Secure Boot and TPM Attestation for seL4-Based Systems

Thesis C Presentation

Nicholas Berridge-Argent
z5208292

Supervised by Ihor Kuz
Co-Supervised by Ben Leslie
Assessed by Gernot Heiser

Term 3, 2021

Contents

The Thesis

- seL4 is a formally verified microkernel
- Designed for systems with strong reliability and security requirements
- Some security problems are out-of-scope for seL4:
  - Hardware
  - *Userspace* software — although the effects are mitigated
  - Vulnerabilities early in the boot process
- Secure boot and TPM attestation help with the latter
- How can we use these with seL4-based systems?

Secure Boot and Attestation

Secure boot

- Uses cryptographic hashes and signatures
- Each boot stage checks the signature of the next
- System stores a list of public keys it trusts
- Only trusted software boots

Attestation

- Each part of the system is hashed before it starts
- Stored in an isolated part of the system with an asymmetric key
- Measurements are sent with a signature
- Signature can be verified on another trusted computer
- Other computer knows it can trust that computer

TPM Attestation

- Trusted Platform Module
- A piece of hardware or software — supported by firmware
- Provides the isolation needed for attestation
- Has other security features, a useful tool
- Hardware/firmware isolation gives an edge over software
- Depends on the platform

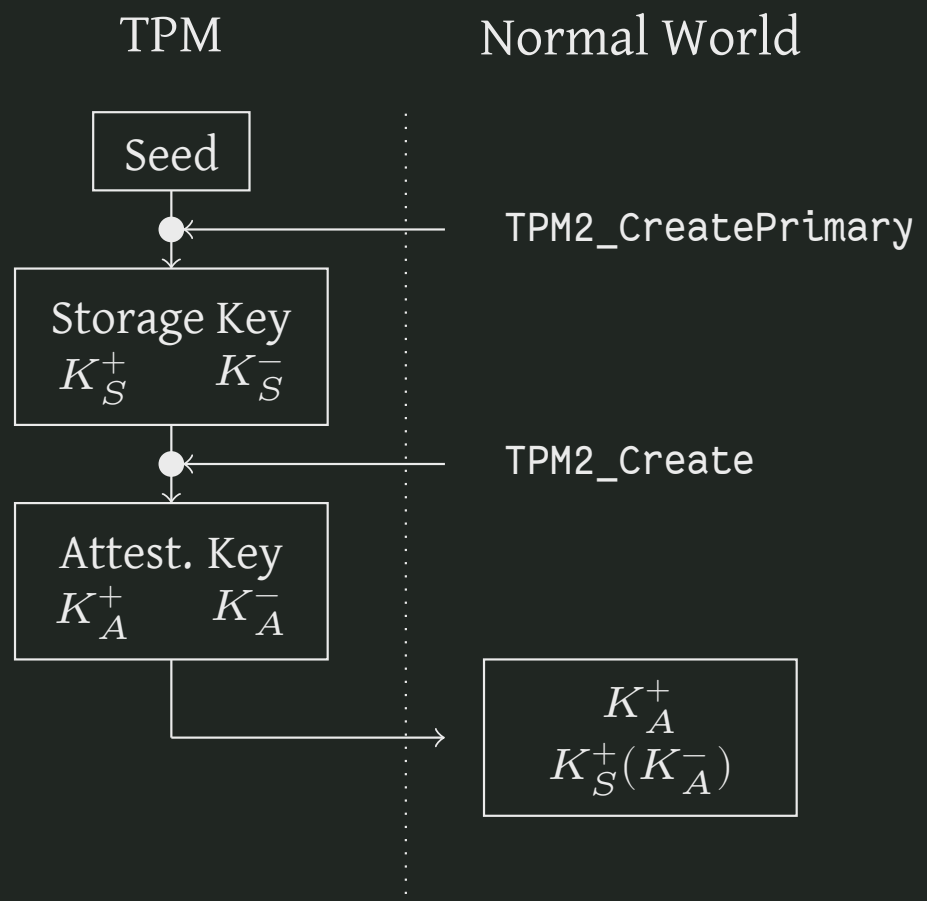
TPM Attestation

- Large computers (desktops, laptops, servers) have dedicated hardware TPMs
- Small computers (embedded, smartphones) cannot justify these
- seL4 mainly targets these smaller computers
- ARM processors support TrustZone — firmware isolation
- With TrustZone, we have fTPM, by Microsoft Research
- All TPMs follow a standard (1.2 or 2.0) — fTPM implements 2.0 exactly
- Requires an ARM processor and secure storage

How Attestation Works

During manufacturing / provisioning:

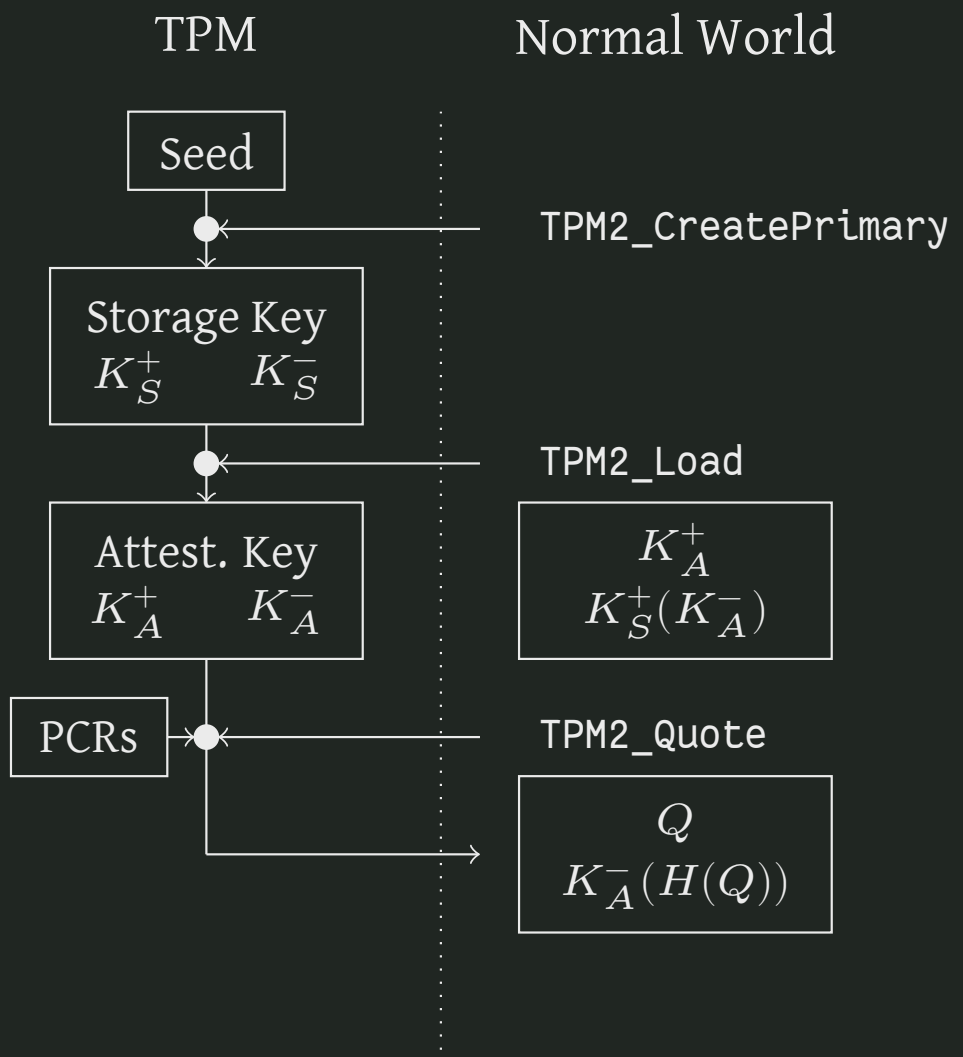
- TPM is provisioned with a 'storage seed' which never changes
- Storage seed generates an asymmetric storage key which never changes
- TPM generates an asymmetric attestation key
- TPM provides the public key of the attestation key, and the private key encrypted by the storage key



How Attestation Works

During boot:

- Storage key has to be regenerated, but has the same keys
- System loads the public and encrypted private key of the attestation key back into TPM
- TPM decrypts the attestation private key using the storage key
- System is measured into the TPM's PCRs (Platform Configuration Registers)
- PCRs are hashed, bundled with a clock value, and signed with the attestation private key — called a 'quote'
- An external verifier uses the attestation public key to check the quote



Getting to Attestation

To get working attestation, we need:

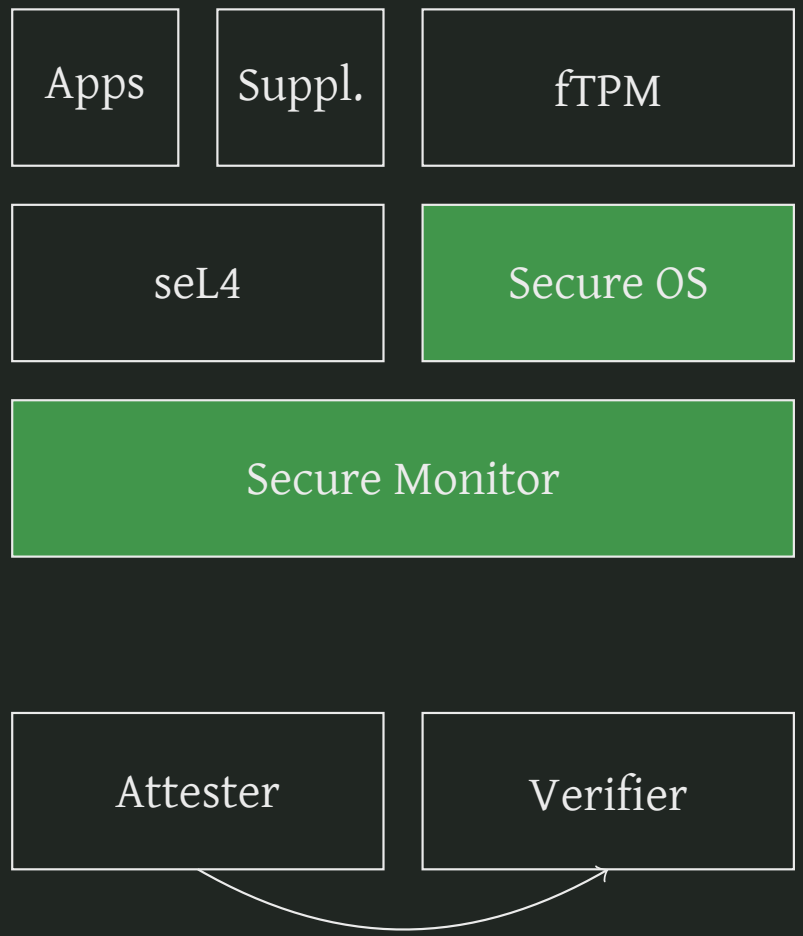
- A suitable platform
- TrustZone, and OP-TEE — a TrustZone operating system
- Secure storage access, through MMC
- fTPM — a ‘trusted application’ running on OP-TEE
- Communication with all of the above from seL4
- Implement the standard attestation process
- Something to verify the attestation

Contents

Platform

- Chosen platform was the i.MX8 M Quad Evaluation Kit
- Based on the i.MX8 M Quad applications processor
- ARMv8, AArch64
- TrustZone is enabled and available
- Supports open source configurations of all the required software
- Has a (somewhat) well documented boot process

Contents



TrustZone

- Firmware feature on modern ARM processors
- Provides normal world and secure world on different execution levels
- Powered by a ‘coarse’ context switch, with the `smc` instruction
- May contain a secure world operating system and secure world services
- Can check for these and communicate with them using `smc`

SMC Instruction

- Follows a standard calling convention
- Needs to be written in assembly
- Needs privileged mode! — pain for microkernels
- In future needs a kernel object
- For now implemented with `seL4_DebugRun` and a trampoline page

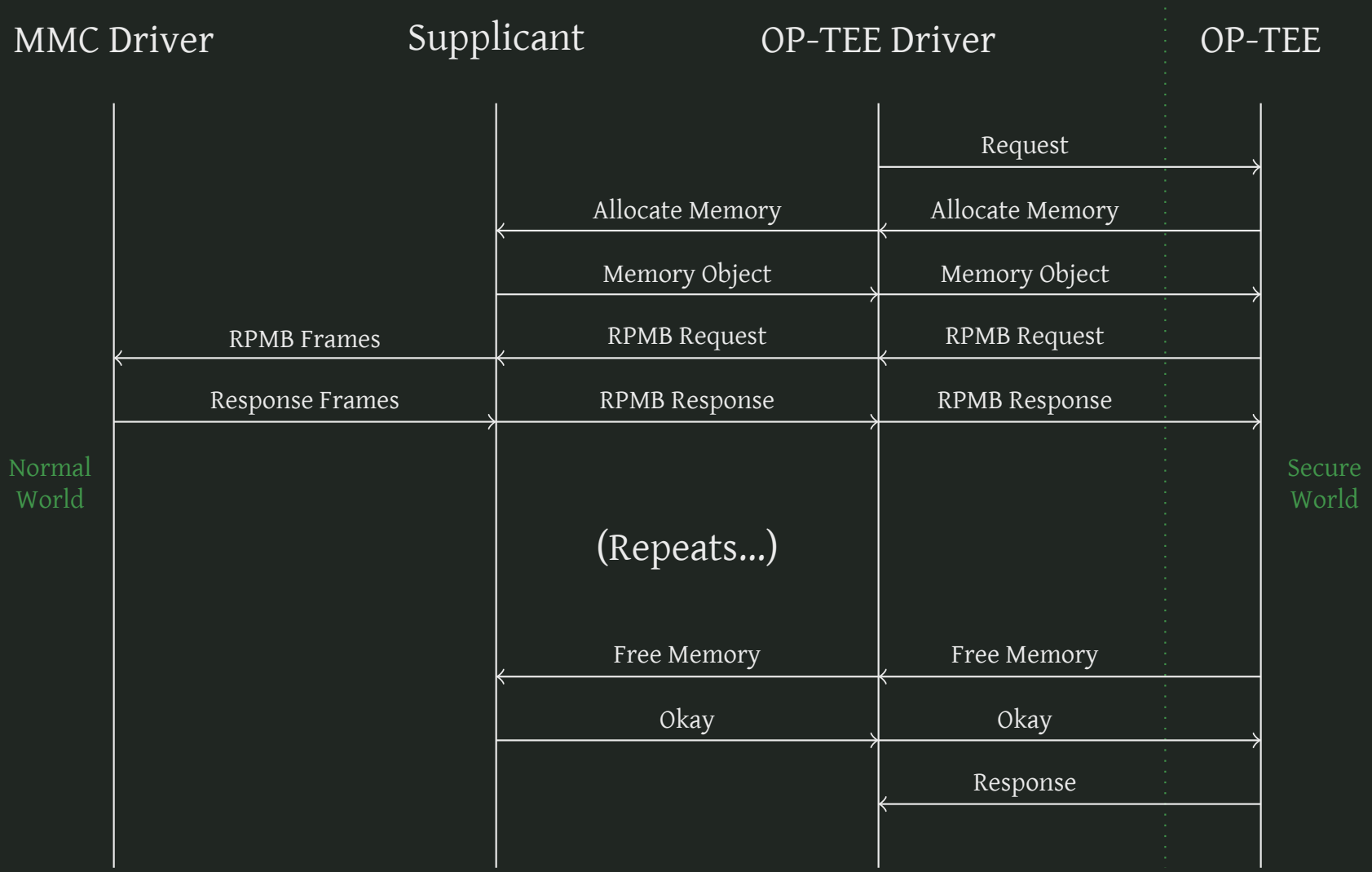
OP-TEE

- Out of the box only have secure monitor, need OP-TEE
- i.MX8 supports open source OP-TEE, with provided Makefile
- Makefile builds secure monitor, OP-TEE and U-Boot, provides microSD image
- i.MX8 configured to boot to microSD with hardware switch
- OP-TEE can be communicated with using `smc`

OP-TEE Supplicant

- OP-TEE is a secure world operating system, designed to have a minimal TCB
- Some functionality is supported in the normal world OS
- Needs a normal world supplicant
- Supplicant handles FS access, secure storage, shared memory
- Most trusted applications use shared memory for commands

OP-TEE Supplicant



Contents

MMC

- Multi-Media Card — standard for embedded storage
- Made up of host controllers and ‘cards’, either embedded or microSD/SD
- i.MX8 has two controllers, one for the embedded card and one for the microSD slot
- Embedded has a RPMB (Replay Protected Memory Block) for secure storage
- Host controllers respond to standard commands
- seL4 ecosystem has a similar driver for i.MX6, U-Boot has a driver for i.MX8
- Neither U-Boot’s driver nor the official documentation worked in seL4, required tweaks

RPMB

- Special partition that responds to a different set of commands
- Commands issued by writing one ‘block’, responses by reading
- Commands are relayed from OP-TEE through the supplicant
- Normal world can access the partition, but cannot access the key
- Key must be programmed once during manufacturing / provisioning

Contents

fTPM

- fTPM is open source, includes Makefile that builds an ELF file
- fTPM is a TA (trusted application) for OP-TEE
- TAs are usually loaded from the file system
- fTPM is needed before a file system is available
- Include fTPM as an ‘early TA’
- Linked inside the OP-TEE binary, loaded at boot time

fTPM

- Communicate with OP-TEE TAs using sessions
- Start a session with a TAs (constant) UUID
- Send commands through shared memory objects, `smc` calls
- These commands wrap TPM 2.0 standard commands

Contents

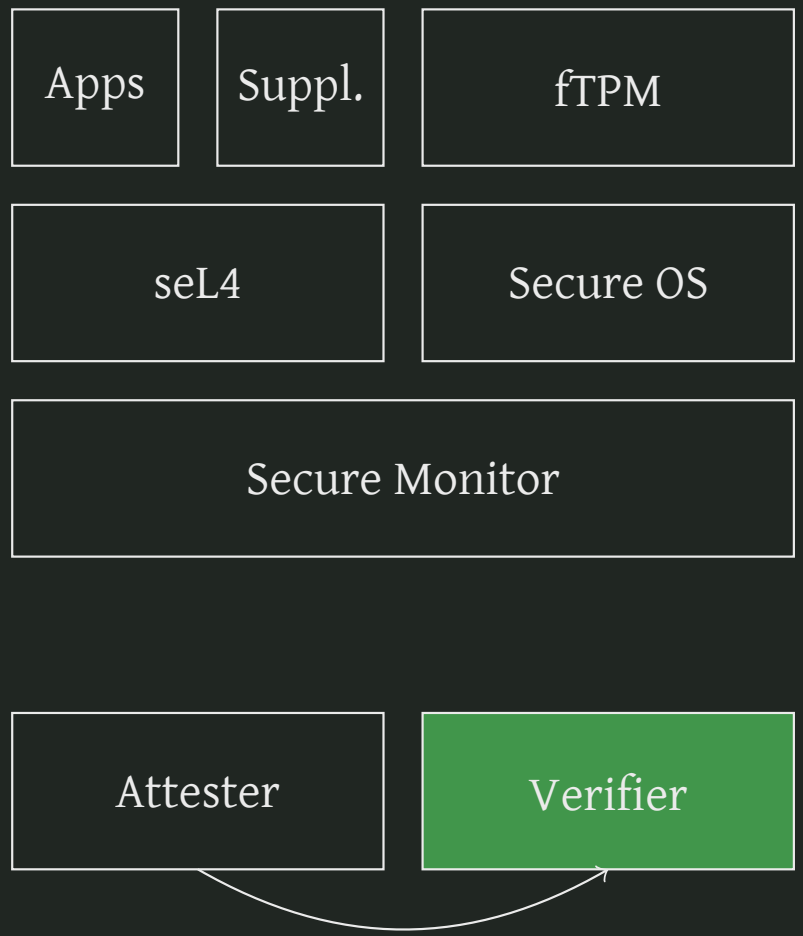
Key Provisioning

- Attestation follows a standard workflow described earlier
- Commands used are standard, we need to send the commands
- To provision the AK, extend U-Boot with new commands:
  - `tpm2 create_srk` to create the storage key
  - `tpm2 create_ak` to create the attestation key
- U-Boot has FS access, can save the attestation key for later

Attestation in seL4

- Attestation key linked into root server image
- Loaded when the root server starts and opens fTPM session
- Attestation through `TPM2_Quote` available at any time

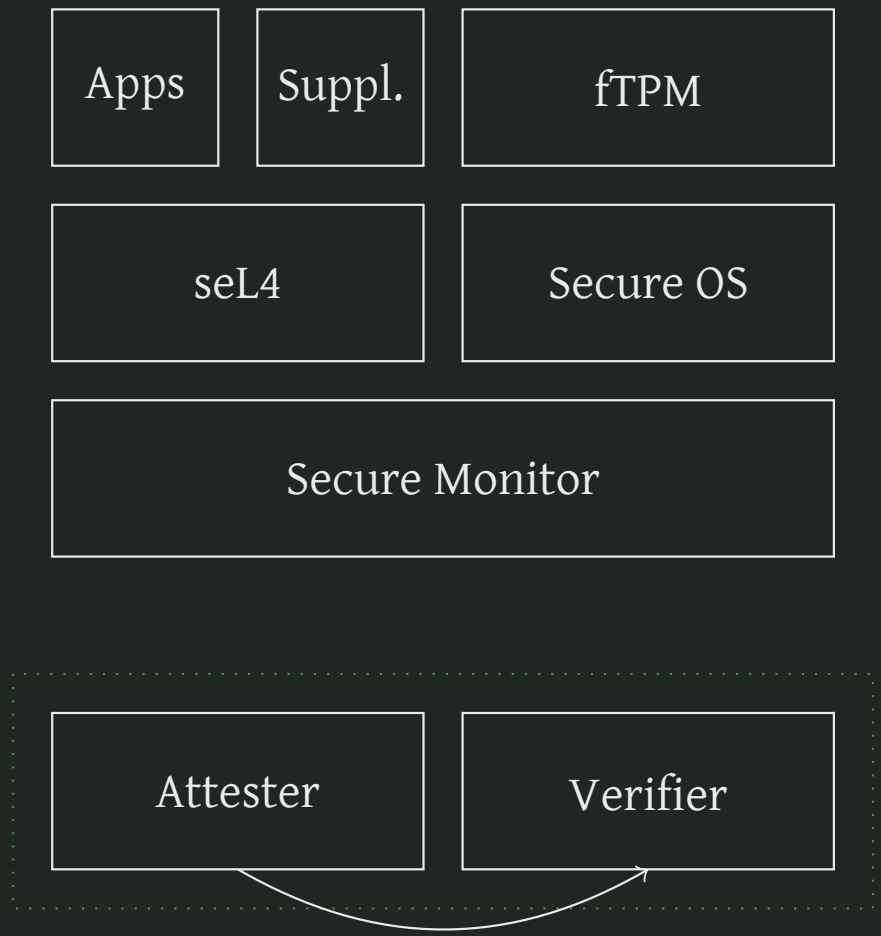
Contents



Verification

- `TPM2_Quote` produces a signature according to pre-configured signature scheme
- Using OpenSSL, verify the signature contents on another, trusted computer
- This is a standard attestation process
- Supported by a web application running on another computer
- Performs the standard verification process, checks PCR digest in quote, signature, clock values
- Maintains a history of clock values to check for replay attacks

Contents



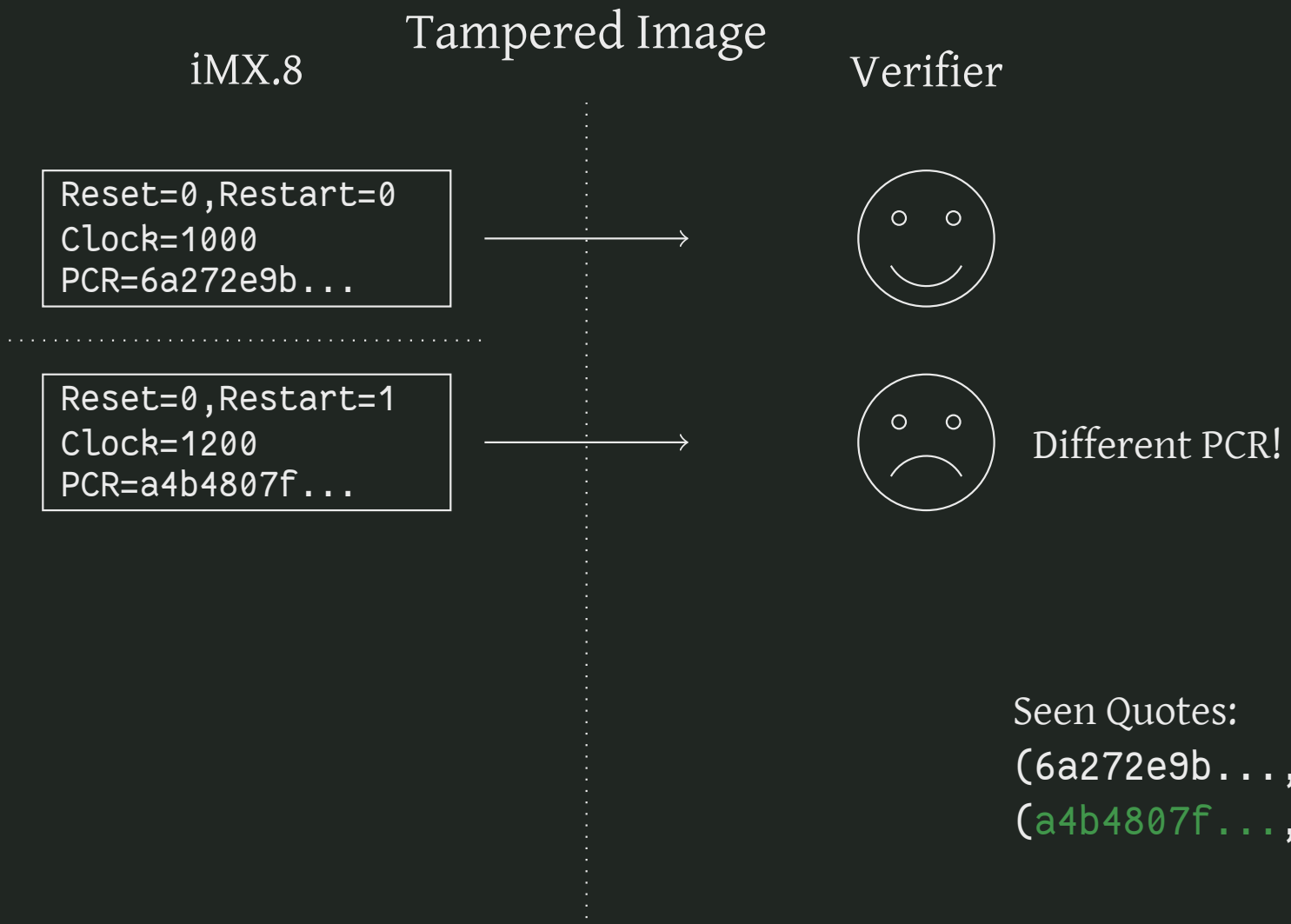
Setup

- Attestation and verficiation implemented as described
- seL4 image loaded from TFTP, measured by U-Boot into PCR 0 before booting
- seL4 root server is my secret diary
- Evil Nick is trying to read my diary
- Evil Nick hacked my TFTP server, but has no access to my i.MX8
- Can replace my seL4 image with ‘eL4’ — a version of seL4 which looks the same but isn’t secure
- Diary produces the PCR value, quote structure, and signature
- I can verify these on my laptop, which Evil Nick hasn’t hacked

Demonstration

Let's see this in action!

Security Analysis



Seen Quotes:

(6a272e9b...,1000,0,0)

(a4b4807f...,1200,0,1)

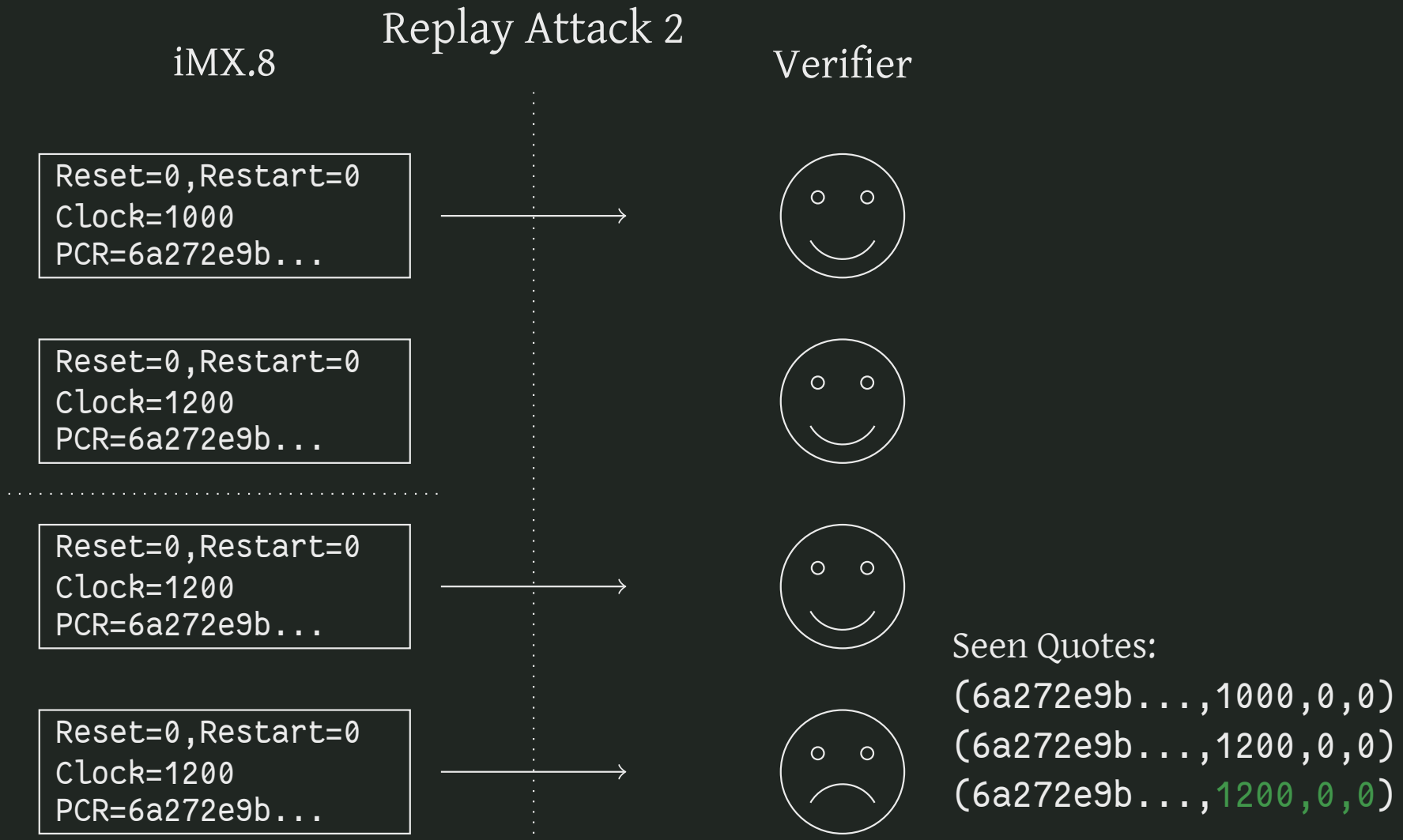
Security Analysis

Seen Quotes:

(6a272e9b...,1000,0,0)

(6a272e9b...,1000,0,0)

Security Analysis



Contents

Analysis & Contributions

- So far: implemented local(-ish) attestation according to protocol
- Ported relevant drivers, added extra U-Boot commands
- Measures and attests the seL4 image
- Implemented all as a (functional) prototype
- Unexpected setbacks, features unimplemented
- No secure boot, attestation provides similar functionality

Near Future: Rest of Thesis C

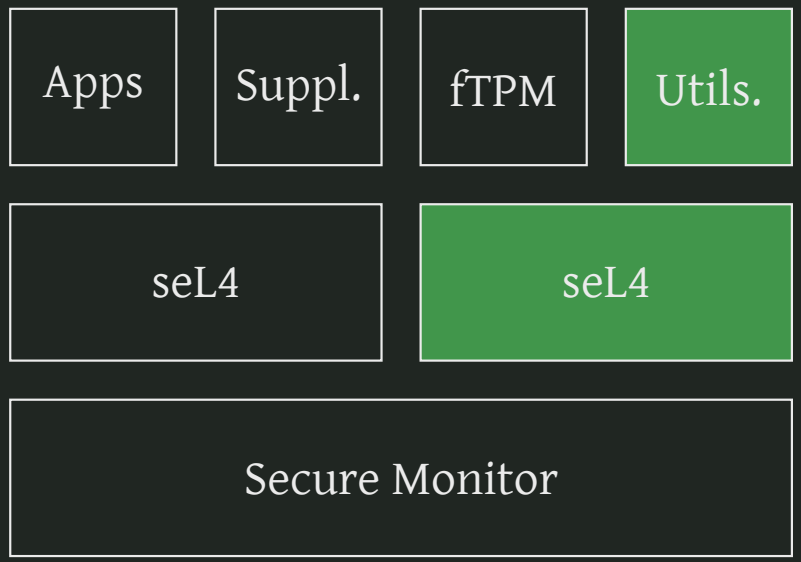
- Clean up code, package as CAmkES module
- Create `smc` kernel object discussed
- Write the thesis report, with some further analysis

Forseeable Future: To Complete

- As it stands, not everything I set out to implement is done
- Remote attestation
- Secure boot
- Measurement of earlier boot stages
- All could be easy, probably no time to investigate

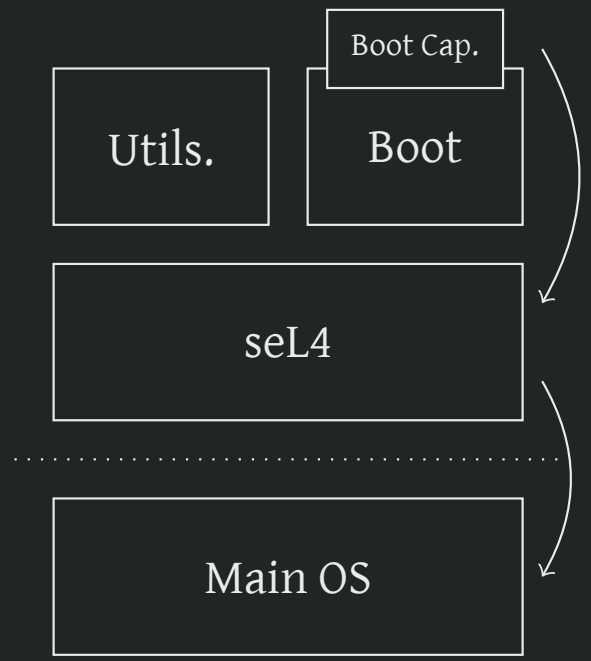
Distant Future: seL4 and fTPM

- OP-TEE and TrustZone designed to provide isolation for fTPM
- TrustZone is at least required because it is available before the normal world on ARM systems
- seL4 is a better choice for the trusted OS (see Nick Spinale's proposal)
- fTPM is currently written for OP-TEE
- In theory, fTPM could run on top of seL4 with some trusted utilities (e.g. RPMB library)



Distant Future: seL4 as a Bootloader

- Advantage of TrustZone and OP-TEE is running early before the bootloader and normal OS
- Bootloaders load OS, handle platform-specific initialisation
- If we trusted the bootloader, we could rely less on TrustZone and OP-TEE
- seL4 may be suitable for use as a bootloader
- Reconfigure the kernel with a 'boot cap', move initialisation utilities to userspace processes



Summary

We have covered:

- Attestation: what and why?
- What's needed for attestation on seL4
- Prototype and demonstration of TPM attestation
- What's left for thesis C and the future

Thanks for Coming!

Enjoy the rest of the thesis presentations