

OP-TEE is ready, lets use it!

OP-TEE use cases and platform readiness

Rouven Czerwinski – r.czerwinski@pengutronix.de



About me

Rouven Czerwinski

Pengutronix e.K.

 Emantor

 rcz@pengutronix.de

OP-TEE

System Integration

Testing



Overview

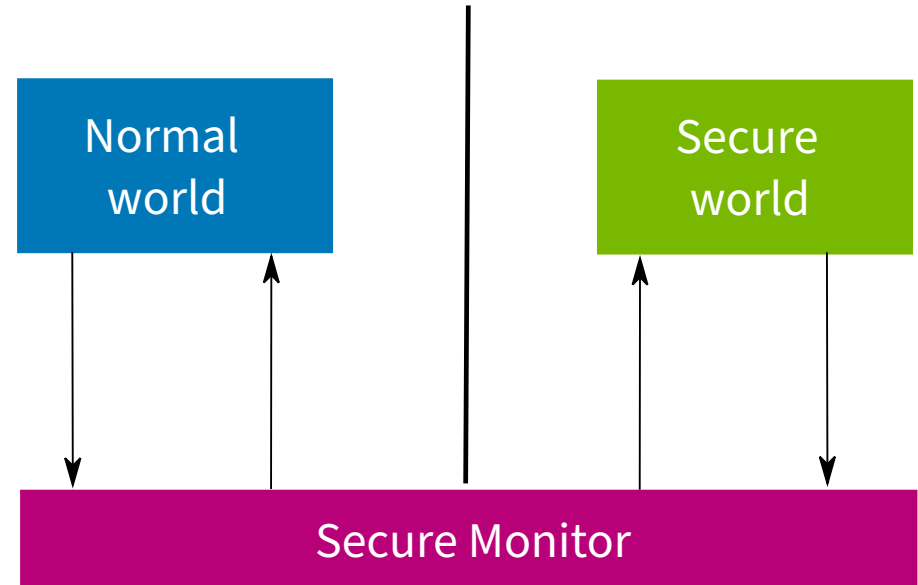
Short Overview:

- TrustZone & OP-TEE
- Solved Use cases:
 - Secret storage: PKCS#11 TA
 - TPM: Microsoft firmware TPM
- Securing OP-TEE



TrustZone (32-bit)

- ARM hardware feature
- Processor switches worlds
 - Normal world, running i.e. Linux (REE)
 - Secure world, running i.e. OP-TEE
- Secure world not accessible from normal world
- Access control for peripherals (serial, SPI, I2C,...) is SoC-specific



OP-TEE

- Open Portable Trusted Execution Environment
- Implementation of the GP TEE specification
- BSD 2/3-clause licensed
- Provides an execution environment:
 - No scheduling
 - Different concurrency models (per Application)

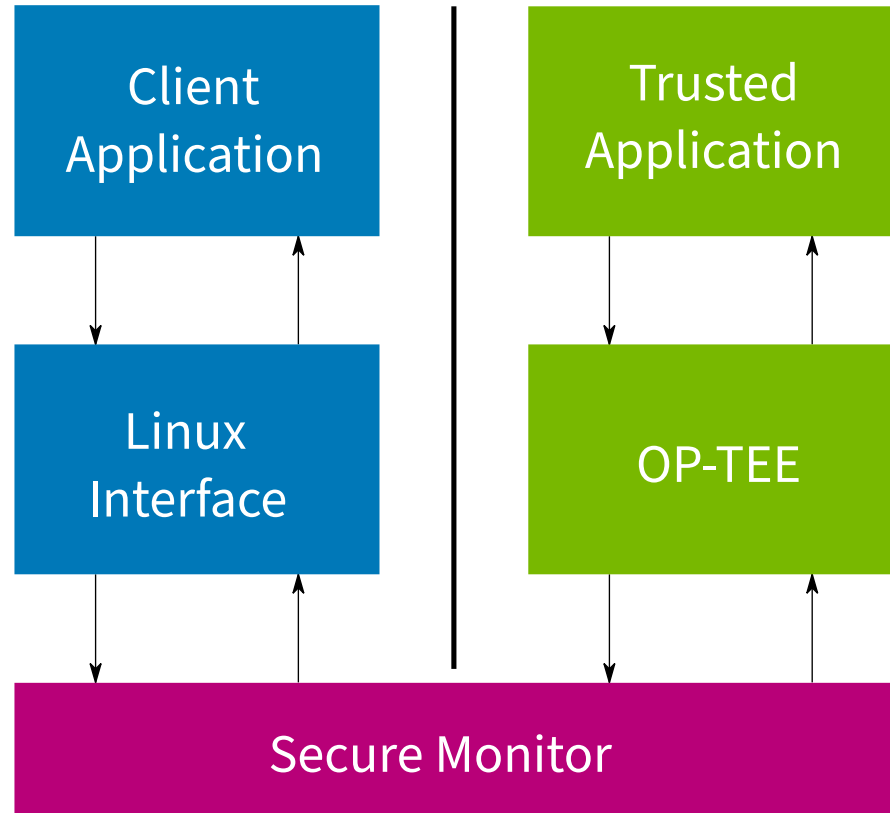


OP-TEE Trusted Applications (TA)

- Idea → use TA for sensitive computations and storage, application within Linux
- Applications split into REE (Linux) and TEE (OP-TEE) part
- TEE part of the application is called Trusted Application (TA), runs within OP-TEE
- Libteec (Tee Client Library) implements interface for TA communication



TA communication flow



OP-TEE features

- Using Replay Protected Memory Blocks (eMMC/NVME feature) for rollback protected storage
- Drivers for common DDR access firewalls (TZC380, TZC400)
- Upstream kernel driver maintained by OP-TEE maintainers
- Platform Support for: i.MX, Layerscape, STM32MP1, qemu, hikey, raspberry pi 3, rockchip and TI AMxx



OP-TEE persistent storage

- OP-TEE → TEE-Supplicant
- OP-TEE encrypts and authenticates data using Unique Key
- GP TEE Persistent storage API for TA
- Tee-Supplicant Storage (no OP-TEE storage driver)
 - Access to eMMC → RPMB (RPMB FS)
 - Store within Linux Filesystem (REE FS)



Use cases

- **TPM** (PCR, Sealing, Attestation)
- PKCS#11 (i.e. Signing, Device Authentication)
- Trusted Keys (Linux Keyring Sealing, under discussion)
- Payment verification?
- Content decryption (DRM)?
- License Management?



PKCS#11 primer

- Standard Programming Interface
 - Hardware Secure Modules (Yubikey, Nitrokey,...)
- i.e. generate Public-Private Keypair, use private key to sign data
- Private key marked unexportable, cannot leave the device
- Supported in:
 - Chromium, nginx, SSH, wpa_supplicant, curl, evolution, Linux module signing, FIT images, RAUC, code signing
 - Usually via OpenSSL/GnuTLS



PKCS#11 Setup

- **PKCS#11 OP-TEE** branch from Etienne Carriere
- PKCS#11 Client branch from Etienne
- Compile with `CFG_PKCS11_TA=y`
- Use resulting `libckteec.so` as PKCS#11 module
- TA is being upstreamed into OP-TEE OS



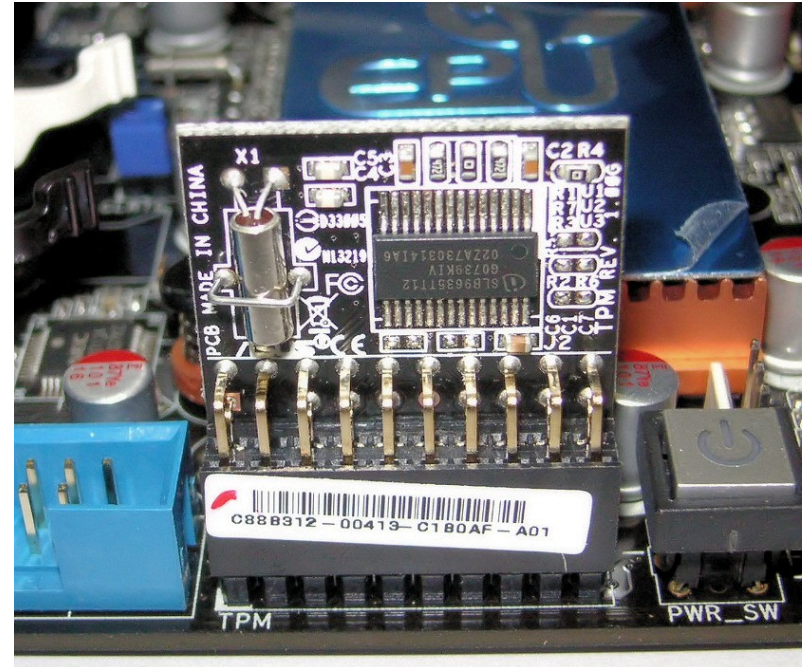
PKCS#11 Demo Time

Demo Time



Trusted Platform Module (TPM) Primer

- Trusted Platform Module (TPM)
- Traditionally component on the board, connected via SPI/I2C
- Nowadays firmware TPM 2.0 on PCs
- Usage: measure boot → unlock trusted keys on correct bootup



TPM module for mainboards



Microsoft Firmware TPM Setup

- kernel driver: `tpm_ftpm_tee`
- TA: `MSRSec fTPM`
- Setup:
 - Enable kernel driver
 - Build `ftpm` with OP-TEE devkit
 - Add device tree node
- `tpm0` device appears



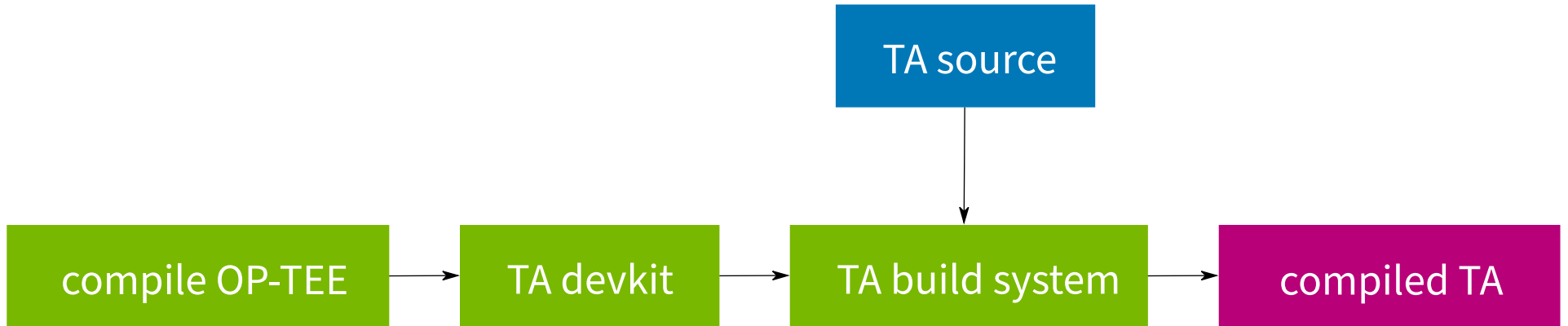
TPM DEMO Time

Demo Time



TA compilation

- OP-TEE development kit
- Compile external TAs



TA Signing

- Public key compiled into OP-TEE
- TA signed after compilation
- OP-TEE verifies TA signature on load
- TA storage locations:
 - Built into OP-TEE
 - Stored in Linux FS



Securing OP-TEE

- Required components:
 - Hardware Random Number Generator (RNG)
 - Unique Key only accessible from secure world (HUK)
 - DDR/SRAM Firewall
 - Device bus access policies
- SoC dependent integration:
 - Secure Boot?
 - Bootloader?



Securing OP-TEE on i.MX6: RNG

- i.MX Cryptographic Accelerator and Assurance Module (CAAM)
- CAAM has an internal Random Number Generator (RNG)
- OP-TEE with runtime CAAM Support:
 - random numbers directly from CAAM
- OP-TEE with CAAM boot-time support:
 - Seed Software RNG from CAAM



Securing OP-TEE on i.MX6: Hardware Unique Key

- Every i.MX6 has a unique one-time programmable key (OTPMK)
- CAAM can provide a hash over this key for verification
- Hash different for normal/secure world
- Only with High Assurance Boot (HAB)!



Securing OP-TEE on i.MX6: DDR Firewall

- i.MX6 has a TrustZone Controller 380 (TZC-380)
- Derive TZC-380 configuration from OP-TEE configuration
- Lock configuration after setup

TZC380 autoconfiguration (search correct region size) and support for i.MX6Q/D
#2913

Edit

Open with ▾

Merged jbech-linaro merged 3 commits into OP-TEE:master from Emantor:topic/tzasc on Apr 15, 2019

Conversation 42

Commits 3

Checks 0

Files changed 7

+137 -0



Emantor commented on Mar 29, 2019 • edited ▾

Contributor 😊 ⋮

Hello,

this PR adds a new function called `tzc380_auto_configure` which searches for the matching configuration for a given start address and size. This configuration is then applied with the given permissions to the controller.

This vastly simplifies configurations for devices with a generic RAM layout, since the necessary information is already available from the header file.

Support for i.MX6QD devices is added as a user of the new function.

Reviewers

jforissier



etienne-lms



Assignees

No one assigned



Labels



Securing OP-TEE on i.MX6: Device Bus Policies

- i.MX6 Central Security Unit (CSU)
- Access policies for DMA masters (GPU, Ethernet, PCIe,...)
- Done for i.MX6UL, others are easy!

Add CSU SA register settings for i.MX6UL #3552

 Merged jforissier merged 2 commits into `OP-TEE:master` from `Emantor:for-upstream/mx6ul_csu`  2 days ago

 Conversation 6

 Commits 2

 Checks 0

 Files changed 3



Emantor commented 3 days ago

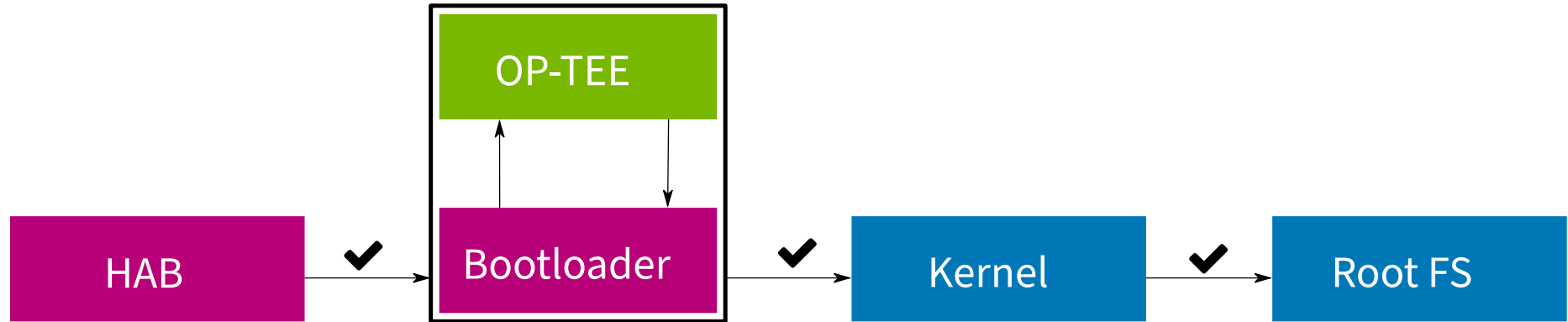
Contributor +  ...

This pr configures the secure access register of the i.MX6 CSU to set almost all non-TrustZone aware masters to non-secure and locks the settings afterwards.



Integration: HAB for Secure Boot

- HAB ROM verifies bootloader, loads OP-TEE
- Verify Kernel via FIT image?
- Verify RootFS with DM Verity?



Integration: Bootloader

- When do you load OP-TEE (for i.MX6)?
 - Before the kernel?
 - Early during Startup of the bootloader?
- SoC dependant:
 - i.MX6 leaves OP-TEE loading to the bootloader
 - STM32MP1 uses TF-A, OP-TEE loading only supported through TF-A
 - i.MX8MQ also uses TF-A



Thanks

- To Jens Wiklander, Jerome Frossier and Etienne Carriere for review, input and maintenance on OP-TEE
- Etienne for the creation of the PKCS#11 TA & Library, Ricardo Salveti for review and additional features
- NXP: Clément Faure, Cedrix Neveux and Silvano Dininno for contribution of the CAAM driver
- To our customers for funding the work



Conclusion

Go and use OP-TEE!