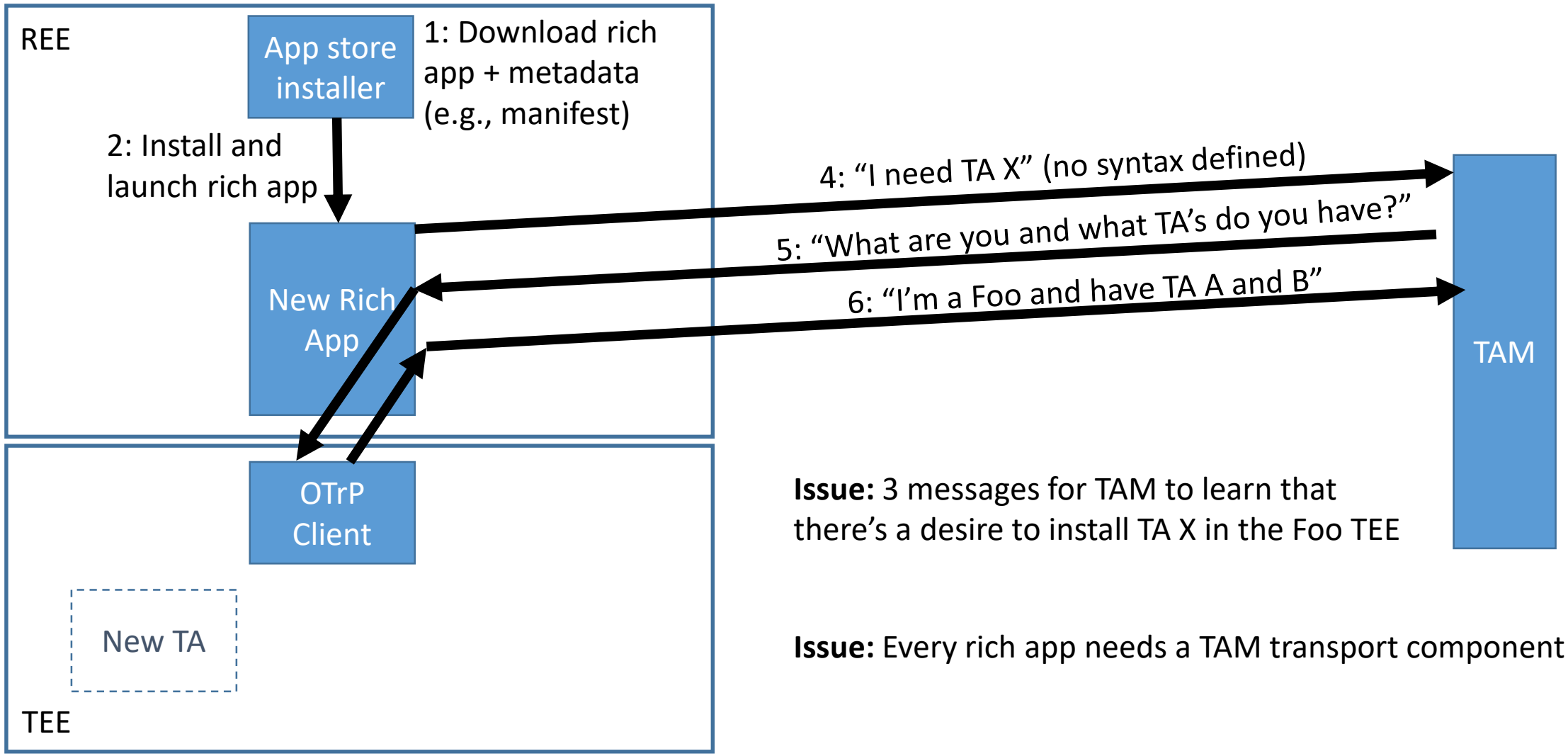
IETF 102
Hackathon Report

Dave Thaler

10 issues found starting implementation (1/4)

1. Section 6.5 explains that the TAM needs to receive a list of one or more TA's that are requested to be installed (S6.5.1 point 1A). However, **no message is defined for doing so, which prevents interoperability**. I think this message needs to be generated by the TEE (not the rich app),
   for reasons I will explain in #3 below.
2. Section 6.5 explains that the **TAM needs to keep track of TAs installed** on all devices, even though its list might be wrong. This has a scalability issue. Instead, I think there should be no such requirement.
   - See David Wheeler's presentation for why this requirement is problematic anyway
3. Putting my issue #1 and issue #2 together means there's an **extra round trip that is unnecessary**. 6.5 says the TAM receive a list of TAs needed, and then the TAM just goes back and asks what is installed, just to get a list of what needs to be installed. This is unnecessary, the TEE can just send a list of one or more TAs that need to be installed and aren't already. Hopefully this explains why I said in issue #1 above why I think the message needs to be generated by the TEE.

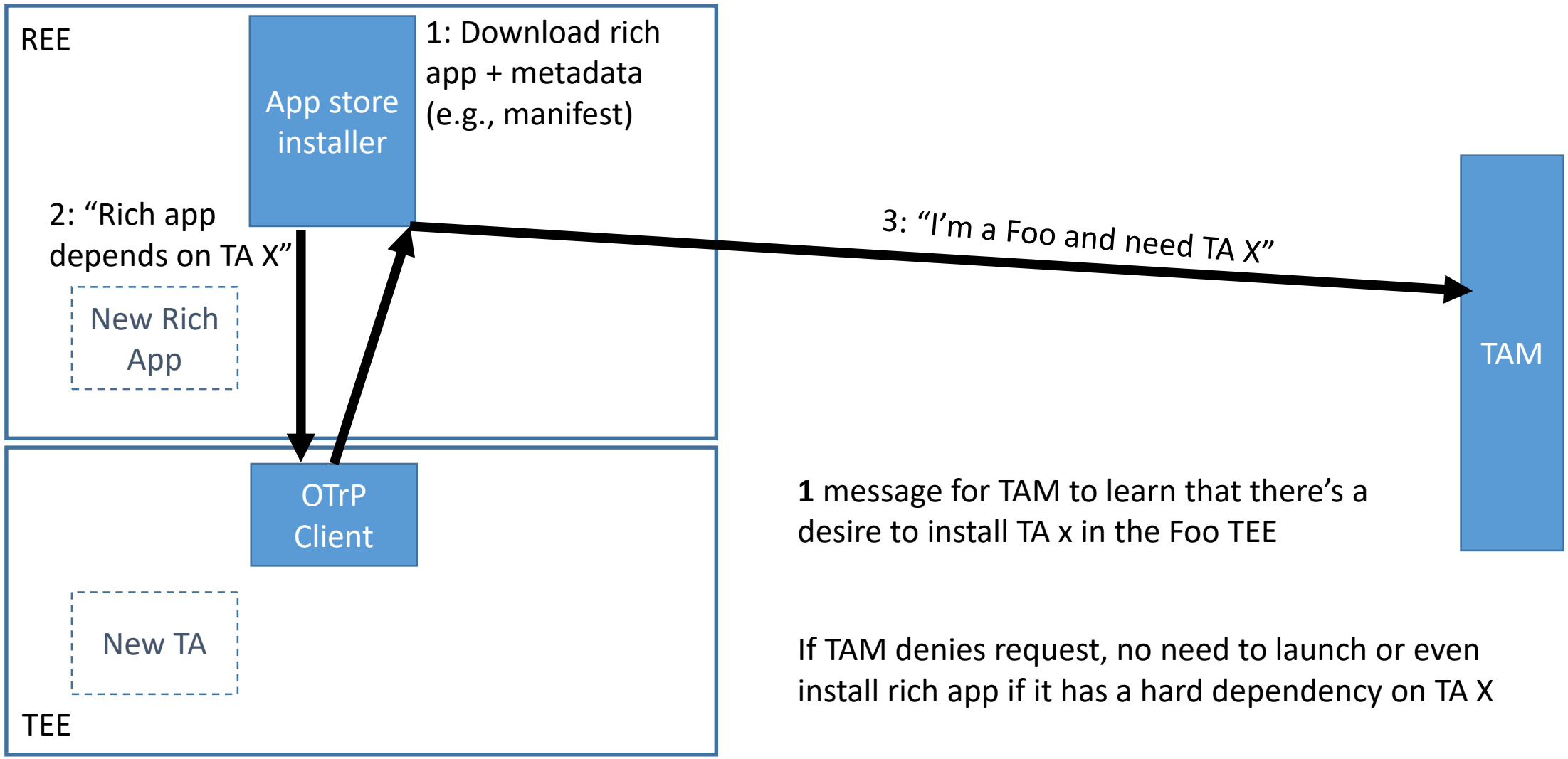
Connection model #1: what the draft says



Issue: 3 messages for TAM to learn that there's a desire to install TA X in the Foo TEE

Issue: Every rich app needs a TAM transport component

Connection model #2: what I wanted



1 message for TAM to learn that there's a desire to install TA x in the Foo TEE

If TAM denies request, no need to launch or even install rich app if it has a hard dependency on TA X

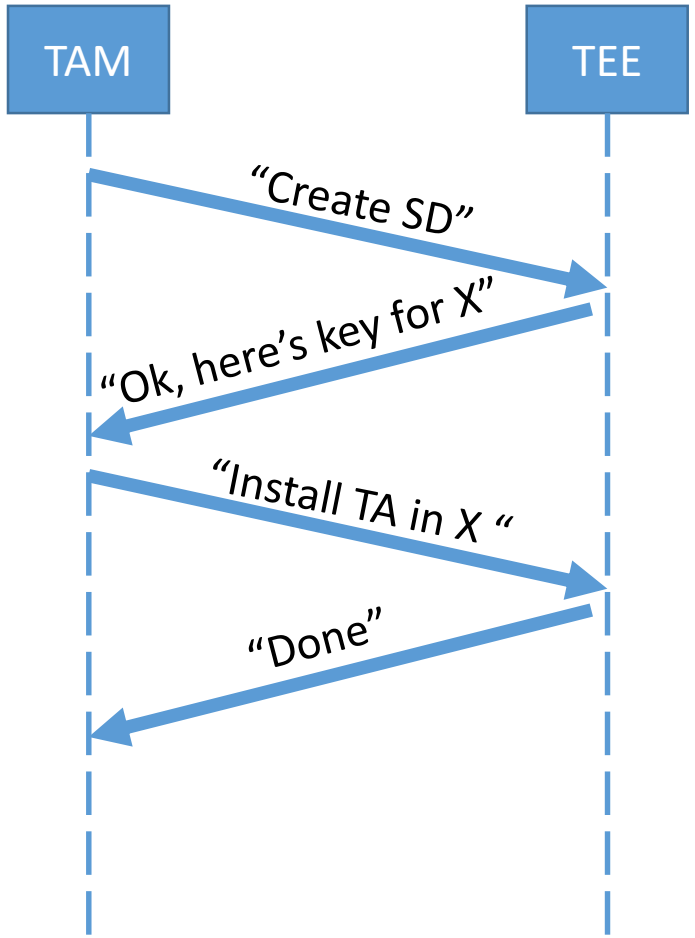
10 issues found starting implementation (2/4)

4. Section 9.1.1 requires a list of OCSP stapling data, but as far as I can see, the document provides **no information or citation about the correct format** for such data.
5. The **“did” field** in 9.2.1.1 seems to be either (a) **redundant** and should be removed, **or** (b) **missing from other messages** like Install TA. The text explains the field is to check that the message was received by the right device. My opinion is that since the TEE has to trust the TAM anyway, it’s the TAM’s responsibility to send messages to the right device over an authenticated channel (whether encrypted or not). So I think it should be removed.
6. Some fields, e.g. “signerreq”, have boolean values that are encoded as strings (“true”, “false”). I think these should be **boolean types, not strings**, which would also have the advantage of better compression if we can use CBOR encoding.

10 issues found starting implementation (3/4)

7. Section 6.5.1 point 9.A implies that to install a TA, one must have an **extra round trip to create an SD** first if one isn't already there. I would expect one common case to be where there is one TA per SD, so that all TAs are isolated from each other. As such, requiring the extra delay is inefficient in time, bandwidth, and processing. All the fields in CreateSD are already present in an InstallTA message (except the "did" field mentioned above in issue #5), so it could be done automatically by the first InstallTA message itself.
8. The scope of **uniqueness of the "rid" and "tid" fields is underspecified**. They just say "unique". I think "rid" is just supposed to be unique within a given {session,"tid"} but I can't tell for sure. And I think "tid" is just supposed to be unique within a given session (not globally across all sessions, all TAMs, all devices), but I can't tell. They're also formatted as strings, but I'm not sure why they can't be **integers** which I think **would be much more efficient**.

CreateSD vs InstallTA



- Key can optionally be used to protect TA binary/data so TAM etc. cannot see them
- If no encryption:
  - CreateSD exchange carries no information that couldn't be piggybacked in InstallTA and be more efficient
- If encryption:
  - Requires key per {SD,TEE} (e.g., per {TA,TEE}), vs. just one per {SP,TEE}.

Don't convolute keys and TA isolation boundaries

10 issues found starting implementation (4/4)

9. I found it confusing that the names of the messages don't match the name values in the messages themselves ("GetDeviceStateResponse" vs "GetDeviceTEEStateTBSResponse", etc.) Having these not match is bug-prone.
10. It's unclear whether a rich app can depend on two TA's from different TAMs, and whether a TA can depend on a TA from a different TAM. In the use case where the device admin runs the TAM and controls all TAs on their devices the answer would be no. But in other use cases I'm not sure. If so, then the question arises about how dependencies are expressed and whether a dependency needs to express which TAM is used. This then begs the questions of whether a TA might be via more than one TAM, or might change TAMs over time. The answers here probably belong in the arch doc.

Recommendations for TEEP Support of Intel® SGX Technology

Overview of SGX & Selected TEEP Topics

David M. Wheeler

david.m.wheeler@intel.com

Apologies…

- If you are really interested in the details of SGX

  This Won't Satisfy Your Curiosity

- The best public paper can be found at:
  - Intel® SGX Explained https://eprint.iacr.org/2016/086.pdf
  - Stanford Seminar YouTube: https://www.youtube.com/watch?v=mPT_vJrlHlg
  - Other Resources: https://software.intel.com/en-us/sgx/resource-library
    https://software.intel.com/en-us/sgx/academic-research

Please refrain from asking deep questions on SGX Architecture that are not relevant to TEEP ☺

We are under a Time Constraint

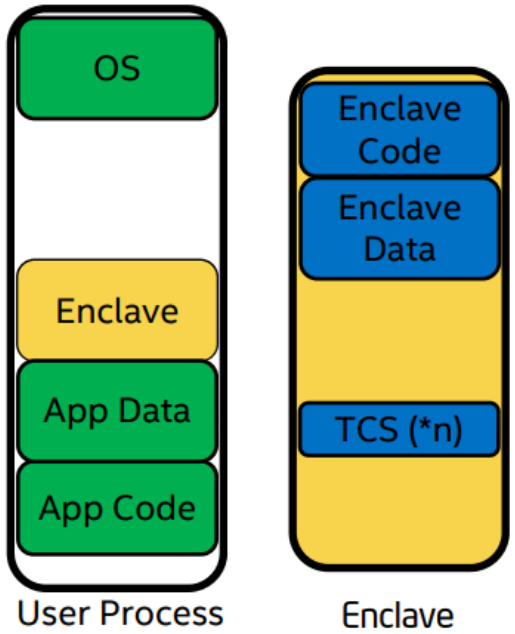
What We Will Cover

- Overview of Intel® Software Guard Extensions (SGX)
- SGX TCB (Trusted Computing Base)
- SGX Attestation

Overview of Intel® SGX

- SGX provides a protected area of memory (EPC Memory) where enclave code and data exist
- Enclave code is packaged with the Client Application, and loaded into EPC Memory by the Application
  - To the REE, both memory areas are within the same process
  - Enclave is prevented from access into Client process memory, AND Client prevented from access to the EPC (see flow chart)
- A special enclave (Launch Control) is used to load an enclave with code and data, and the Launch Control verifies the code during loading (authentication, authorization & integrity)
  - The Enclave must be signed by the Launch Key
- Entering and Exiting the enclave are done through processor instructions
  - EENTER and EEXIT

Trusted execution environment embedded in a process



With its own code and data

Provide Confidentiality

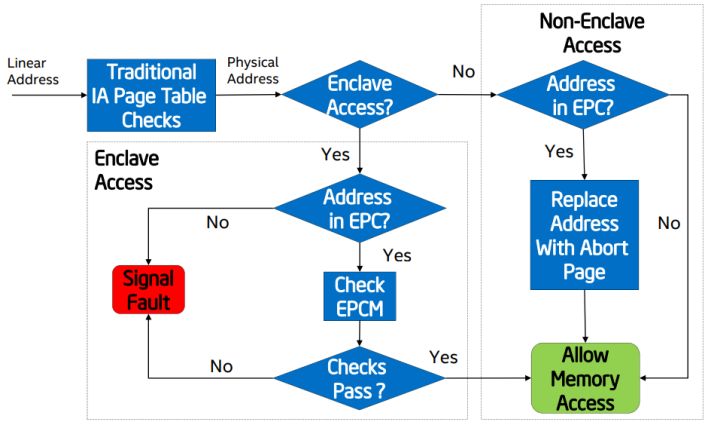
Provide integrity

With controlled entry points

Supporting multiple threads

With full access to app memory

SGX Access Control



EPC = Enclave Page Cache

What is Relevant to TEEP

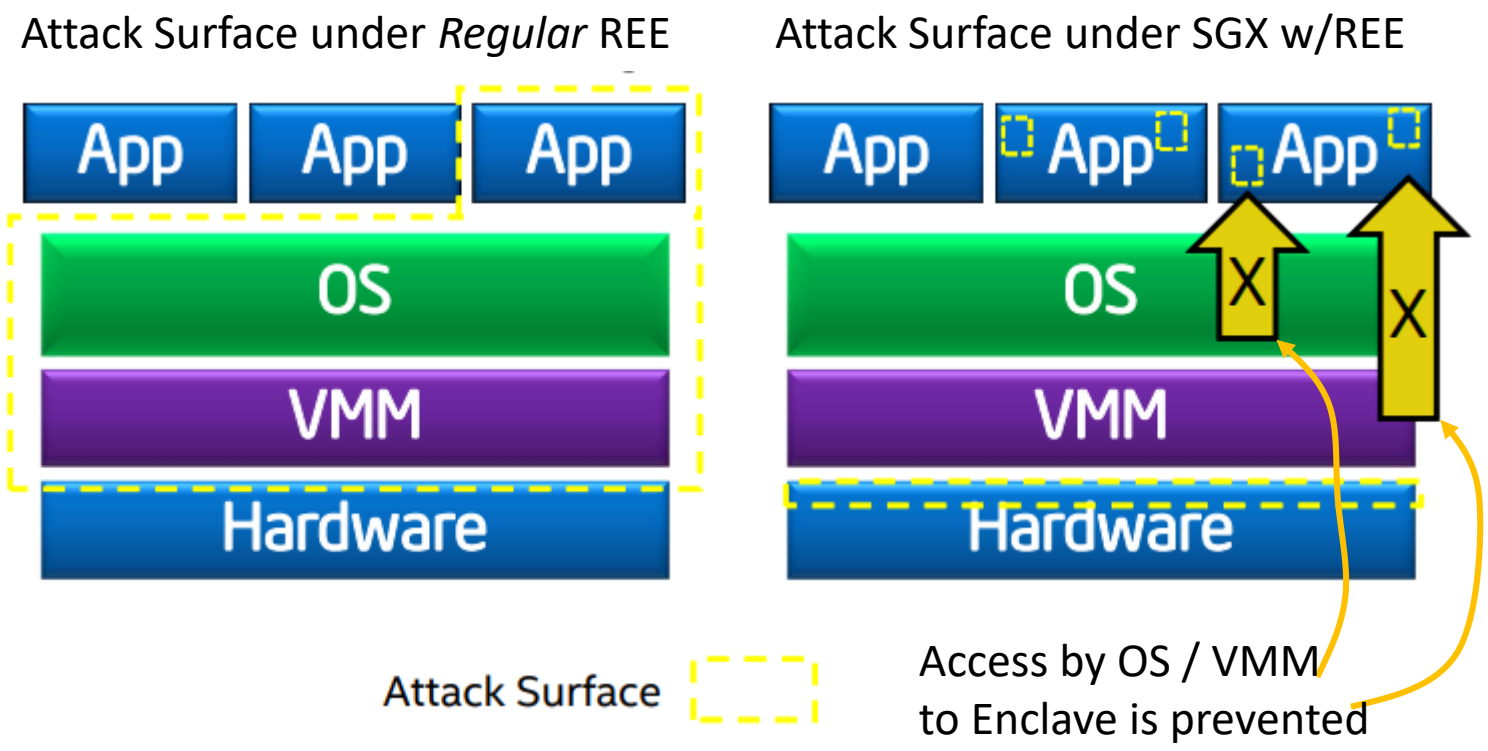
- Trusted Application is **not** Separate from Client Application
  - SGX Applications include both the trusted part (Enclave) and the untrusted part (Client Application)
  - This doesn't prevent a Client Application from presenting all information needed to "authorize" an SGX application to a TEEP Agent
    - Some information is embedded in the enclave
      - Authorized TAM or Service Provider (Mr. Signer), Integrity Proof (Mr.Enclave), Other Rights
    - Other information can be provided by the Client App
      - TAM Identity & authorization signatures, Other stuff?
- There is **no** Security Domain
  - Only one "program" can be loaded a single enclave – multiple separate enclaves can exist simultaneously
  - One can consider an Enclave as a single domain for only one TA
  - Optionally, an implementation of a TEEP Agent can manage TA interactions "as if" they were in the same Security Domain (e.g. secret sharing, secure channels, etc.)
- There is **no** internal Agent watching all Enclaves
  - It isn't possible to report on all the "installed" TAs – installed TA's take no resources from the TEE until loaded
  - It isn't possible to report on all the "running" TAs – as they do not know about each other
    - TEEP Agent could report on all TA's that it loaded as running enclaves
    - Launching an application that contains an enclave does not mean the enclave gets loaded

How would SGX Use TEEP?

- **Install / Uninstall**
  - There is no *real* install/uninstall commands in SGX
    - Any application on the platform file system can carry SGX enclave code (a TA)
    - Same vector as any REE Application install (e.g. HDD, Flash, USB Stick, Network, etc.)
  - One option could be signing the SGX enclave code (TA) so that it can be launched
    - For example:
      1. Service Provider requests TAM to prepare a particular Application for an SGX Platform (e.g. Install)
      2. The TAM holds the Enclave Signing Key for some platforms
      3. TAM authorizes SP, and if OK, then signs the requested Enclave & delivers it to the Platform
    - Simplifies Application Developer deployment
- **Start / Stop**
  - An SGX enclave is launched (Started) by the application (not by the TEE)
  - TEEP Start could be mapped to Client Application launch
    - However, the Client Application can delay the launch of the enclave to a later time

Intel® SGX Trusted Computing Base

- Security Perimeter is the CPU package boundary
  - Data/Code inside CPU is unencrypted
  - Data/Code outside CPU is encrypted/integrity protected
- BIOS is formally outside the TCB
  - BIOS controls how much memory is allocated to EPC, but cannot affect the security of EPC memory
- OS is formally outside the TCB
  - OS controls page tables, but does not control the security or attributes of the pages
  - Interrupts and certain OS features (files, network sockets) are still handled by OS, but considered in Application scope/control
    - State is saved in special EPC memory area for interrupts and context switches



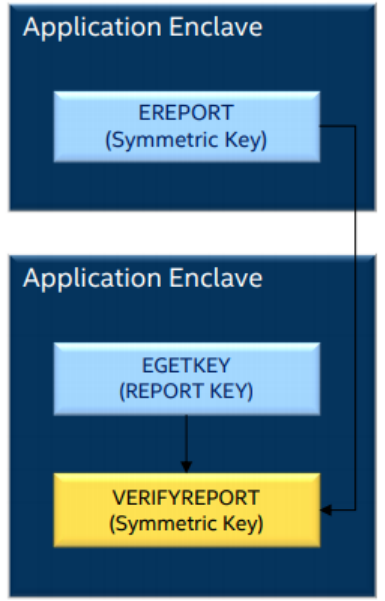
What is Relevant to TEEP

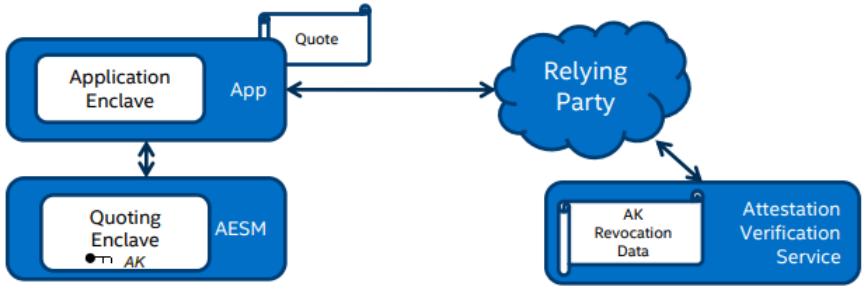
- SGX does not depend on Secure Boot
  - SGX has it's own Roots of Trust for:
    - Measurement (RTM), Integrity (RTI), Verification (RTV),
    - Confidentiality (RTC), Reporting (RTR), Storage* (RTS)
- On an SGX platform, Secure Boot may NOT be turned on
  - Not possible to report from SBM

* The RTS is limited to providing sealing keys – actual storage is based on OS services

Intel® SGX Attestation

- **SGX Includes Two forms of Attestation**
  - Local Attestation – Hash Based
  - Remote Attestation – EPID Signature (Elliptic Curve Group Signature)
- **Local Attestation**
  - AES–CMAC Key Generated from Enclave & Platform
    - Attributes of the Enclave (Signer, Integrity Measurements, Version, etc.)
    - Platform Attributes (Fuses, Microcode Version, CPU Serial #, etc.)
  - Allows inclusion of other message via Hash
  - Can be sent to other enclaves on same platform
- **Remote Attestation**
  - Built from Local Attestation by SGX Signing Enclave
  - Only Signing Enclave has access to EPID key (the RTR)
  - Requires an External Verifier for EPID Signatures





What is Relevant to TEEP

- **Add EPID Digital Signature Algorithm as Optional to Support**
  - Will be supported by default on SGX-Enabled Platforms
  - Must be supported by TAMs to consume attestation from SGX Platforms
    - Or offload to an Intel® SGX Verifier
  - Needed to support SGX Attestation Signatures
  - The only way to verify trust in an SGX Enclave
  - Can use an Attestation to “certify” another RSA or ECDSA key pair
    - This would enable SP to have an Application/TA-specific RSA or ECDSA key pair

- **Local Attestation can be used to provide communication between the TA’s and a TEEP Agent**
  - Can be used to simulate Security Domains and “Universal TEE Knowledge” for reporting state

ISO/IEC 20008-2 Known Patent Rights

- The following are the known (to me on 7/16/2018) IPR claims on EPID
- I make no claim on the part of Intel or other parties that this list is complete or accurate

1. ISO/IEC 20008-2 (EPID Group Signature)
   - NEC corporation – RAND/reciprocal
   - Electronics and Telecommunications Research Institute (ETRI) – RAND/reciprocal

1. ISO/IEC 20009-2 (SIGMA Protocol: P2P Attested Channel)
   - China IWNCOMM Co., LTD. – RAND
   - Electronics and Telecommunications Research Institute (ETRI) – RAND/reciprocal

Other Crypto Recommendations

- **NIST Recommends moving to larger Key Sizes**
  - NIST Recommendations

Algorithm
RSA 3072-bit or larger
Diffie-Hellman (DH) 3072-bit or larger
ECDH with NIST P-384
ECDSA with NIST P-384
SHA-384
AES-256

Date	Minimum of Strength	Symmetric Algorithms	Factoring Modulus	Discrete Logarithm Key	Discrete Logarithm Group	Elliptic Curve	Hash (A)	Hash (B)
(Legacy)	80	2TDEA*	1024	160	1024	160	SHA-1**	
2016 - 2030	112	3TDEA	2048	224	2048	224	SHA-224 SHA-512/224 SHA3-224	
2016 - 2030 & beyond	128	AES-128	3072	256	3072	256	SHA-256 SHA-512/256	SHA-1
2016 - 2030 & beyond	192	AES-192	7680	384	7680	384	SHA-384 SHA3-384	SHA-224
2016 - 2030 & beyond	256	AES-256	15360	512	15360	512	SHA-512 SHA3-512	SHA-256 SHA-512/256 SHA-384 SHA-512 SHA3-512

https://www.iad.gov/iad/library/ia-guidance/ia-solutions-for-classified/algorithm-guidance/assets/public/upload/CNSA-Suite-and-Quantum-Computing-FAQ.pdf

- **Minimal to support should be**
  - RSA-3072, RSA-4096, RSA-2048
  - ECDSA using NIST P-384, NIST P-256
  - ECDSA using Ed448-Goldilocks, Ed25519
  - EPID 2.0 Group Signature (Elliptic Curve w/ Bilinear Maps, TCG DAA group signature scheme)
    - Based on ISO Standard – ISO/IEC 20008-2:2013 Information technology -- Security techniques -- Anonymous digital signatures -- Part 2: Mechanisms using a group public key
    - https://software.intel.com/en-us/articles/intel-enhanced-privacy-id-epid-security-technology

What TEEP Services are Relevant to SGX?

- **TEEP on SGX will likely operate much differently than on a TZ platform**
  - The TEEP Agent's counterpart "inside the TEE" will be an enclave just like every other enclave
    - The *TEEP Service Enclave* will perform services as if it were managing the whole TEE
      - But can only manage Enclaves that "cooperate" – have a TEEP Agent Helper library as part of their enclave/App
    - The *TEEP Service Enclave* will provide information on a "best effort" basis – may not know about all enclaves
  - TEEP will only "see" the applications installed/started/stopped through TEEP
- **Get Device State is a "Best Effort" Service**
- **Install/Uninstall a TA is equivalent to same operation on a Client Application**
  - TEEP Agent can report on TAs installed through TEEP, but not on ALL TAs/Applications
  - TEEP Agent cannot prevent TAs / Client Applications from being deleted (Denial of Service)
  - TEEP Agent may not be able to delete / remove a TA (depends on implementation)
- **TEEP Services are Useful in an SGX Environment, but will be limited**

Summary Recommendations for TEEP SGX Support

- TEEP **MUST** support TA delivery within a Client Application
- TEEP **MUST** support EPID Signature Algorithm as Optional
- TEEP **SHOULD** look to support longer key sizes due to Post-Quantum recommendations
- TEEP **SHOULD NOT** require Secure Boot Attestation
  - SBM and TFW are not required of all platforms
  - Attestation Report should be flexible, allowing only required platform-specific elements
- TEEP **SHOULD** further explore the Security Domain Concept and only if valuable and necessary, then develop a crisp definition and model for Security Domains
  - This crisp model should encompass platforms that create SD's of size One
- TEEP **MUST AVOID** definitions of operations that are very platform specific
  - Secure Boot, specific types of reporting, and platform state
  - Some reporting needs to be considered 'Best-Effort' or contain a quality-of-reporting