



Independent subsidiary of Qualcomm Innovation Center, Inc.

Kernel TEE Subsystem Evolution

Sumit Garg

Senior Staff Engineer, QUALCOMM India Private Limited

#OSSummit



Who am I?

- A member of the Foundries.io team. Foundries.io is an independent subsidiary of Qualcomm Innovation Center, Inc.
- Have a keen interest in platform security
 - Linux kernel: TEE subsystem reviewer
 - OP-TEE: Author for Rust no-std Trusted Applications support
 - TF-A and OP-TEE: firmware encryption framework maintainer
 - TF-A and OP-TEE: platform maintainer
- Other areas of interest
 - U-Boot: OF_UPSTREAM devicetree maintainer
 - ftrace for OP-TEE
 - meta-arm-toolchain

What is TEE?

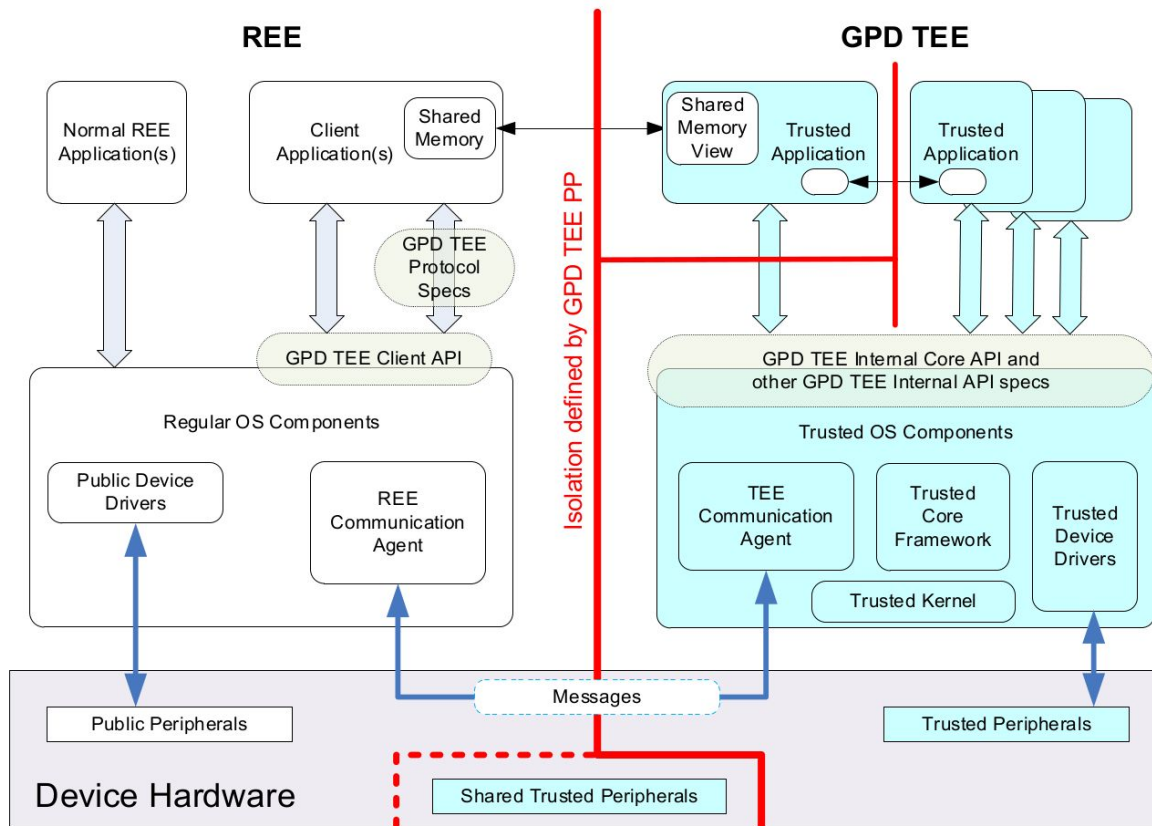
Trusted Execution Environment

- Isolated execution environment running alongside Rich Execution Environment (REE) like Linux OS.
- Provides capability to isolate trusted code and corresponding resources like memory, devices, etc.
- Isolation is backed by hardware security features such as Arm TrustZone, AMD Secure Processor, RISC-V Trusted domain etc.

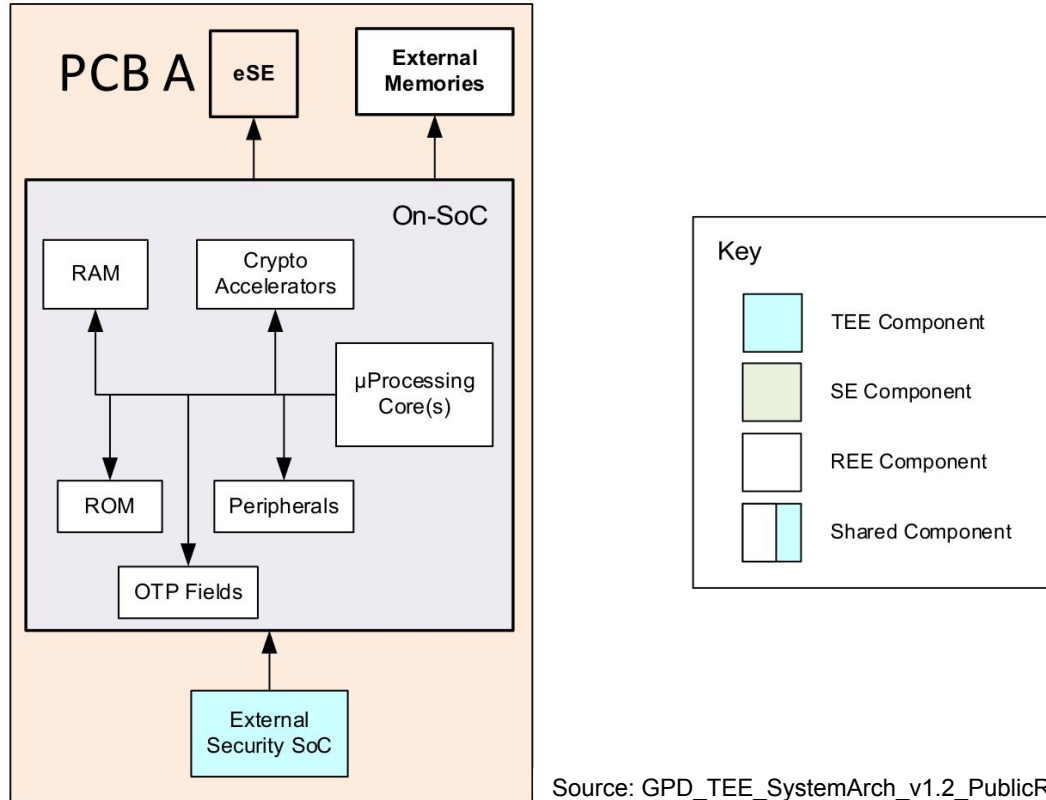
Industry standard: GlobalPlatform Defined TEE ([specs](#))

Generic TEE Software Architecture

Figure 3-1: TEE Software Architecture

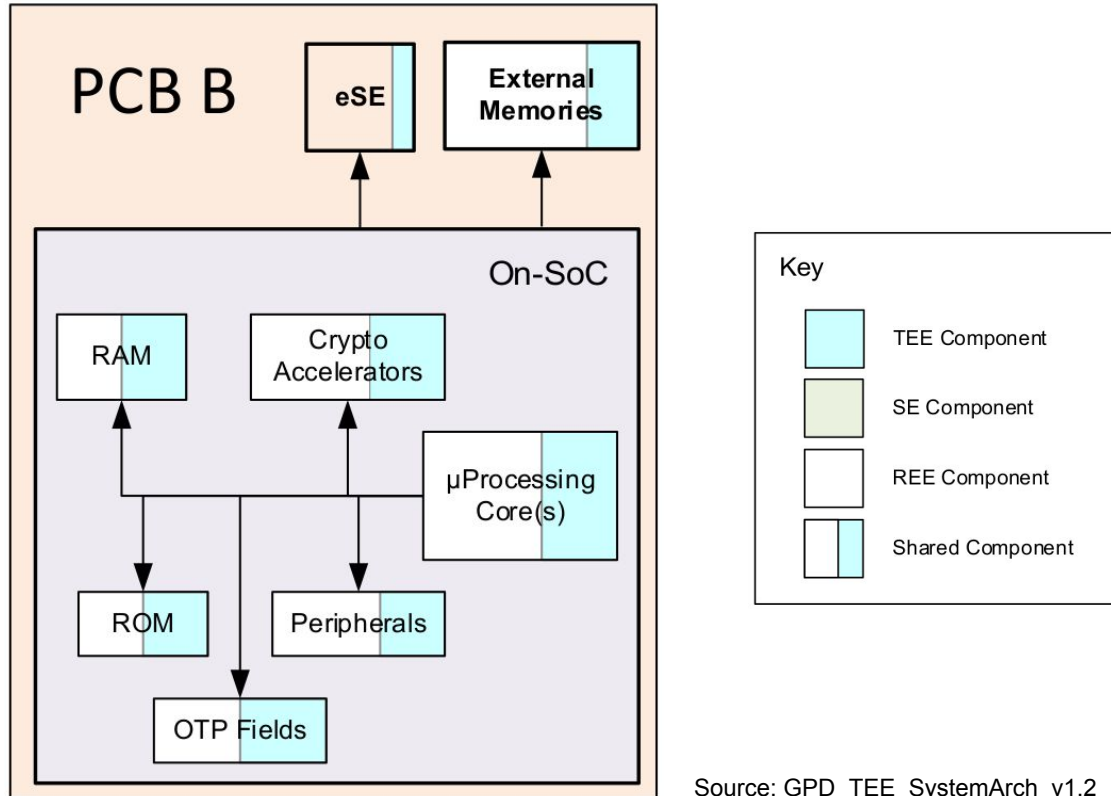


Hardware Realizations of TEE (External)



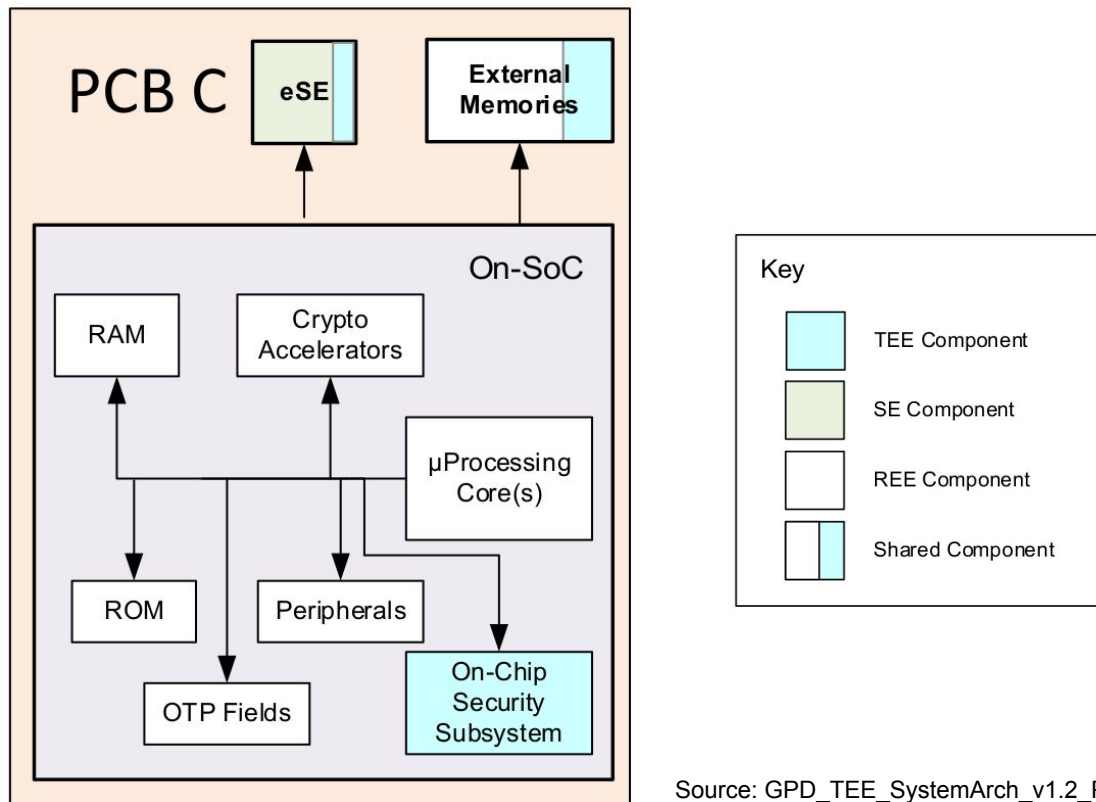
Source: GPD_TEE_SystemArch_v1.2_PublicRelease

Hardware Realizations of TEE (Shared)



Source: GPD_TEE_SystemArch_v1.2_PublicRelease

Hardware Realizations of TEE (On-chip)



Source: GPD_TEE_SystemArch_v1.2_PublicRelease

Open Portable TEE (OP-TEE)

An open source TEE implementation with **mainline** Linux kernel driver

Supported architectures: arm, arm64 and risc-v (early stages)

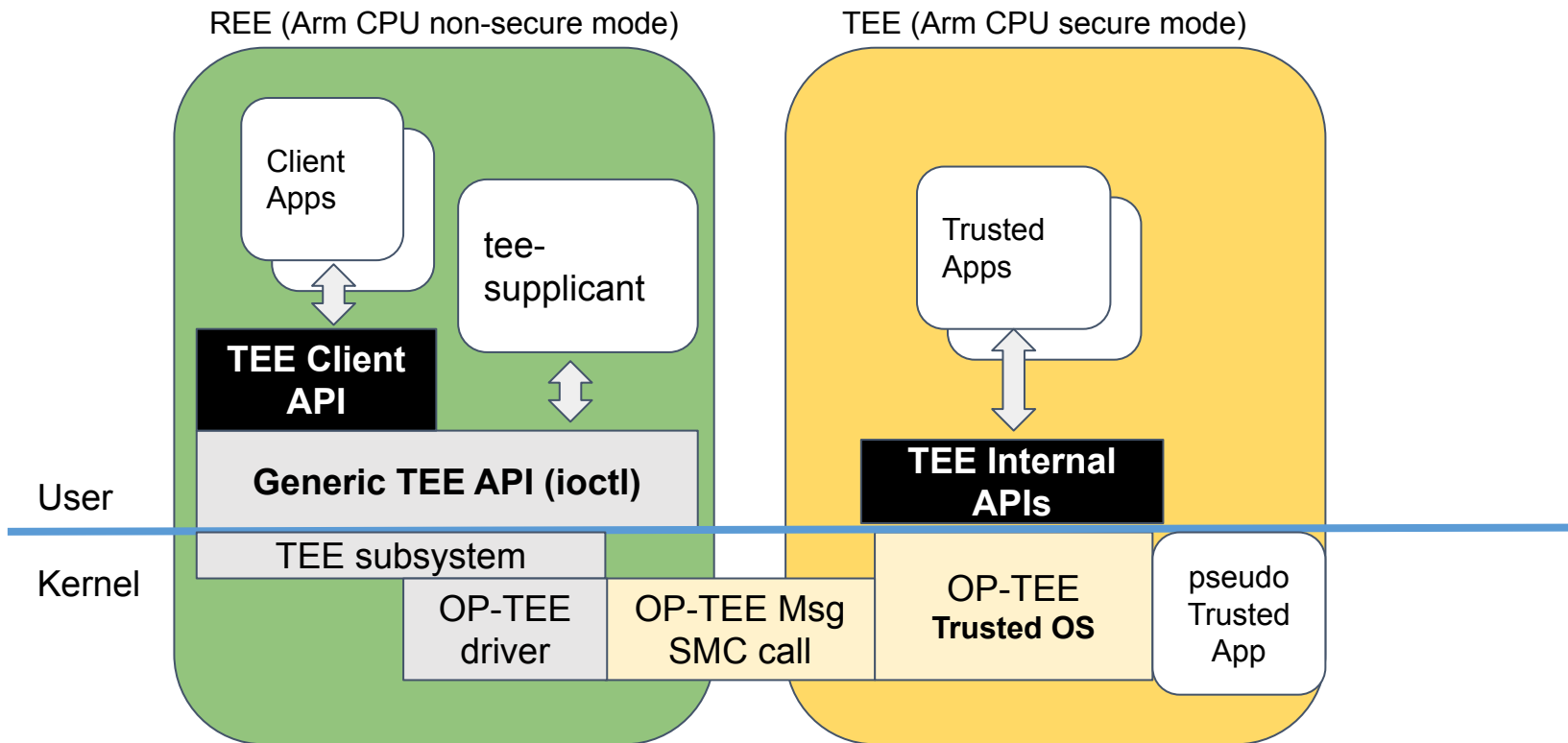
Design goals:

- Isolation
- Small footprint
- Portability

You don't need to have actual hardware to get hands-on experience with OP-TEE:

- Use Qemu, steps [here](#).

OP-TEE architecture (Arm TrustZone)



Kernel TEE subsystem

Introduced in Linux kernel version 4.12

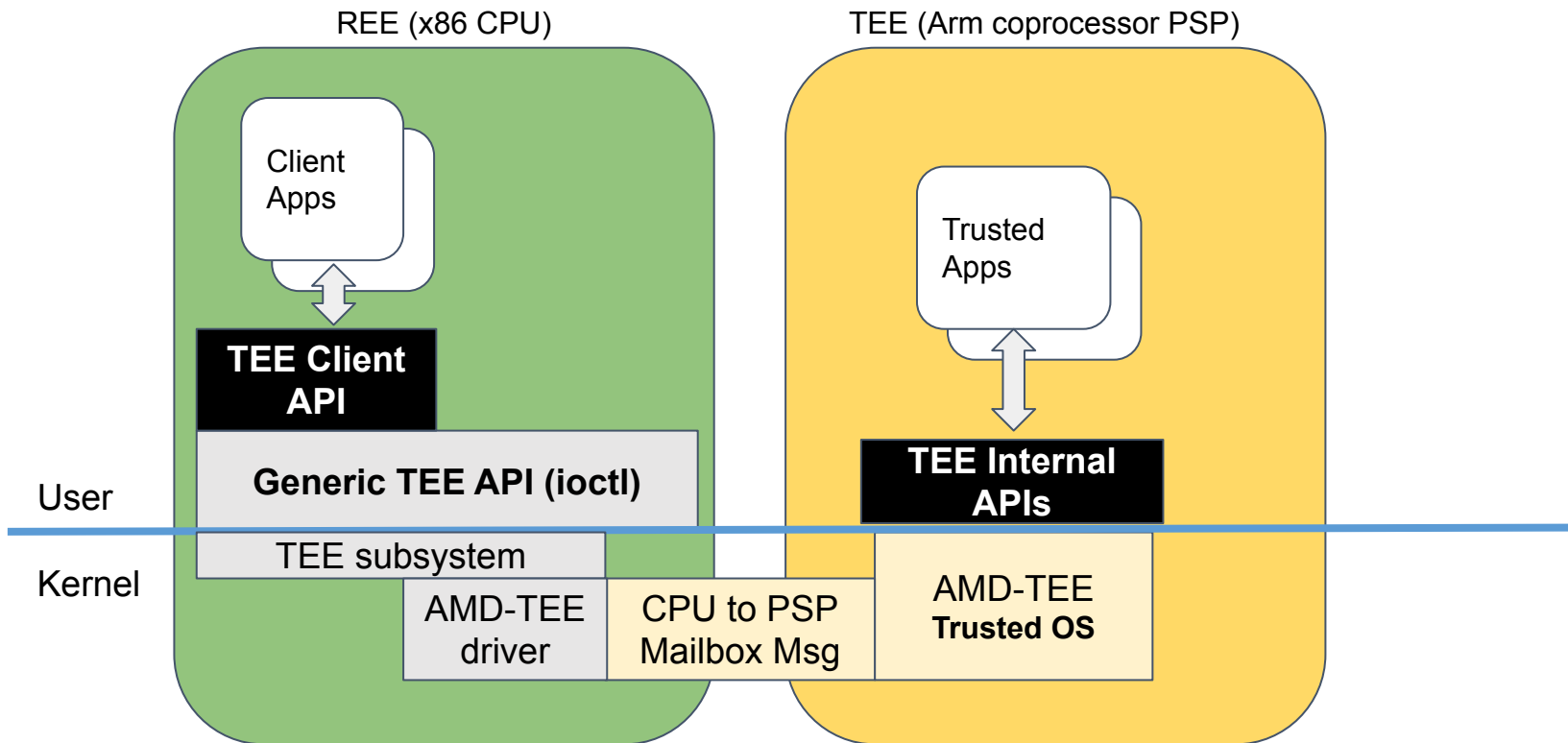
TEE subsystem

- Registration/unregistration of TEE drivers
- Shared memory between normal world and secure world
- User-space TEE IOCTL interface

OP-TEE driver

- Targets ARM and ARM64
- Shared memory support via reserved memory
- TEE supplicant support

Evolution with AMD-TEE



Evolution with TEE bus

Allows kernel to automatically react to the TAs provided by TEE in order to load and probe drivers corresponding to these TAs.

The TAs are identified via **Universally Unique Identifier (UUID)**. The drivers need to register a table of supported TA UUIDs.

TEE bus framework registers following APIs:

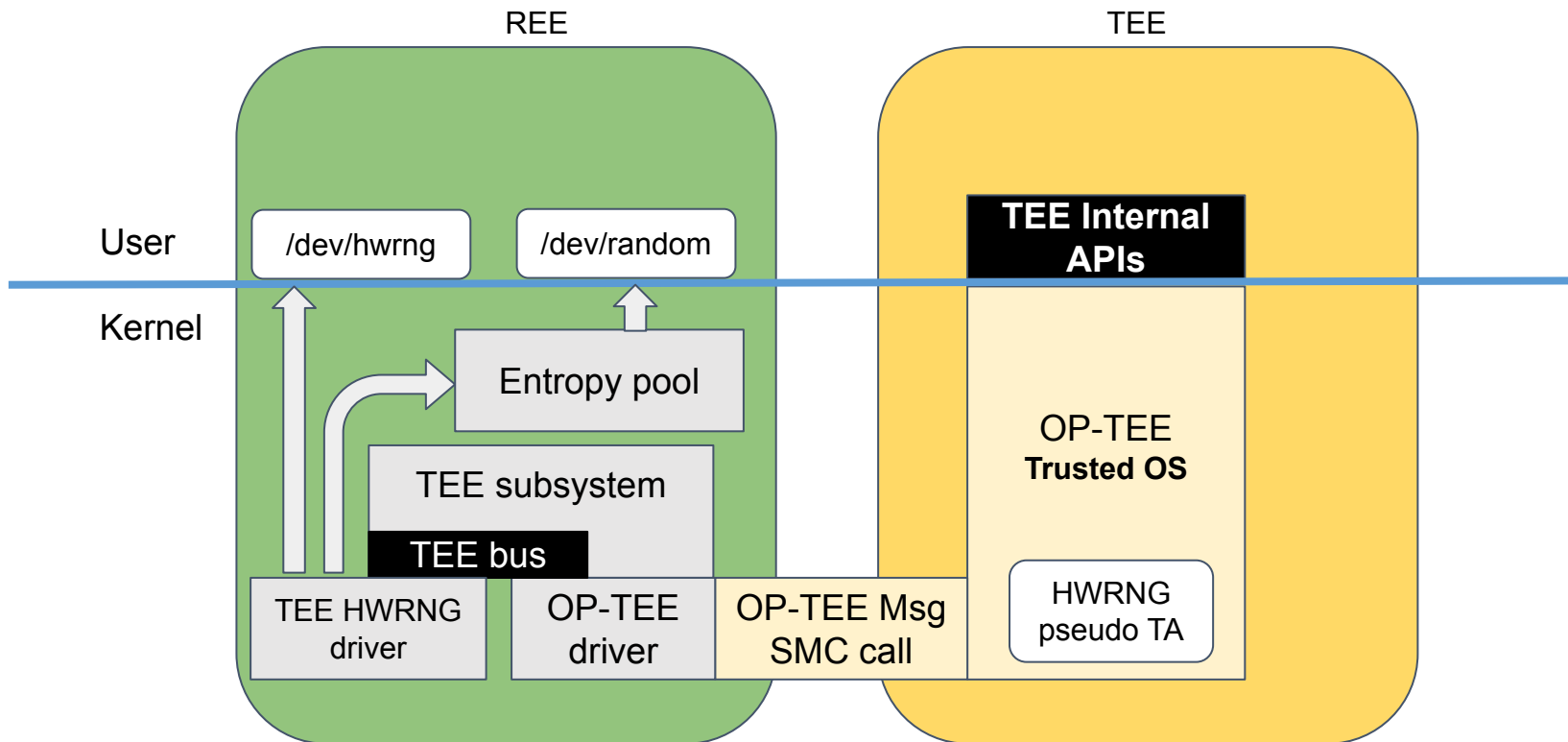
- **match()**: iterates over the driver UUID table to find a corresponding match for TA UUID.
- **uevent()**: notifies user-space (udev) whenever a new TA is registered on TEE bus for auto-loading of modularized drivers.

TEE bus enumeration

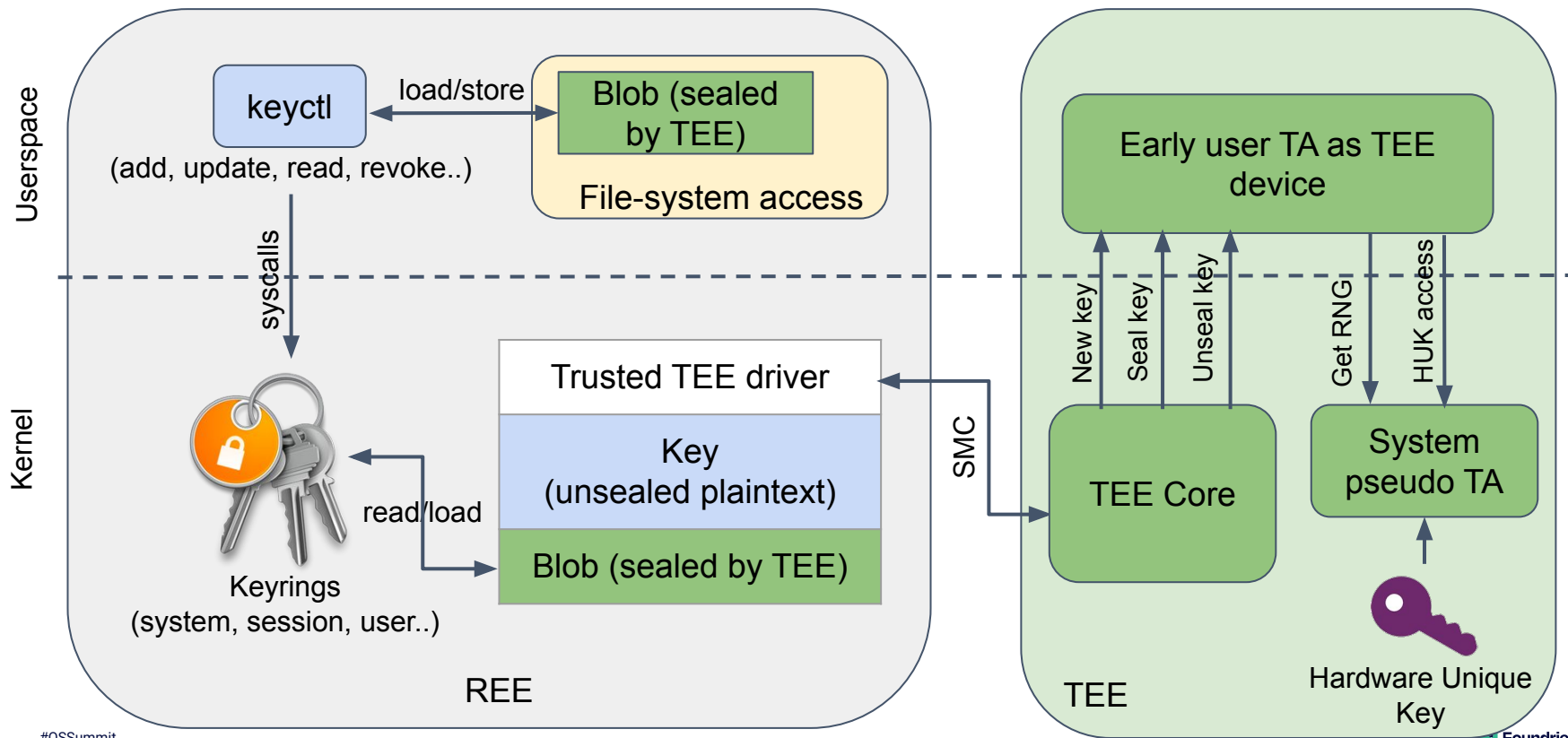
Linux kernel TEE bus framework allows for device enumeration to be **implementation specific** like for OP-TEE etc. OP-TEE provides a pseudo TA to enumerate TAs which can act as devices/services for TEE bus.

To enumerate devices a device pseudo TA function is invoked to fetch an array of device UUIDs, which are then registered with TEE bus as "**optee-ta-<UUID>**" devices.

HWRNG use-case



Trusted Keys use-case



Firmware TPM use-case

A firmware TPM with regards to OP-TEE is implemented as a trusted application. It works transparently with the existing TSS stack as compatible with a regular TPM chip as possible.

Microsoft's fTPM

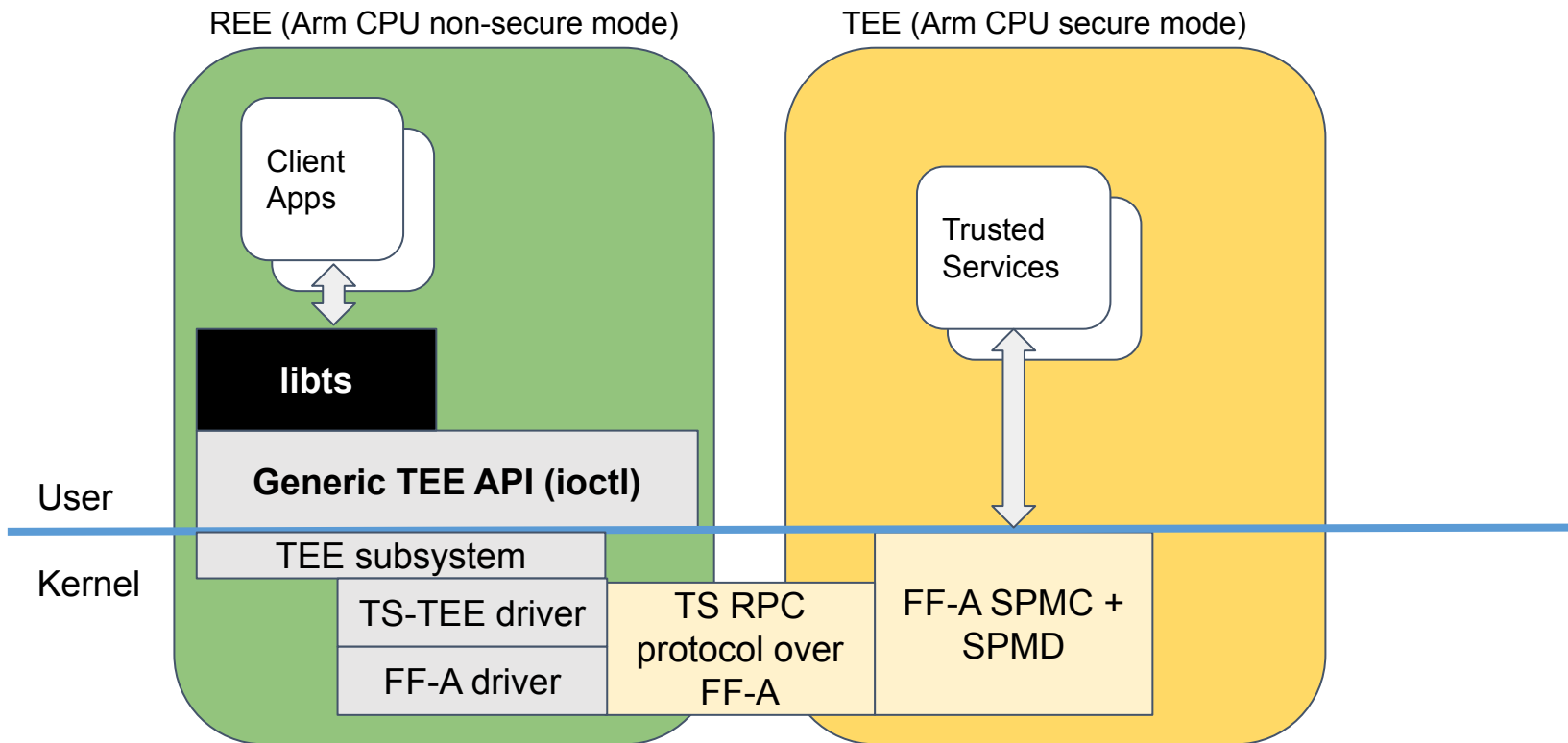
- TCG TPM 2.0 spec has lots of code written in C (main contributor is Microsoft).
- Microsoft took that code and created fTPM.
- Runs on Arm (TrustZone) using OP-TEE as a reference implementation.
- OP-TEE fTPM implementation fork is maintained here: [optee_ftpm](#)
- Linux kernel fTPM driver based on TEE bus
 - `drivers/char/tpm/tpm_ftpm_tee.c`

Platform specific use-cases

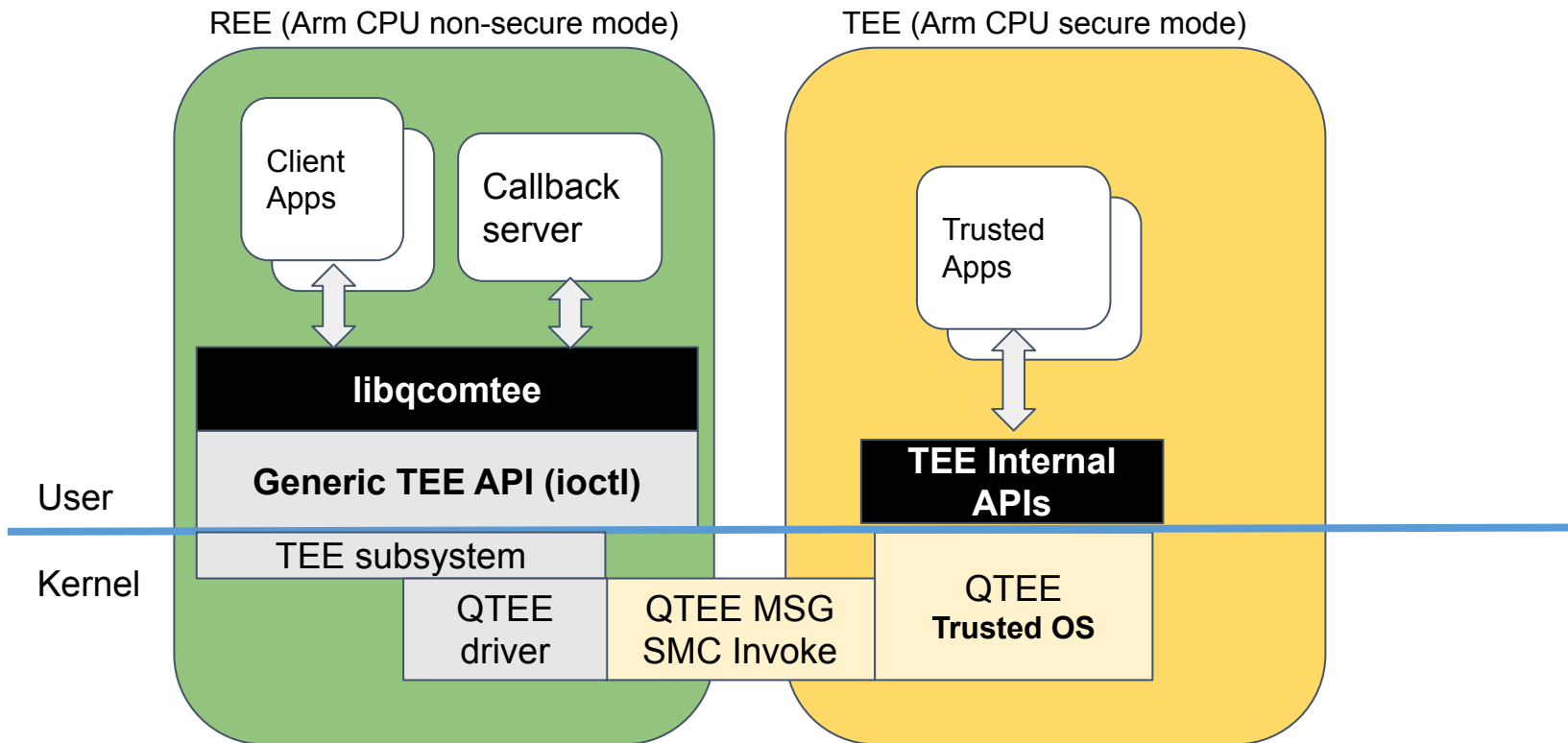
There are numerous kernel drivers registering on TEE bus to access secure platform specific services, like:

- SCMI OP-TEE transport
 - `drivers/firmware/arm_scmi/transport/optee.c`
- EFI runtime variable services
 - `drivers/firmware/efi/stmm/tee_stmm_efi.c`
- Real time clock services
 - `drivers rtc/rtc-optee.c`
- Broadcom firmware service
 - `drivers/firmware/broadcom/tee_bnxt_fw.c`
- Remoteproc tee driver for secure firmware loading
 - Under review [here](#)

Evolution with Trusted Services-TEE



Evolution with Qualcomm TEE (QTEE)



Protected DMAbufs use-case

Secure DMA heaps have been discussed upstream for many years but haven't found it's way into the mainline kernel, the major reasons being:

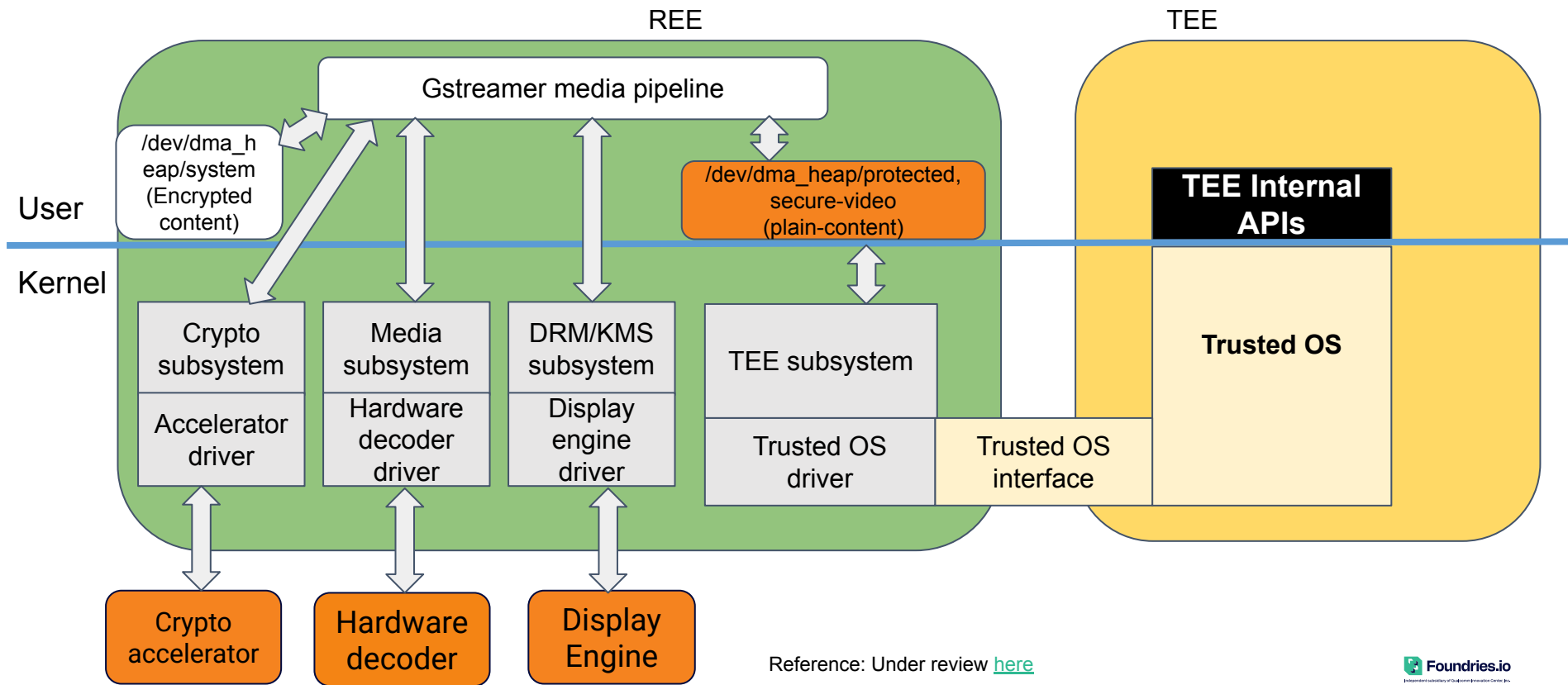
- A missing end-to-end open source use-case
- The difficulty to encode vendor specific metadata leading to vendor specific secure heaps being exposed to user-space

The Digital Rights Management (DRM) secure media pipeline use-case is the major driver here but with binding that the user-space and TEE components not being open.

Kernel TEE subsystem came to the rescue here:

- Abstract out the vendor specific metadata to the backend TEE implementation
- Provide a common infrastructure to enable wider use-cases like protected DMAbufs based key/crypto operations etc.

Protected DMAbufs use-case - Secure media pipeline



Future work...

- Let's strive to keep healthy Linux TEE ecosystem
- Protected DMAbufs use-case
 - Demonstrate end-to-end use-cases for better adoption
- More TEE implementation drivers...



Independent subsidiary of Qualcomm Innovation Center, Inc.

Thank you

Follow us on

