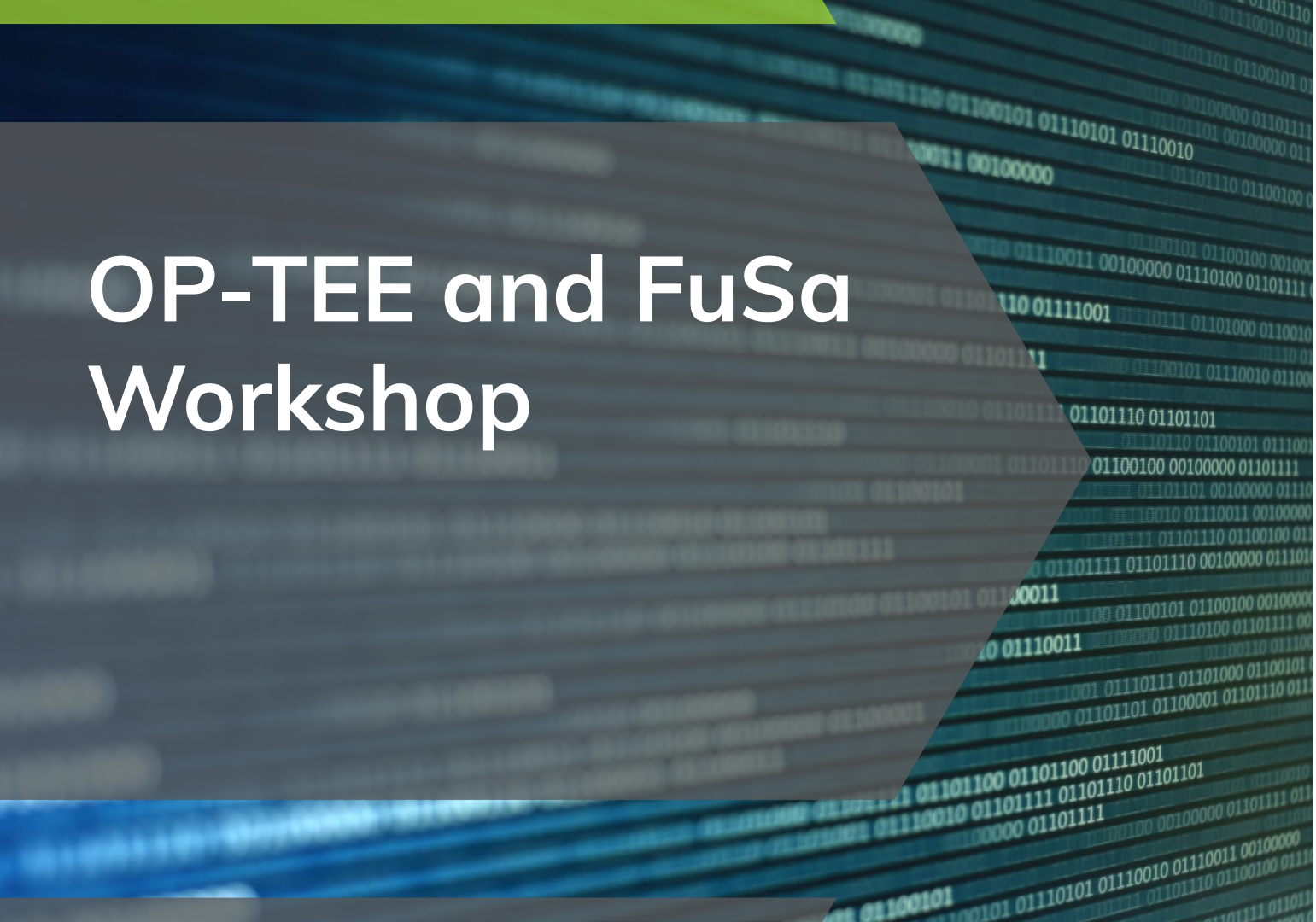
OP-TEE and FuSa Workshop



Agenda

- Vision
- Use cases discussion
- What is the right scope?
- State of the research
- “Who do what” discussion (Development, LTS, archiving, assessment)
- Next steps
- Open topics for discussion

Vision

TrustZone is a key asset to build value for trust and safety.

Adding discrete processors to deal with trust and safety can be expensive in space, lower performance and in some cases lower security.

Safety ready OP-TEE is one aspect of a broader effort to make TEEs a first class citizen of the computing world.

Why consider safety standards for OP-TEE ?

We see usage of OP-TEE in safety-critical contexts

- Automotive, robotics, aeronautics, drones...

Cortex R-82 and Cortex AE class processors opens new doors to safety environments that can also run Linux and thus the firmware need to be ready for those solutions

(Trusted Substrate is functional on a 32 bit environment that does not have EL3 and hence can be adapted to very diverse architectural contexts)

Use Cases

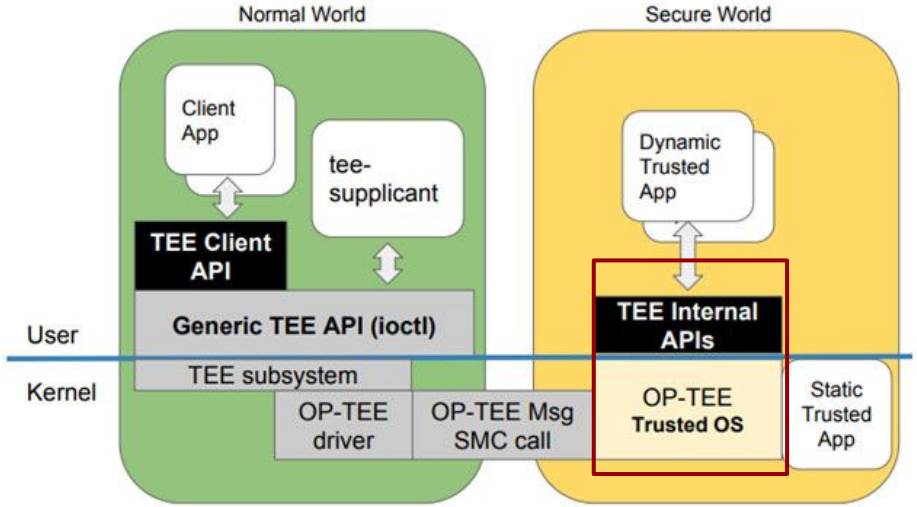
Use-Cases

- Firmware role when booting and operating safety certified OS or hypervisor
- Safety certified TA (a core is dedicated to this and stays in secure state)
- Safety certified VMM with Hafnium as hypervisor
  - VMM role and FF-A backend
  - Secure device sharing amongst safety certified Secure Partitions
- RAS handling in OP-TEE in Secure SRAM

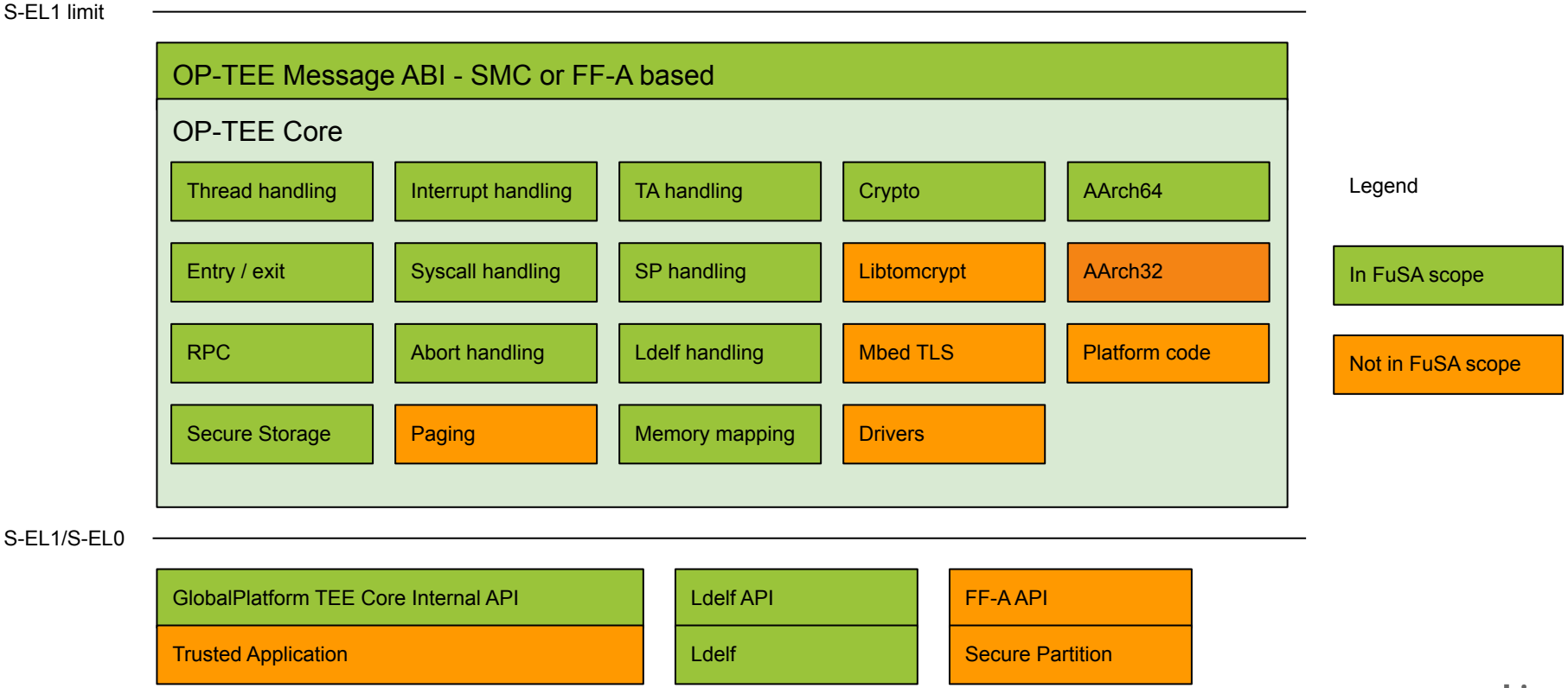
Scope

What is the right scope ?

- To start with focus on only secure world component of OP-TEE
- Any external dependencies should be captured in assumptions.
  - Keep non-secure world out of picture
  - Keep secure firmware out of picture



What is the right scope ?



State of Research

Linaro

What is Functional Safety ?

- Functional Safety is absence of unreasonable risk due to hazards (potential source of harm) caused by malfunctioning behavior of the complex electronic systems. [ISO 26262]
- Functional safety seeks to reduce the level of risk in a device or system.
- Ability of the system to react on potentially dangerous condition by using safety function and reduce the risk.
- Examples include the deactivation of a medical infusion pump should it malfunction or the automatic activation of an overflow valve when a certain liquid or pressure level has been reached.

Standards

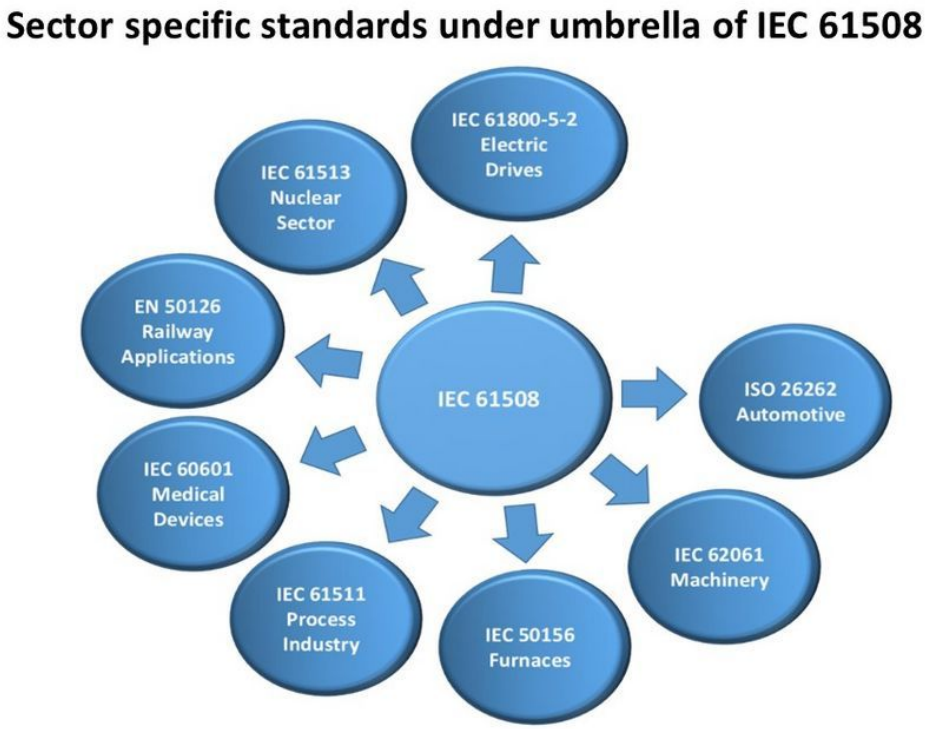


Image - https://www.tuv-nord.com/in/en/services/product-certification-ce-atex-sil-risk-assessment/functional-safety/

Linaro

More on standards

IEC 61508 is a basic functional safety standard applicable to all industries. It defines functional safety as: “part of the overall safety relating to the EUC (Equipment Under Control) and the EUC control system which depends on the correct functioning of the E/E/PE safety-related systems, other technology safety-related systems and external risk reduction facilities.” **The fundamental concept is that any safety-related system must work correctly or fail in a predictable (safe) way.**

ISO 26262 is an adaptation of IEC 61508 for Automotive Electric/Electronic Systems. It is being widely adopted by the major car manufacturers.

Based on our understanding so far, we would target IEC61508 which is the parent of all standards. There should be 5 to 10 % effort required to shift to other standards when artifacts are ready.

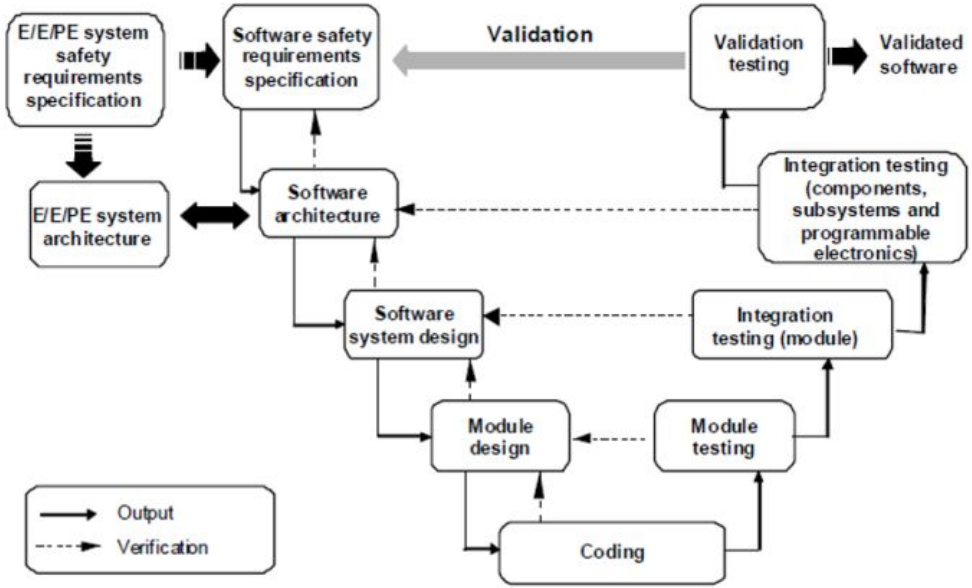
Safety Integrity Levels (SIL/ASIL)

- Safety Integrity Level (SIL) is a risk classification scheme
- The higher the risk, higher the rigour or ‘integrity’ of the development process and the necessary risk reduction
- Based on current requirements received, we plan to target SIL2 level of IEC 61508

Linaro

What do the standards demand ?

- All functional safety standards start by defining a product life cycle.
- Safety requirements need to be fully defined for the project



What do the standards demand ?

Documentation

- Requirements specification
- Documentation on architecture, design and modules, coding standard
- Testing on module and integration level
- Validation of requirements
- Tools, reference hardware configuration

Processes

- Able to demonstrate that requirements are correctly implemented by architecture, unit design, code
- The requirements, architecture, unit design, code and testing comply to the best practices described by the safety regulations
- The development process complies with regulations
- Able to demonstrate that all processes were strictly followed

What do the standards demand ?

Basic Prerequisites

- Documentation of
  - Software Development Process
  - Software Environment
  - Safety Plan
  - Guidelines for Modelling ,Coding
  - Verification Plan
  - Compatibility, dependency and process consistency guidelines

Work Product

- Documentation of software development activities, plans and processes

Long Term Maintenance

- The safety of the end system has to be ensured over its life e.g. 15 years
- Safety-critical Bug investigations shall be performed promptly
- The entire project safety case shall be retained and retrievable

Coding Standards

- Certification does not mandate MISRA-C compliance but it is a de-facto standard for embedded safety, last release 2012
- Need to identify which rules should apply.
  - Establish base coding guideline
  - Agreement with community
- MISRA-C is proprietary and single user license
  - How to make it available to everyone ?
- Commercial Tools for checking MISRA compliance
  - Parasoft's tools
  - Coverity
  - Helix QAC
- Open source tools
  - gcc
  - cpptest
- How to integrate with CI ?

Approach

- A **Safety Element out of context (SEooC)** is a safety-related element which is not developed for a specific item. This means it is not developed in the context of a particular system or vehicle.

- When there is no customers to define input requirements, the product is called Safety Element out of Context (SEooC) and its designed based on **assumptions**.

- It is important to identify the possible use-cases, define these assumptions and decide the scope.

FuSa Assessment and Certification

The purpose of **Functional Safety Assessments** (FUSA) is twofold:

- To ensure that all the activities and documentation for the particular **Safety** Lifecycle (SLC) phase have been completed as per requirements;
- To help prevent systematic failures from being introduced.

FuSa assessment can be done by third parties. It helps in reducing the certification time, provides confidence.

How do we ensure the artifacts we create meet the standards requirements ?

- Have certification/assessment partners join the effort right from the beginning

How should we fund it ?

How to achieve this in an open source project ?

- It is not impossible
  - Tailoring is required
- Efforts underway in
  - ELISA
  - XEN
  - Zephyr
- We need to sync up and pick up from their learnings

Who does what ?

	OP-TEE Open Source maintainers	Linaro**	OEM/Product vendor
Code changes			
MISRA-C			
Documentation			
Commercial Tools			
Assessment			
Certification			
Long term maintenance/ Archive			

** Linaro along with the Assessors

Next Steps

Linaro has the TEE expertise, but we're lacking experience working with safety certification. We need to educate ourselves, talk to the industry, talk to maintainers, create initial plans etc.

Initial Deliverables being planned

a) A report detailing and describing the full scope and giving some rough estimates of what it would take in terms of efforts to have OP-TEE ready for SIL-2/IEC-61508 (equivalent to ASIL-B in ISO-26262).
b) A report describing where OP-TEE is in a need for doing MISRA fixes and give rough estimates of efforts it would take to implement all fixes.

Open topics for discussion

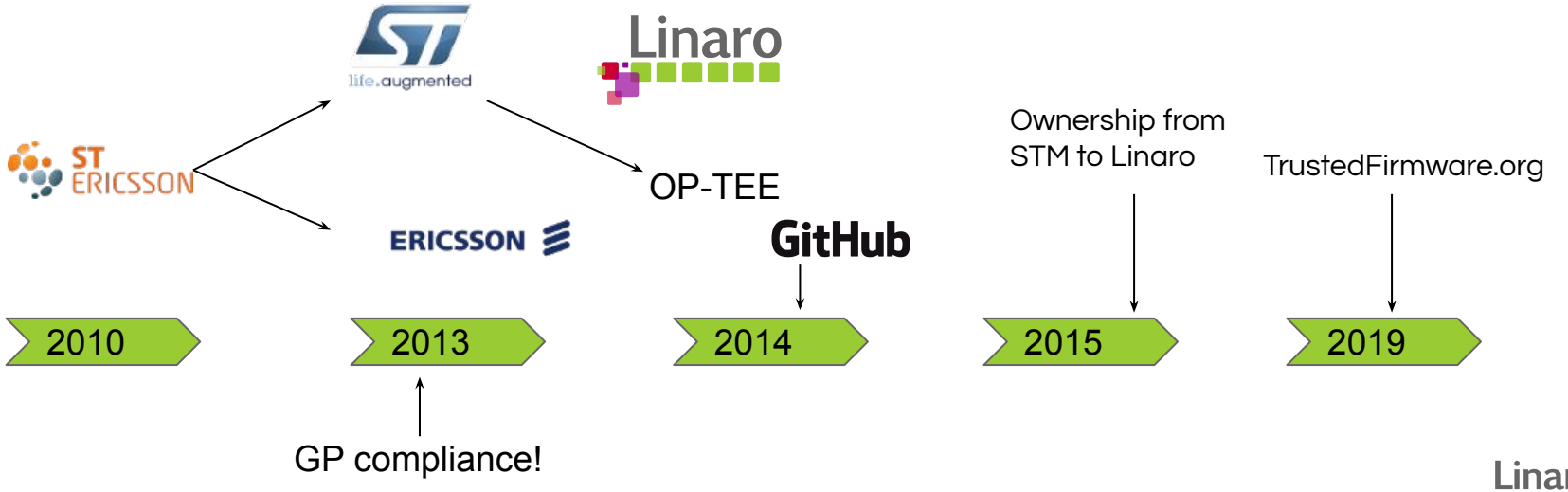
- Safety personnel (Linaro and contractors)
- Other considerations from participants?
- Community organizations and funding?

Appendix A

Appendix A Index

OP-TEE - History and origins

- **O**pen **P**ortable **T**rusted **E**xecution **E**nvironment
- Origins from ST-Ericsson / STMicroelectronics proprietary TEE
- Been developed for >10 years
- GlobalPlatform based TEE



OP-TEE Details

Documentation

https://optee.readthedocs.io/en/latest/general/about.html

Coding Standards

https://optee.readthedocs.io/en/latest/general/coding_standards.html#

Contribution and Reviews

https://optee.readthedocs.io/en/latest/general/contribute.html

- Through github
- Pull Requests are created
- Review feedback - Add fixup patches on top of your existing branch.
- Finalizing your contribution - Once the sender and reviewers have agreed on the patch set, which is when all the people who have commented on the pull request have given their Acked-by: or Reviewed-by:,sender need to rebase, squash commits , add tags and send the PR.

OP-TEE Testing

Testing / quality / SCM

- xtest: The test framework for OP-TEE
- Run and fix checkpatch.pl warnings and errors on all files
- Uses Travis and Linaro CI for continuous integration and automated testing (using QEMU)
- Travis daily cron jobs (full setup, all boards)
- Coverity
- IBART testing
- GP-TEE Compliance - GP test v2.0.0.2
- optee-examples

OP-TEE gits

- optee_client
- optee_os
- optee_test
- optee linux driver
- optee_examples
- optee_benchmark
- optee_docs
- build
- Manifest

Quick Overview of Design

LOC data (using cloc)

Repo	C	C/C++ Headers	Assembly
optee_os	219920	41310	7359
optee_client/libteec	955	133	0
optee_client/tee-supplicant	2836	218	0
optee_test	74588	25948	0
optee_examples	1405	125	0
linux/drivers/tee	4230	581	0

Cyclomatic Complexity (optee_os)

We did some experiments using - https://metrixplusplus.github.io

Results :

[Detailed sheet]

```
ruchika@linaro:~/REFERENCE/optee_os$ python ~/metrixplusplus/metrix++.py view --db-file=/home/ruchika/CYCL_COMP/optee.db
[LOG]: WARNING: Logging enabled with INFO level
[LOG]: INFO:    Processing: ./
./:: info: Overall metrics for 'std.code.complexity:cyclomatic' metric
       Average        : 3.55308137
       Minimum        : 0
       Maximum        : 261
       Total          : 29519.0
       Distribution   : 8308 regions in total (including 0 suppressed)
         Metric value : Ratio : R-sum : Number of regions
                    0 : 0.366 : 0.366 : 3038   |||||||||||||||||||||||||||||||||||
                    1 : 0.154 : 0.520 : 1280   |||||||||||||||
                    2 : 0.115 : 0.635 :  958   ||||||||||||
                    3 : 0.079 : 0.714 :  660   |||||||
                    4 : 0.051 : 0.765 :  420   |||||
                    5 : 0.045 : 0.811 :  378   ||||
                    6 : 0.032 : 0.843 :  267   |||
                    7 : 0.026 : 0.868 :  213   ||
                    8 : 0.018 : 0.887 :  152   |
                    9 : 0.019 : 0.905 :  154   |
                   10 : 0.014 : 0.919 :  117   |
                   11 : 0.012 : 0.931 :   98   |
                   12 : 0.009 : 0.940 :   72
                13-14 : 0.016 : 0.955 :  129   |
                15-16 : 0.009 : 0.964 :   75
                17-19 : 0.010 : 0.974 :   82
                20-23 : 0.008 : 0.982 :   68
                24-27 : 0.006 : 0.988 :   51
                28-40 : 0.006 : 0.994 :   48
               41-261 : 0.006 : 1.000 :   48
```

Linaro

OP-TEE licenses

- BSD 2-Clause
  - Most of the code, but there are also a few libraries with MIT, Apache2
- GPLv2
  - OP-TEE's Linux kernel driver
  - xtest host application (normal world)

- Contributor License Agreement (CLA)
  - Developer Certificate of Origin (DCO)
    https://optee.readthedocs.io/general/contribute.html

- SPDX: In 2017 we added SPDX to all files

OP-TEE releases

- Quarterly releases
  - Right after Linaro Connects
  - Somewhere in between two Linaro Connects
- What devices?
  - Linaro tests all devices at our hands
  - For other devices we rely on external contributors (maintainers)
- Release notes?
  - Latest changes can be read about in the CHANGELOG.md file
  - Follows Semantic Versioning 2.0.0

OP-TEE security issues

- Security advisories page at optee.org
  - https://www.op-tee.org/security-advisories
  - We request CVE's when appropriate
- Vulnerability Reporting Process
  - The OP-TEE project as part of the TrustedFirmware.org organization is using the security incident process as described at the TrustedFirmware.org security incident page.
- How to report security issues?
  - To report an issue, please follow the process as specified here. The email address to use can be found at the Mailing Aliases page.

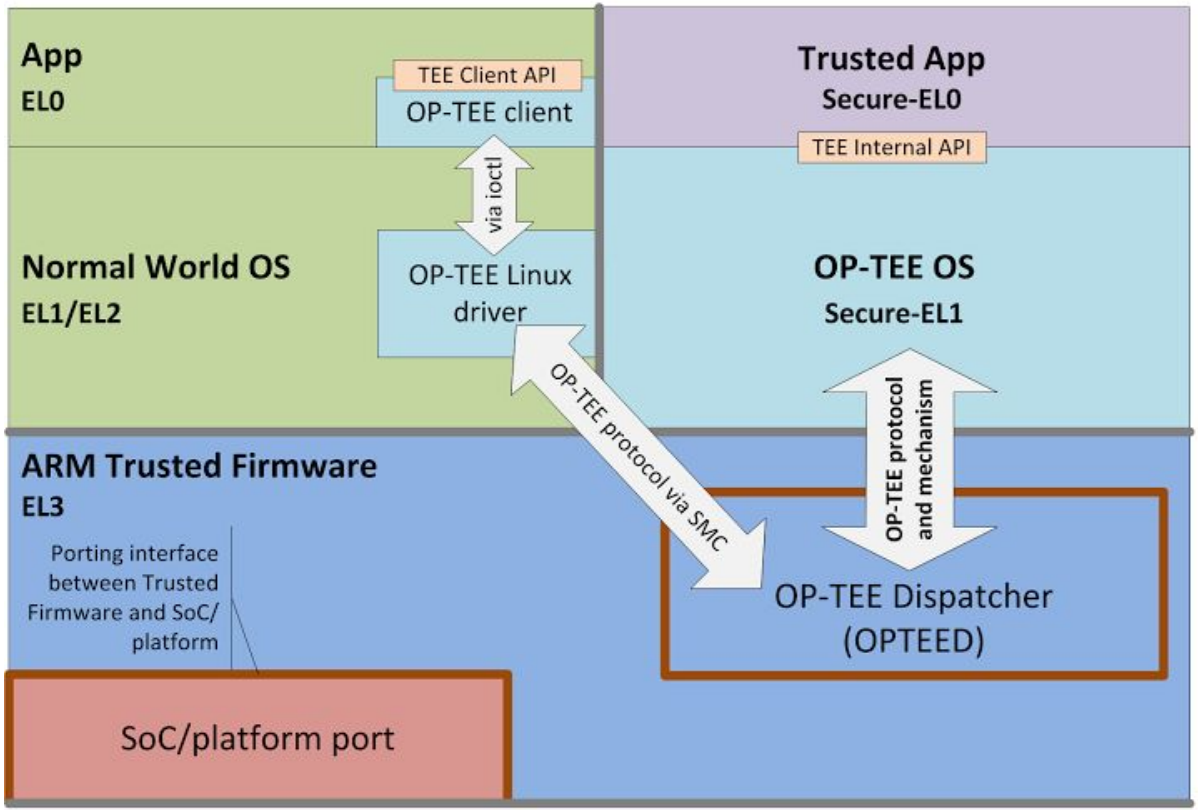
Appendix A Index

- OP-TEE - History and Origin
- OP-TEE
  - Details
  - Testing
  - Gits
  - LOC data
  - Cyclomatic Complexity
- OP-TEE Licenses
- Releases
- Security Issues

- ARM Trusted Firmware and OP-TEE
- Normal world
  - optee_client
  - Generic TEE driver
  - Shared Memory
- Secure world
  - SMC Interface
  - GlobalPlatform APIs
  - Cryptographic Abstraction Layer
  - Secure Storage
    - REE FS
    - RPMB FS
  - Trusted Applications
- Xtest

Linaro

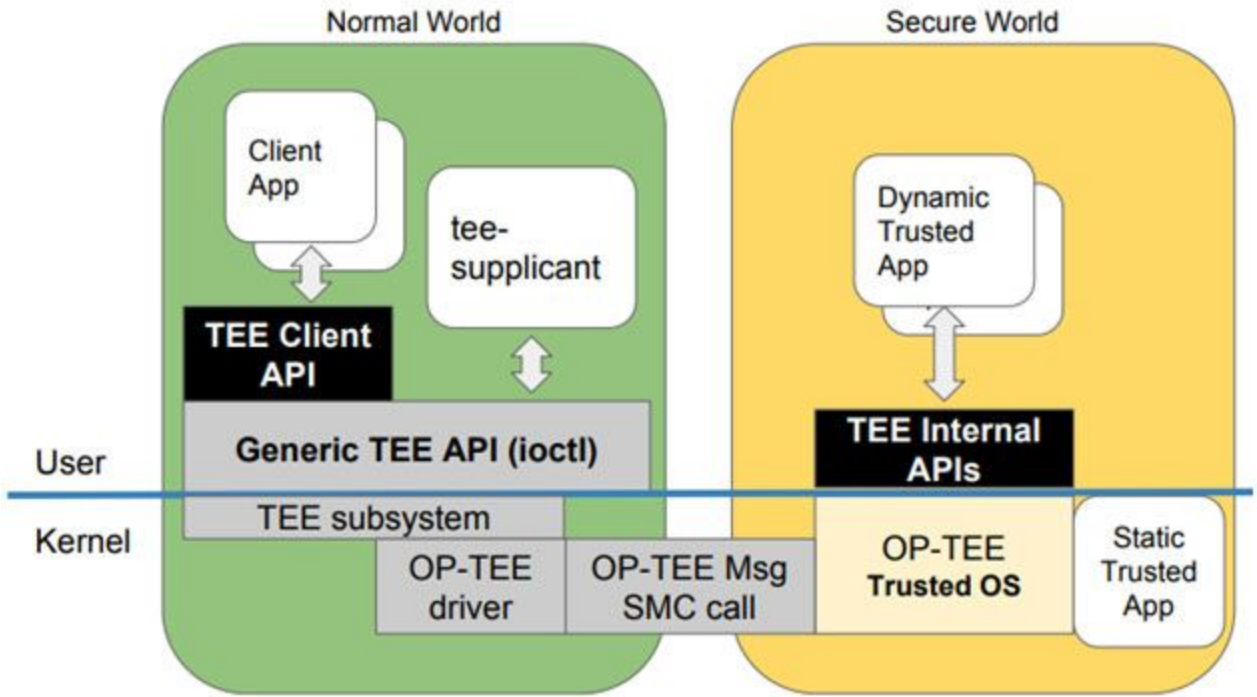
ARM Trusted Firmware and OP-TEE



ARM Trusted Firmware
github.com/OP-TEE
SoC supplier
OS/hypervisor supplier
Trusted App supplier
Global Platforms spec.
Internal OP-TEE interface

SMC Calls to EL3 are specified by the SMC Calling Convention PDD (ARM DEN 0028A)

OP-TEE is an open source Trusted OS implementing the Global Platform TEE specifications

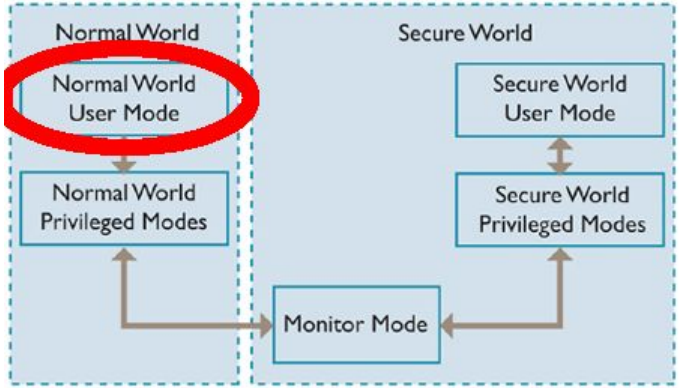

Normal World
Secure World
Client App
tee-supplicant
Dynamic Trusted App
TEE Client API
TEE Internal APIs
User
Generic TEE API (ioctl)
Kernel
TEE subsystem
OP-TEE driver
OP-TEE Msg SMC call
OP-TEE Trusted OS
Static Trusted App

optee_client

- **Libtee.so**
  - Client library serving user space with TEE access
  - Implements GP Client API v1.0
  - Communication with the TEE goes through the Generic TEE Linux kernel driver

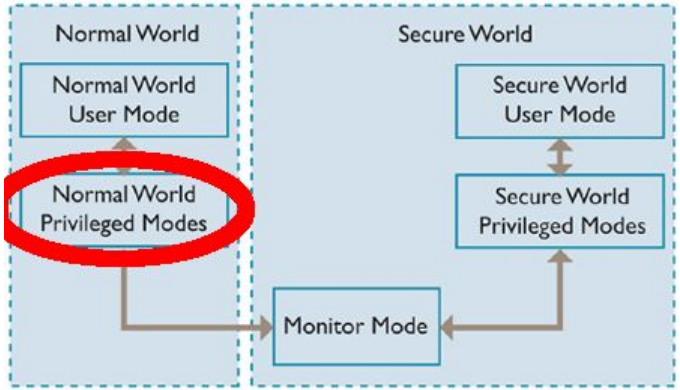
- **Tee-supplicant**
  - Daemon serving secure world with normal world features, like for example file system access.

    Communicates with secure world using RPC messages

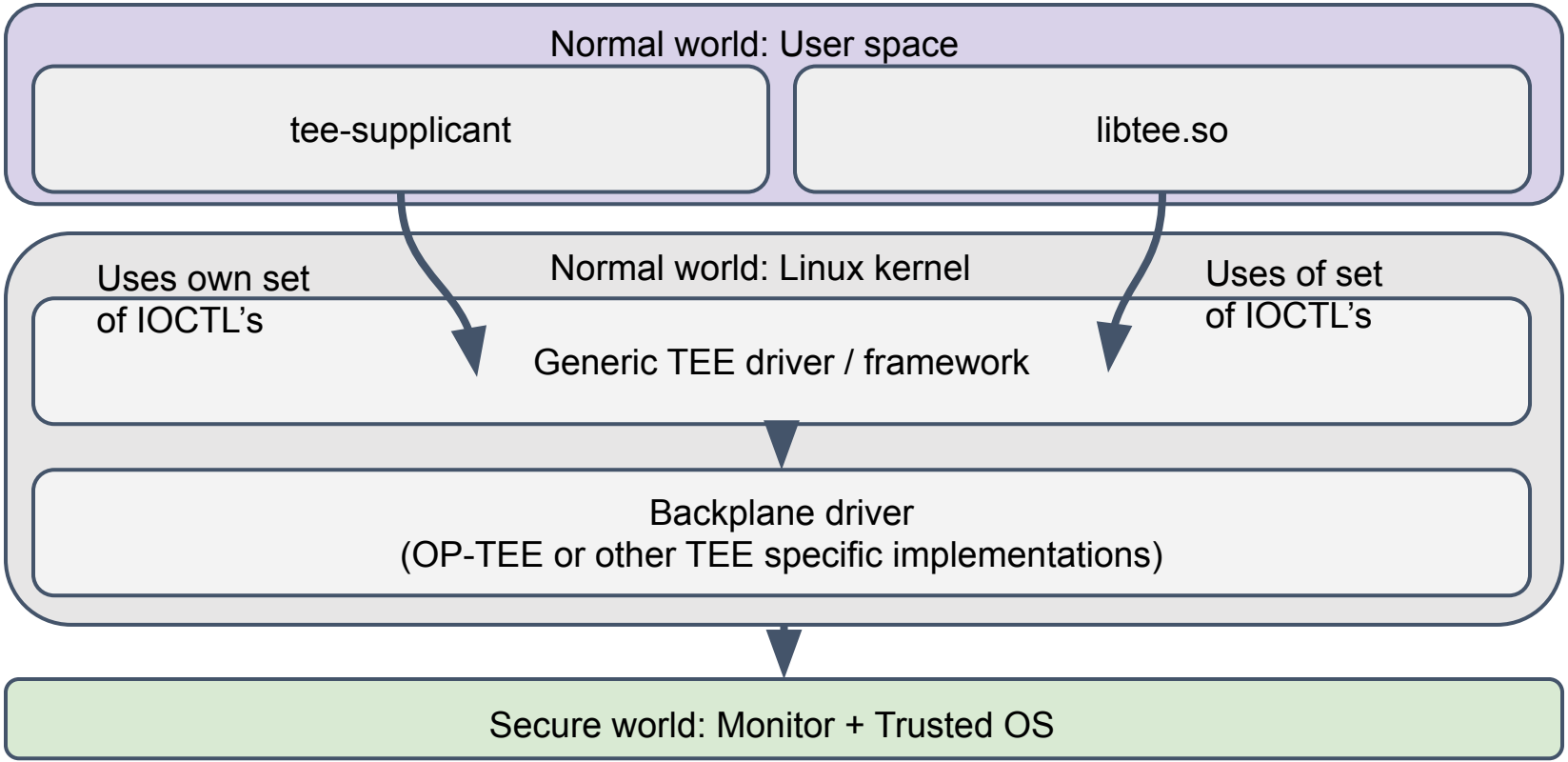


Generic Linux kernel TEE driver

- Consists of a TEE framework and drivers
  - A generic framework handling communication with user space
  - A backplane driver aimed for a special platform / hardware
- Manages the shared memory pool
- (Contains a “kernel API” exposing the same functionality as GP Client API does in optee_client)

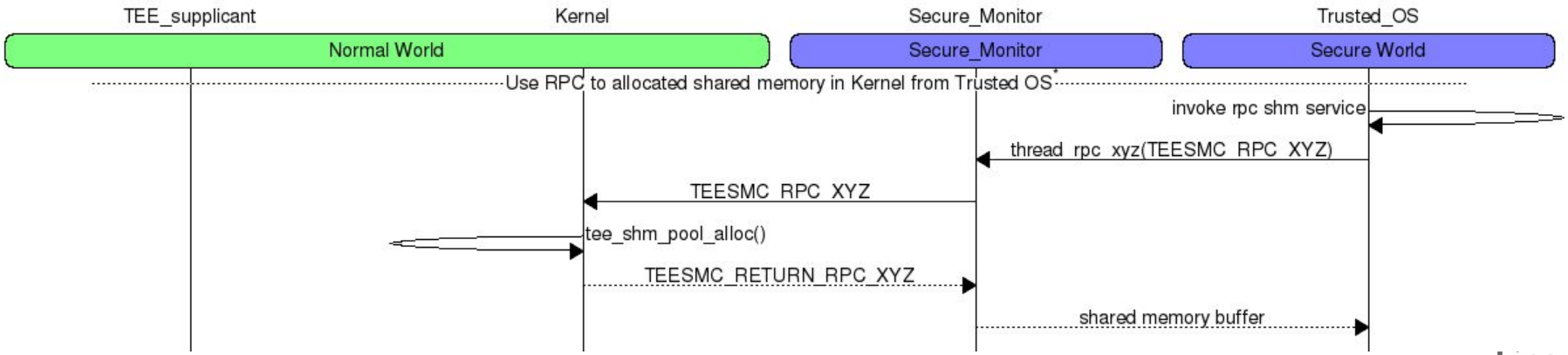


Generic TEE driver communication



Shared memory

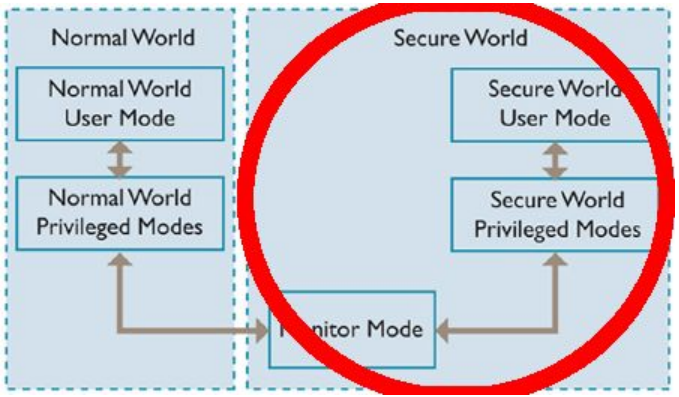
- OP-TEE's Linux kernel driver maintains shared memory
- It uses a piece of DRAM (configured by secure world) that is handled as a memory pool
- Memory allocations can be triggered both from normal world and secure world (involves RPC messaging)



Appendix A Index

Linaro

Secure world



SMC Interface - teesmc

- OP-TEE follows Arm's SMC calling convention
- However the SMC calling convention does not mandate how the TEE data should be handled, therefore we have created optee_msg protocol

GlobalPlatform

- GP Client API v1.0
  - Used by clients both in user space and Linux kernel
- GP Internal API v1.1
  - Cryptographic API
  - Secure Time
  - Secure Storage
  - Arithmetical API
- GP Socket API
  - TCP and UDP support in Trusted Applications

Cryptographic Abstraction Layer

- Default cryptographic software library in OP-TEE is LibTomCrypt, but mBedTLS can also be used from a Trusted Application point of view
- TEE Cryptographic Operations API is a set of APIs in OP-TEE that makes it easier to add support for other cryptographic engines (both software and hardware)

```
-   some_function()                                 (Trusted App) -
[1]   TEE_*()                       User space      (libutee.a)
------- utee_*() ------------------------------------------------
[2]       tee_svc_*()               Kernel space
[3]         crypto_*()                              (libtomcrypt.a and crypto.c)
[4]           /* LibTomCrypt */                     (libtomcrypt.a)
```

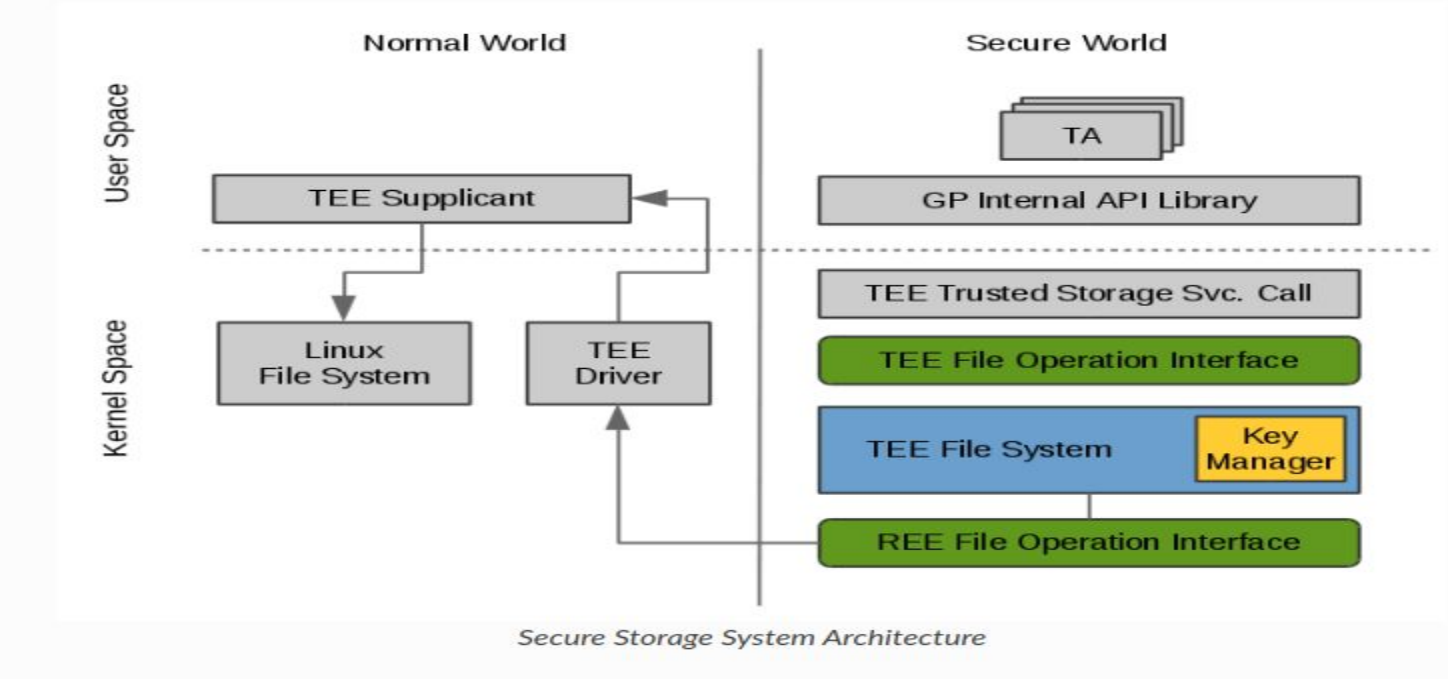
- [3] The crypto_*() functions implement the actual algorithms and helper functions.
- For more details, please read the crypto documentation

Secure Storage

- Secure Storage in OP-TEE is implemented according to what has been defined in GlobalPlatform's TEE Internal Core API (here called Trusted Storage). This specification mandates that it should be possible to store general-purpose data and key material that guarantees confidentiality and integrity of the data stored and the atomicity of the operations that modifies the storage (atomicity here means that either the entire operation completes successfully or no write is done).
- There are currently two secure storage implementations in OP-TEE:
  - The first one relies on the normal world (REE) file system.
  - The second one makes use of the Replay Protected Memory Block (RPMB) partition of an eMMC device, and is enabled by setting CFG_RPMB_FS=y.

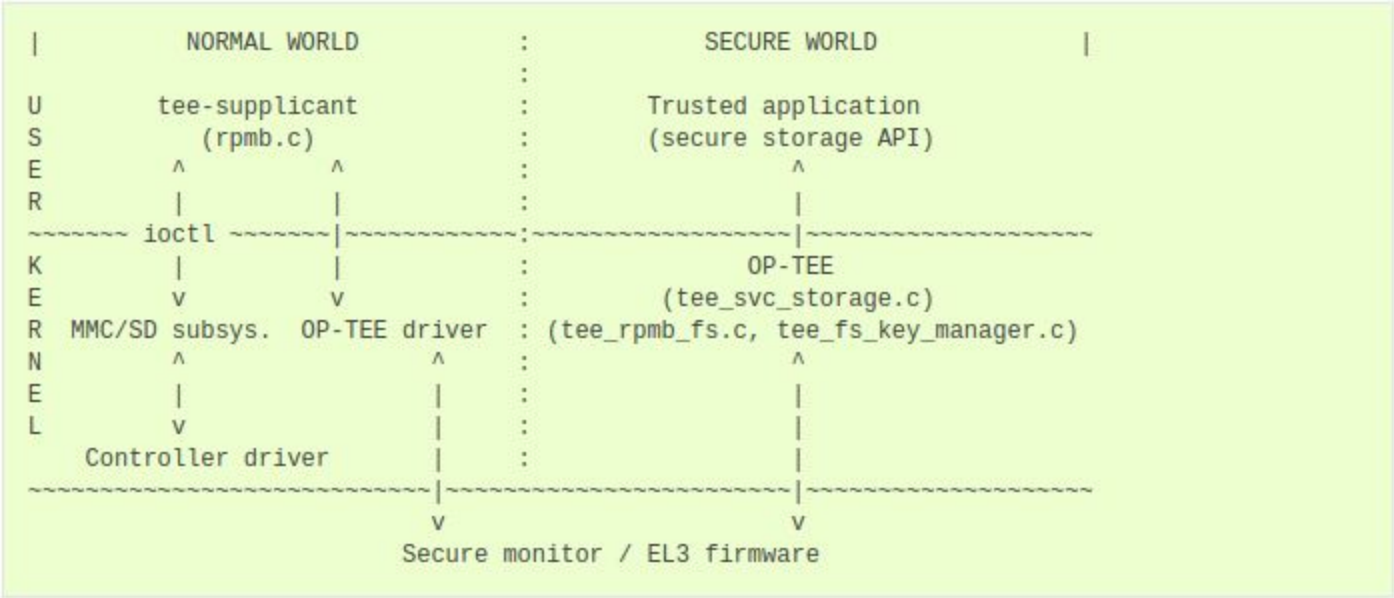
For more details, please read the secure storage documentation

REE FS Secure Storage



Secure Storage System Architecture

RPMB secure storage



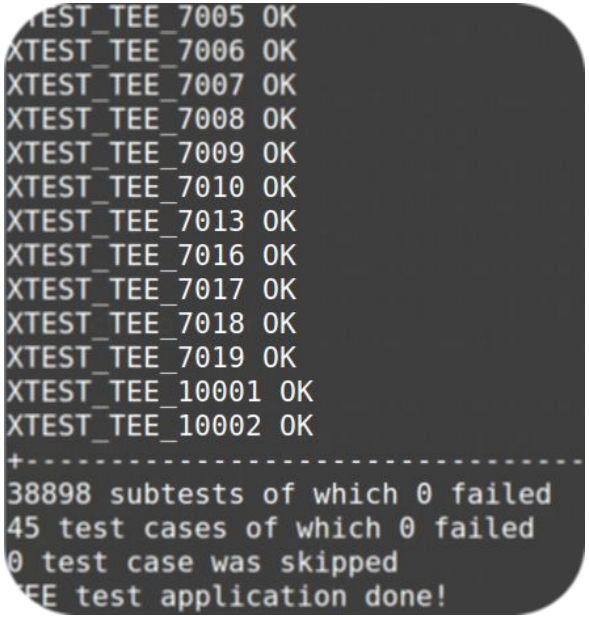
Trusted Applications

- User TA's
  - Stored as signed ELF files in flash or as installed TA's in secure storage
  - Loaded using RPC messaging
  - Signed using RSA signature (PKCS #1 v1.5, SHA256)
  - Isolation (separate stack, heap etc)
  - TA 2 TA communication

- Static / Psuedo TA's
  - Compiled into TEE core, running in kernel space
  - Cannot really compare this to a User TA more than they use the same interface

xtest

- This is the main test suite for OP-TEE
- Possible to extend the test suite to also make use of the GP TEE Compliance test suite
- Uses TA-dev-kit from optee_os and TEE client API from optee_client



Linaro

Thank you