OP-TEE 101

A Gentle Introduction to Trusted Execution and OP-TEE
What Does This Talk Cover?

● **Why Trusted Execution and What is it?**
  ○ Introduces Trusted Execution and GlobalPlatform Standards.
  ○ Provides information on key concepts and architecture.

● **Why OP-TEE and What is it?**
  ○ An Open-source Portable Trusted Execution Environment.

● **How does OP-TEE work?**
  ○ ARM TrustZone and OP-TEE Design.

● **How to get Started with OP-TEE?**

● **Where to go for more information?**
Who am I?

- **Russell Wayman**

- Project Manager, Member Services, Linaro:
  - Project Manager for the ARM, STMicroelectronics and Qualcomm Landing Teams.

Outline

● Trusted Execution and GlobalPlatform TEE Specifications
  ● ARM TrustZone®
  ● OP-TEE
  ● Trusted Boot
  ● Getting Started with OP-TEE
Security for Smart Connected Devices

Smart connected devices:
- Smartphones, tablets, digital TV systems...
- For: Social, business, purchases, media...

Need to protect:
- Bank details and passwords,
- Personal data, photos and video streams,
- Authorised access to mobile networks and online services,
- Paid-for multimedia content in line with digital rights.
Trusted Execution Environment

- An Isolated Execution Environment.
- Runs alongside a “Rich OS” e.g. Android, Ubuntu.
- Provides a higher level of security than the Rich OS.
- Provides trusted services to applications in Rich OS.

A Trusted Execution Environment is intended primarily to secure against software attacks or “shack attacks” (low budget hardware attacks).

Providing security against tampering and physical attack requires a “Secure Element” such as a Smartcard or Crypto IC.
GlobalPlatform and History of TEE

- Created in 1999 to standardize smart card infrastructure.
- A non-profit consortium with 130+ members.
- Develops the standards for secure mobile platforms:
  - Cards, Devices and Systems.
- Publishes TEE Specifications.
- Runs TEE Compliance Program and Certification Scheme.

Key TEE Specifications published:
- TEE Client API Specification v1.0 July 2010,
- TEE Internal API Specification v1.0 Dec 2011,
  - Internal Core API v1.1 June 2014.
Architecture of Platform with TEE

Rich OS Application Environment

Client Applications

Rich OS (REE)

TEE Client API

Trusted Execution Environment

Trusted Applications

TEE Internal APIs

TEE Comm Agent

Trusted Core Framework

Trusted Drivers

Trusted OS Components

Hardware-secured resources

Hardware Platform

Trusted Applications
**Context**: Connection from CA to the Trusted OS.

**Session**: Connection from CA to TA.

**Command**: Unit of Communication from CA to TA.

**Shared Memory**: Shared buffers allocated by Client API or by CA and registered.
TEE Client API: Functions

Client API Functions (**TEEC_xxx**):
InitializeContext,
FinalizeContext

RegisterSharedMemory,
AllocateSharedMemory,
ReleaseSharedMemory

OpenSession, CloseSession

InvokeCommand,
RequestCancellation
TEE Client API: Principles

- Blocking functions.
- Can use from multiple concurrent threads.
- Can request cancellation,
  - From another thread!
- Client-side memory allocation,
  - Pointers not opaque handles.
- Source-level portability but not binary-level,
  - So need to recompile for a different Trusted OS.
Trusted Application APIs required to...

- Communicate with Client Application and other Trusted Applications,
- Store data: in heap and secure file system,
- Get the time from system and real-time clocks,
- Use cryptographic operations,
- Get a PIN or login details from a User,
- Communicate with a Secure Element,
- Connect securely to services over the internet.
Trusted Application and TEE APIs

**Entry Points:**
- Create
- Destroy
- OpenSession
- CloseSession
- InvokeCommand

**Trusted Application:**

**TEE Internal APIs:**

**TEE Core API:**
- Trusted Storage
- Time
- Cryptographic
- Arithmetical
- Property Access
- Cancellation Support
- Memory Management

**Other TEE APIs:**
- Trusted User Interface API
- Secure Element API
- Sockets API
TEE Core API (``TEE_xxx functions``)

- Trusted Storage
- Time
- Cryptographic
- Arithmetical
- Property Access
- Cancellation Support
- Memory Management
TEE Core API (TEE_

**Trusted Storage**
Create/Open/Close/…
TransientObject or PersistentObject

**Time**
GetSystemTime/TAPersistentTime/REETime...

**Cryptographic**
Cryptographic Functions: Message Digest, Symmetric Cipher, MAC, Authenticated Encryption, Asymmetric Encryption and Decryption, Key Derivation, Random Data Generation

**Arithmetical**
Big Integer operations: Conversion, Logical, Basic Arithmetic, Modular Arithmetic...

**Property Access**
GetPropertyAs[String, Bool, u32…] , Enumeration...
(Configuration Properties associated with TA, Client and TEE)

**Cancellation Support**
GetCancellationFlag, MaskCancellation...

**Memory Management**
Malloc, Realloc, Free, MemMove…
Other TEE APIs (TEE_xxx functions)

**Trusted User Interface API**

Create Message Box, PIN Screen, Login Screen

**Secure Element API**

Query the available Readers, Query the Secure Elements, Exchange Data with the SEs

**Sockets API**

Include basic network client functionality in a Trusted Application. Supports: UDP/TCP/IP, TLS
Other TEE APIs (**TEE_xxx functions**)

**Trusted User Interface API**
- TUICheckTextFormat,
- TUIGetScreenInfo,
- TUIInitSession/CloseSession
- TUIDisplayScreen

**Secure Element API**
- SEServiceOpen, GetReaders...
- SEReaderOpenSession...
- SESessionOpenChannel...
- SEChannelTransmit...

**Sockets API**
- open, close, send, recv, error, ioctl

 Include basic network client functionality in a Trusted Application.
 Supports: UDP/TCP/IP, TLS
GlobalPlatform: What’s Next?

GlobalPlatform® specifications under development:

- TEE Administration Framework,
  - Online and offline methods for installation, update and removal of security domains and Trusted Applications,
- Trusted UI API v1.1,
- TEE Fingerprint Biometry API.
Outline

- Trusted Execution and GlobalPlatform TEE Specifications
- ARM TrustZone®
- OP-TEE
- Trusted Boot
- Getting Started with OP-TEE
ARM Four-Compartment Security Model

User mode/System mode.
Running processes are isolated from each other by the OS and the MMU.

Allows multiple virtual OSs.
Each VM isolated from others. Bus masters can be virtualized through System MMU.

TrustZone® extensions.
System can be physically partitioned into Normal world and Secure world.

Physically separate ICs.
Offering tamper-proof secured processing and storage.
ARM TrustZone®

Partitions SoC into Normal World and Secure World

Non-Secure (NS) bit (bit 0) in the Secure Configuration Register in CP15
ARM TrustZone®

- Partitions SoC into **Normal World** and **Secure World**.
- CPUs can timeslice between worlds.
- Hardware logic in the bus fabric enforces separation.
- Security-aware debug infrastructure.

- **2004**: Introduced in the ARMv6Z architecture.
- **2007**: Part of the ARMv7 architecture specification.
- **2013**: ARMv8-A architecture introduced changes e.g EL3.
- **2015**: ARMv8-M extends TrustZone to the Cortex-M.
ARM TrustZone®: CPU Architecture

- Extra processor execution level:
  - EL3 (ARMv8 - Monitor mode in ARMv7).

- SMC instruction causes entry to EL3:
  - Secure Monitor Call,
  - Vectors execution to a defined location.

- Monitor Software runs at EL3:
  - Manages switch between worlds, setting/clearing SCR bit 0.
ARM TrustZone®: Memory System

- MMUs provide 2 virtual address spaces:
  - Code in Secure World can access either address space.

- TLB entries tagged to identify world that entered the data.

- Caches hold data from both states:
  - Tag bit to record the world of the transaction.
ARM TrustZone®: System-on-Chip

- Address space partitioning into Secure/Non-secure regions:
  - TrustZone Memory Adaptor for on-chip SRAM,
  - TrustZone Address Space Controller for off-chip RAM.

- Secure peripherals:
  - TrustZone Protection Controller supports run-time switching of security state for peripherals on APB bus.
ARM TrustZone® and Trusted OS

- Provides hardware foundation for Trusted Execution.
- No guarantee: Trusted OS software must be written correctly!
- Could theoretically run full feature-rich OS in Secure World.
- In practice, Secure world software much simpler/smaller:
  - **Simple**: Reduced complexity means less chance of errors,
  - **Small**: Keep key Secure-world resources in on-chip RAM.
- Trusted OS can separate Trusted Applications using MMU.
- Peripherals can be switched into Secure World
  - e.g. for input of PIN or Login details.
Outline

- Trusted Execution and GlobalPlatform TEE Specifications
- ARM TrustZone®
- OP-TEE
- Trusted Boot
- Getting Started with OP-TEE
OP-TEE: Why an Open Source TEE?

- Provides a shared basis for product TEE developments.
  - Collaboration and consolidation not re-invention/fragmentation.
  - OP-TEE has BSD 2-clause license (GPLv2 for Linux driver, test suite).
- Provides a full example for research and education.
  - Historically hard to learn about Trusted Environments.
- Can be included in reference platform deliveries.
- More eyes on security-critical code!

There are commercial Trusted OS products e.g. from Trustonic and Sierraware, why an Open Source one?
History of OP-TEE

- 2008: Ericsson and NXP
- 2009: ST Life-Augmented
- 2010: OP-TEE
- 2013: GlobalPlatform certified
- 2014: Linaro becomes owner of OP-TEE
- 2015: GitHub
OP-TEE: What’s Implemented?

Client API v1.0 ✔

Internal Core API v1.1 ✔
- Internal Client API
- Property Access
- Cancellation Support
- Memory Management
- Trusted Storage
- Time
- Cryptographic
- Arithmetical

Trusted UI API v1.1 ✗ Not Yet

Secure Element API v1.0 ✔

Sockets API v1.0 ✗ Not Yet
OP-TEE: What’s Implemented?

Client API v1.0

Internal Core API v1.1
- Internal Client API
- Property Access
- Cancellation Support
- Memory Management
- Trusted Storage
- Time
- Cryptographic
- Arithmetical

Trusted UI API v1.1

Secure Element API v1.0

Sockets API v1.0

Targets Supported:

ARMv7-A
- Allwinner A80
- ST's Cannes board (b2120 / b2020)
- Texas Instruments DRA746
- QEMU
- Freescale: FSL i.MX6 UltraLite EVK Board
- Freescale: FSL Is1021a

ARMv8-A
- 96Boards HiKey (HiSilicon Kirin 620)
- ARM Juno board
- MediaTek MT8173 EVB Board
- FVP, Foundation and Fast Models

Not Yet
OP-TEE: Architecture

Normal World

Client App

tee-suppliant

Generic TEE API (ioctl)

TEE Client API

 tee-suppliant

Secure World

Dynamic Trusted App

TEE Internal APIs

OP-TEE Trusted OS

Static Trusted App

User

Kernel

TEE subsystem

OP-TEE driver

OP-TEE Msg

SMC call

OP-TEE: Architecture

Client

App

Dynamic

Trusted

App

Generic TEE API (ioctl)

TEE Client

API

Normal World

Secure World

Dynamic

Trusted

App

TEE Internal

APIs

OP-TEE

Trusted OS

Static

Trusted

App

User

Kernel

TEE subsystem

OP-TEE driver

OP-TEE Msg

SMC call

OP-TEE: Architecture

Client

App

Dynamic

Trusted

App

Generic TEE API (ioctl)

TEE Client

API

Normal World

Secure World

Dynamic

Trusted

App

TEE Internal

APIs

OP-TEE

Trusted OS

Static

Trusted

App

User

Kernel

TEE subsystem

OP-TEE driver

OP-TEE Msg

SMC call

OP-TEE: Architecture

Client

App

Dynamic

Trusted

App

Generic TEE API (ioctl)

TEE Client

API

Normal World

Secure World

Dynamic

Trusted

App

TEE Internal

APIs

OP-TEE

Trusted OS

Static

Trusted

App

User

Kernel

TEE subsystem

OP-TEE driver

OP-TEE Msg

SMC call

OP-TEE: Architecture

Client

App

Dynamic

Trusted

App

Generic TEE API (ioctl)

TEE Client

API

Normal World

Secure World

Dynamic

Trusted

App

TEE Internal

APIs

OP-TEE

Trusted OS

Static

Trusted

App

User

Kernel

TEE subsystem

OP-TEE driver

OP-TEE Msg

SMC call
OP-TEE: Linux Kernel Subsystem

- **TEE subsystem:**
  - Manages Shared Memory,
  - Provides generic API as ioctl.

- **tee-suppliant:**
  - Helper process for TEE.

- **OP-TEE driver:**
  - Forwards command from the Clients to OP-TEE,
  - Manages RPC requests from OP-TEE to the supplicant.

For more details, please read [optee_design.md](https://github.com) at GitHub.
OP-TEE: Managing Trusted Apps

- Trusted Apps:
  - **Static**: run in kernel mode,
  - **Dynamic**: run in user mode:
    - Stored in File System,
    - Loaded by OP-TEE
      - Using tee_supplicant.

- MMU L1 Translation tables:
  - Large: 4GiB for OP-TEE kernel,
  - Small tables: 32MiB:
    - TA Virtual Memory,
    - One per thread.
OP-TEE: Communication & Scheduling

- Entry into secure world from SMC (or FIQ arriving).

- Command arriving → Allocated to thread (if available), TA context set up and called;
  - If supplicant needed, RPC is started and thread suspended.

- Return to normal world on task completion, RPC (or IRQ arriving).
OP-TEE: Trusted Storage

- Implements GP Trusted Storage `PersistentObject` functions.

- TEE File System encrypts data:
  - Secure Storage Key (per device),
  - File Encryption Key (per file).

- Data stored in Linux File System
  - Managed by `tee_supplicant`.

For more details, please read `secure_storage.md` at GitHub.
OP-TEE: Paging

- OP-TEE uses ~256KiB RAM for kernel memory:
  - SRAM is normally somewhere between 64-256KiB.

- Running in TrustZone protected DRAM is one option,
  - But may prefer SRAM for security and performance reasons.

- OP-TEE’s pager supports on-demand paging of read-only parts using virtual memory:
  - DRAM is used as backing store,
  - Each page integrity-protected using SHA-256 hash.

For more details, please read [optee_design.md](https://github.com) at GitHub.
OP-TEE: Cryptographic Libraries

- Default cryptographic software library in OP-TEE is LibTomCrypt.

- TEE Cryptographic Operations API makes it easier to add support for other cryptographic engines (both software and hardware).
  - The system calls use a struct `crypto_ops` which assigns function pointers to the actual implementations.

- Some algorithms may be disabled at compile time if they are not needed, in order to reduce the size of the OP-TEE image and reduce its memory usage.

For more details, please read `crypto.md` at GitHub.
OP-TEE: What’s Next?

- **Ongoing:**
  - Upstream OP-TEE’s Linux kernel driver,
  - Enhance Secure Storage:
    - Block-based, roll-back handling, address performance issues,
  - Replay Protected Memory Block (RPMB):
    - Tiny (~128KiB) block in eMMC for data storage,
  - GP Test Suite v1.1 Support

- **Planned:**
  - GlobalPlatform Trusted UI v1.1,
  - Paged Trusted Application.
Outline

- Trusted Execution and GlobalPlatform TEE Specifications
- ARM TrustZone®
- OP-TEE
- Trusted Boot
- Getting Started with OP-TEE
Trusted Boot

Starts with trusted bootloader in ROM.

Each component must authenticate the next stage.

Public Key for the root of trust programmed into OTP on-chip memory.

Hardware Platform

System Running

Normal World OS

Normal World Bootloader

Secure OS Boot

Flash Bootloader

ROM SoC Bootloader

Reset
ARM Trusted Firmware

- An open-source reference implementation of firmware for ARMv8 Platforms, providing trusted boot and runtime services.

- Provided as part of Linaro ARM Platforms releases for Juno and ARM FVP (Fixed Virtual Platform) models.

- OP-TEE is compatible with ARM Trusted Firmware:
  - OP-TEE dispatcher officially merged in ARM-TF,
  - Currently a fork due to lack of support for parsing OP-TEE header in ARM-TF.

- OP-TEE team works with ARM to ensure everything is working!
  - IRQ / FIQ model for OP-TEE and ARM-TF was designed together.
Outline

- Trusted Execution and GlobalPlatform TEE Specifications
- ARM TrustZone®
- OP-TEE
- Trusted Boot
- Getting Started with OP-TEE
- Further Information
Getting Started with OP-TEE and QEMU

$ sudo apt-get install android-tools-fastboot autoconf bison cscope curl flex gdisk libc6:i386 libfdt-dev libglib2.0-dev libpixman-1-dev libstdc++6:i386 libz1:i386 netcat python-crypto python-serial uuid-dev xz-utils zlib1g-dev

$ sudo apt-get install ccache

$ mkdir -p $HOME/devel/optee
$ cd $HOME/devel/optee
$ repo init -u https://github.com/OP-TEE/manifest.git -m default.xml
$ repo sync

$ cd $HOME/devel/optee/build
$ make toolchains
$ make all run

For more details, please read README at GitHub
Getting Started: Running OP-TEE

**Type:**
```
c
```
on **QEMU** command line
modprobe optee_armtz
tee-suppliant&
on serial console command line
Type: `xtest` on serial console command line
OP-TEE Test suite (xtest) is running!
Further Information

- GlobalPlatform TEE Specifications
- ARM Trusted Base System Architecture
- OP-TEE
- ARM Trusted Firmware
GlobalPlatform TEE Specifications

1. **TEE Systems Architecture v1.0** – explains the hardware and software architectures behind the TEE. December 2011.
2. **TEE Client API Specification v1.0** – enables communication between applications running in a Rich OS and trusted applications residing in the TEE. July 2010.
3. **TEE Internal Core API Specification v1.1** – enables trusted applications within a TEE to perform the general operations of a security application, such as cryptography, secure storage, communication and general tasks, such as timekeeping and memory management. June 2014. [1.0 in December 2011].
5. **TEE Sockets API Specification v1.0** – specifies standards to enable trusted applications to directly make use of internet protocol interfaces, rather than send packets to a client application for internet transfer. June 2015.
6. **Trusted User Interface API Specification v1.0** – allows a trusted application to securely display text and graphics, and ask the user to perform an action ranging from navigation to entry of an associated PIN- or Password-backed ID. June 2013.

ARM Trusted Base System Architecture

- ARM Document RM DEN0007
  - Available under NDA from ARM
- Describes the requirements for a Trusted ARM SoC:
  - To meet OMTP and Global Platform “Boundary 1” specifications
  - Covers: CPU, Caches, Interrupt Control, Debug, On-chip Boot ROM, Trusted RAM, Interconnect, Non-volatile memory, Watchdog timers, Clock source, Random number generator entropy source, User input, Power management, Boot Process

CPUs must be at least Level 1 Standard Configuration, including TrustZone extensions…

Cache accesses from one world must not be able to read, write or modify a line from a different world…

The interrupt controller must support at least eight non-secure Software Generated Interrupts and eight secure SGIs…

Trusted Base System must always boot from On-chip Boot ROM…

Trusted RAM area for use by the Trusted OS and Applications, minimum size…
OP-TEE on GitHub

- In GitHub: See https://github.com/OP-TEE

- README at https://github.com/OP-TEE/optee_os/blob/master/README.md

- Documentation at: https://github.com/OP-TEE/optee_os/tree/master/documentation

- Mail the team at: mailto:op-tee@linaro.org
Presentations on OP-TEE (1/2)

LCU14-103: How to create and run Trusted Applications on OP-TEE
https://www.youtube.com/watch?v=6fmwhqrOmpc (Video)
http://www.slideshare.net/linaroorg/lcu14103-how-to-create-and-run-trusted-applications-on-optee?related=4 (Slides)

LCU14:107: OP-TEE on ARMv8
http://www.slideshare.net/linaroorg/lcu14-107-optee-on-armv8?related=5 (Slides)

LCU14-302: OP-TEE Porting and Future Enhancements
https://www.youtube.com/watch?v=QgaGJow7hws (Video)
http://www.slideshare.net/linaroorg/lcu14-302-how-to-port-optee-to-another-platform?related=3 (Slides Porting)

HKG15-303: Secure Playback Using OP-TEE
https://www.youtube.com/watch?v=WJS5ygNGaQ8 (Video)

HKG15-307: OP-TEE paging
https://www.youtube.com/watch?v=hCYjBPxEbY (Video)
http://www.slideshare.net/linaroorg/hkg15307-optee-paging (Slides)
Presentations on OP-TEE (2/2)

HKG15-311: OP-TEE for Beginners and Porting Review
https://www.youtube.com/watch?v=Fksx4-bpHRY (Video)
http://www.slideshare.net/linaroorg/hkg15311-optee-for-beginners-and-porting-review (Slides)

HKG15-505: Power Management Interactions with OP-TEE and Trusted Firmware
https://www.youtube.com/watch?v=hQ2ITjHZY4s (Video)
http://www.slideshare.net/linaroorg/hkg15-505-power-management-interactions-with-optee-repaired (Slides)

SFO15-200: Linux kernel generic TEE driver
https://www.youtube.com/watch?v=BhLndLUQamM (Video)
http://www.slideshare.net/linaroorg/sfo15200-linux-kernel-generic-tee-driver (Slides)

SFO15-205: OP-TEE Content with Microsoft PlayReady on ARM TrustZone
https://www.youtube.com/watch?v=defbtpsw6h8 (Video)
http://www.slideshare.net/linaroorg/sfo15205-optee-content-decryption-with-microsoft-playready-on-arm-53111683 (Slides)

SFO15-503 Secure Storage in OP-TEE
https://www.youtube.com/watch?v=pChEdObYLRM (Video)
http://www.slideshare.net/linaroorg/sfo15503-secure-storage-in-optee?related=1 (Slides)
ARM Trusted Firmware

User Guide on GitHub: https://github.com/ARM-software/arm-trusted-firmware/blob/master/docs/user-guide.md

LCU13 An Introduction to ARM Trusted Firmware
https://www.youtube.com/watch?v=q32BEMMxmfw (Video)
http://www.slideshare.net/linaroorg/arm-trusted-firmwareforarmv8alcu13 (Slides)

LCU13: ARM Trusted Firmware Deep Dive
http://www.slideshare.net/linaroorg/trusted-firmware-deepdivev10 (Slides)

LCU14-500: ARM Trusted Firmware Roadmap and Progress
https://www.youtube.com/watch?v=je0_yYgKdc (Video)
http://www.slideshare.net/linaroorg/lcu14-500-arm-trusted-firmware (Slides)

LCA14-102: Adopting ARM Trusted Firmware
https://www.youtube.com/watch?v=h98jBQrxKg (Video)
http://www.slideshare.net/linaroorg/lca14-102-adoptingarmtrustedfirmware (Slides)

HKG15-502: ARM Trusted Firmware Evolution
https://www.youtube.com/watch?v=l2Mv-R9IMPs (Video)
http://www.slideshare.net/linaroorg/hkg15502-arm-trusted-firmware-evolution (Slides)

SFO15-101: Security requirements on ARM v8-A boot architecture
https://www.youtube.com/watch?v=q5nF9tSrak4 (Video)
http://www.slideshare.net/linaroorg/sfo15101-security-requirements-on-armv8a-boot-architecture (Slides)