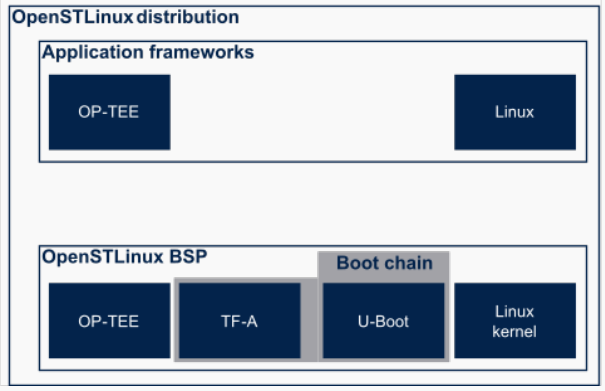
OP-TEE overview

Stable: 13.03.2020 - 08:30 | Revision: 13.03.2020 - 08:30

Contents

1 Overview of the OP-TEE open source project

OP-TEE allows the development and integration of secure services and applications under trusted execution environments, that is execution environments isolated from the Linux® -based OS .



Zoom out to OpenSTLinux distribution

Description extracted from the OP-TEE site[1]:

> *"OP-TEE is an open source project, which contains a full implementation to make up a complete Trusted Execution Environment using the ARM®TrustZone®. technology. OP-TEE meets the GlobalPlatform TEE System Architecture specification. It also provides the TEE Internal core API v1.1 as defined by the GlobalPlatform TEE Standard for the development of Trusted Applications. OP-TEE Trusted OS is accessible from the Linux based OS using the GlobalPlatform TEE Client API Specification v1.0, which also is used to trigger secure execution of applications within the TEE."*

OP-TEE is delivered under a BSD style license and can run secure (trusted) applications without restriction on their licensing model.

The OP-TEE project is maintained by the Linaro Security Working Group.

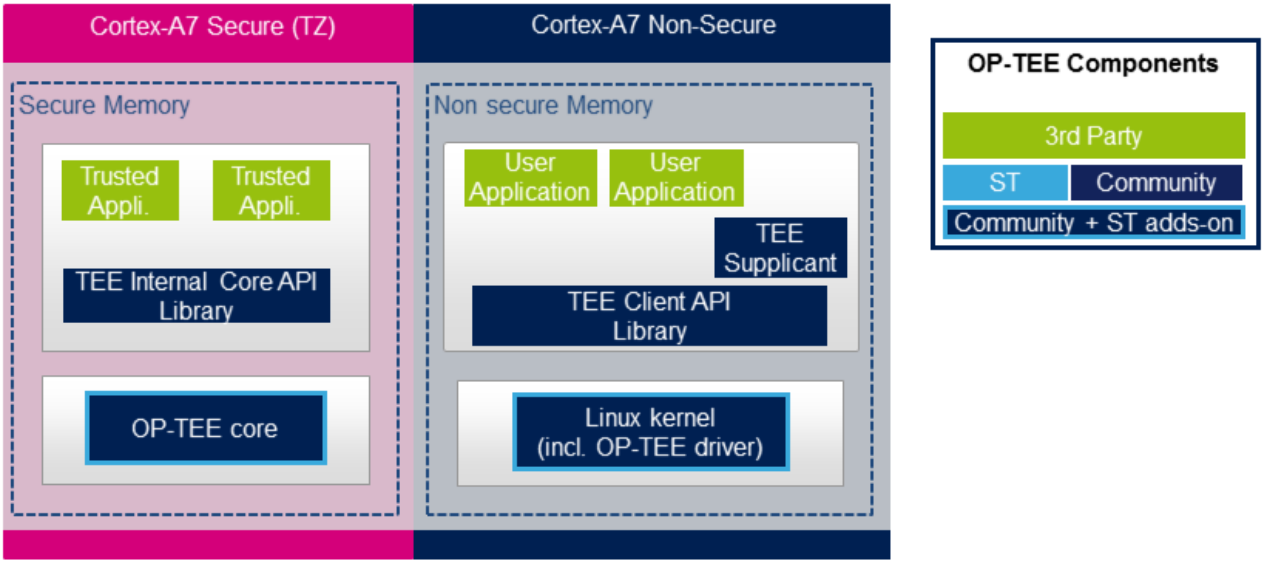
- OP-TEE official site[1]
- OP-TEE source repositories [2][3][4]
- OP-TEE documentation[5]

GlobalPlatform Device TEE specifications (TEE Client API, TEE Internal Core API and few more) is available from the GlobalPlatform site[6].

2 Architecture

The OP-TEE project includes several secure and non-secure embedded components, as well as some tools for development and debugging purposes.

The figure below shows the main OP-TEE embedded components, namely the OP-TEE core and trusted application standard libraries on the secure side, and the Client API library, the OP-TEE supplicant daemon and the OP-TEE Linux kernel driver on the non-secure side.



2.1 OP-TEE core

The main OP-TEE component is the OP-TEE core. The OP-TEE core execution is done in Arm®Cortex®-A secure state while the non-secure world (likely a Linux based OS) is done in the non-secure state of the processor. The OP-TEE core executes in secure privileged (kernel) mode, while trusted applications are executed in secure user mode.

OP-TEE can load signed trusted applications stored in the Linux OS file system or embedded in the OP-TEE core boot image.

On devices with secure external memory, the OP-TEE core runs as a monolithic image in the secure memory. On devices with a small secure memory, the OP-TEE core can run in paging-on-demand configuration: a small resident agent is loaded in the small secure memory and can securely page-in/page-out data from/to the non-secure (or less secure) external memory.

OP-TEE core source files can be found from optee_os repository [2].

2.2 OP-TEE trusted libraries

OP-TEE embeds utility libraries for trusted application development including the GlobaPlatform Device TEE Internal Core API Library, which provides the standard services a trusted application can expect from the TEE. OP-TEE supports the loading of static and dynamic libraries in the TEE.

The OP-TEE standard trusted application libraries source files can be found in the optee_os repository[2].

2.3 TEE Linux driver

The OP-TEE Linux driver is part of the Linux kernel since release 4.12.

The OP-TEE Linux driver is enabled via the CONFIG_OPTEE configuration directive through the usual Linux kernel configuration means. The driver can be probed thanks to a device tree node.

2.4 TEE Client API

The OP-TEE project embeds an implementation of the GlobaPlatform Device TEE Client API specification for Linux based OS. This TEE Client API specification is partly implemented as a userland library and partly as a Linux kernel OP-TEE driver. The API allows userland clients to invoke trusted applications and the OP-TEE core services exported to non-secure world with a standard API.

The OP-TEE Client API library source files can be found in the optee_client repository[3].

2.5 TEE supplicant

The OP-TEE core can rely on non-secure remote services. OP-TEE embeds an implementation of a non-secure userland supplicant, that can be invoked by the OP-TEE core through the OP-TEE Linux kernel driver. An example of such service is the access to a non-volatile media device that is controlled in the non-secure world.

The OP-TEE supplicant source files can be found in the optee_client repository[3].

2.6 Host tools

The OP-TEE optee_os component, once built, generates a so-called Trusted Application Development Kit to ease the development and integration of trusted applications on a target system. The Trusted Application Development Kit includes the libraries, with their header files and makefile scripts, that allow the generation of signed trusted applications from their respective source files.

Optee_os package also provides a tool to analyse call stack backtraces in case of trusted application and/or OP-TEE core crash. Refer to script **symbolize.py** in optee_os source tree[2].

3 Booting with OP-TEE

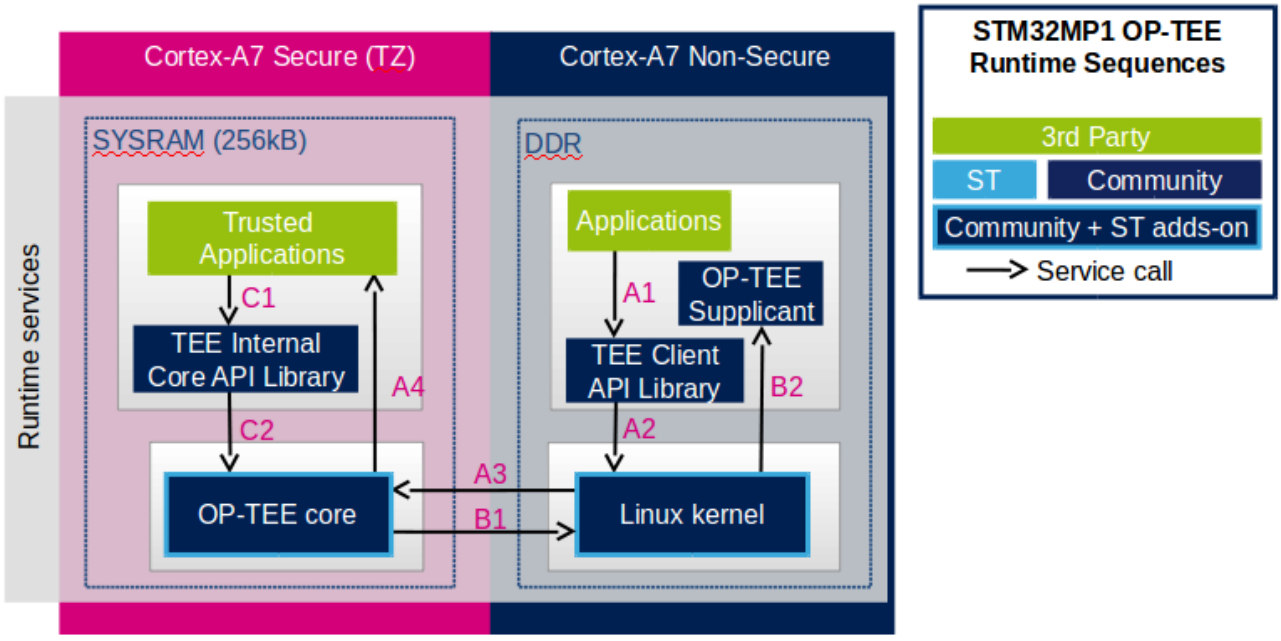
The OP-TEE core is a secure firmware. It must be booted prior to the non-secure world on Arm Cortex-A core(s). The secure bootloader must therefore load the OP-TEE core images in memory and run its initialization prior to executing the first booted non-secure image.

Refer to the target system boot sequences for more details.

4 Invoking the OP-TEE services from Linux based OS

Once the Linux kernel is booted, the OP-TEE core is already initialized and ready to serve.

The figure below shows the main run time sequences in which the OP-TEE can be involved.



Sequence A: an non-secure application invokes a service from a trusted application.

The non-secure application calls the TEE Client API library (**A1**), which in turns invokes (**A2**) the Linux kernel OP-TEE driver. The OP-TEE driver invokes the secure world (**A3**) and reaches the OP-TEE core. The last OP-TEE core transfers the request (**A4**) to the target trusted application. Once the trusted application has completed the request, the system branches back to the calling application with the request status.

If an invoked trusted application is not yet loaded into the TEE, the OP-TEE core loads it by calling remote services through the non-secure TEE supplicant as described in **sequence B** below.

In addition, any invocation of the TEE from the non-secure world must go through the Linux kernel OP-TEE driver.

Sequence B: the OP-TEE core must invoke a non-secure remote service.

The OP-TEE core invokes (**B1**) the Linux kernel OP-TEE driver which in turns notifies the TEE supplicant daemon (**B2**) for a request. Once the supplicant has completed the request, the system branches back to the OP-TEE core with the request status.

Sequence C: a trusted application invokes an OP-TEE core service.

Most of the services defined by the GlobaPlatform Device TEE Internal Core API must be executed in OP-TEE core privileged mode. The trusted application calls the corresponding service from the TEE Internal Core API library (**C1**), which issues a system call (**C2**) to the OP-TEE core. Once the core has completed the request, the system branches back to the calling trusted application with the request status.

5 Experiencing OP-TEE on a target

First make sure your setup includes OP-TEE in the boot sequence. If the OP-TEE core console traces are enabled, you should see the OP-TEE banner after secure bootloader traces and before non-secure bootloader traces. The OP-TEE core banner looks like this:

```
I/TC:  OP-TEE version: <some-reference-version-info> #1 Mon Jun 25 08:59:21 UTC 2018 arm
I/TC:  Initialized
```

The Linux kernel boot traces also show the successful probing of the OP-TEE Linux kernel driver:

```
optee: probing for conduit method from DT.
optee: initialized driver
```

The OP-TEE non-secure components are stored in the file system:

- By default the TEE supplicant is installed at **/usr/bin/tee-supplicant**.
- By default, the TEE Client API library is installed at **/usr/lib/teec.so**.
- By default the TEE regression test tool is installed at **/usr/bin/xtest**.

In the default OP-TEE configuration, trusted applications are stored in the non-secure filesystem at **/lib/optee_armtz/*.ta**.

OP-TEE provides means to protect the trusted application binary images from corruption as image signature or installation in the OP-TEE secure storage. In any case, it is likely that Othe P-TEE core needs to invoke a non-secure service to retrieve the trusted application(s) from some non-secure filesystem data in order to load trusted application(s) in the TEE. This service requires the availability of the OP-TEE supplicant.

Therefore, once the non-secure OS has booted, it must launch the OP-TEE supplicant as a background daemon. Use the following shell command to start the OP-TEE supplicant from a booted Linux system, :

```
sh> tee-supplicant &
```

The OP-TEE package comes with some examples and regression tests. Use the following embedded shell command to run the regression tests:

```
sh> xtest
```

or to run only selective tests:

```
sh> xtest 1002     # Invokes some OP-TEE internal core services
sh> xtest 1004     # Invokes a trusted application loaded from the non-secure filesystem
```

6 References

- [1.01.1] https://op-tee.org
- [2.02.12.22.3] https://github.com/OP-TEE/optee_os
- [3.03.13.2] https://github.com/OP-TEE/optee_client
- https://github.com/OP-TEE/optee_test
- https://optee.readthedocs.io/
- https://globalplatform.org/

Open Portable Trusted Execution Environment

Linux® is a registered trademark of Linus Torvalds.

Operating System

TrustZone®

Arm® and TrustZone® are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Trusted Execution Environment

Application programming interface

Arm® is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Cortex®

Device Tree