Demystifying ARM TrustZone TEE Client API using OP-TEE

Heedong Yang
Department of Computer Engineering, Hannam University
Daejeon, Republic of Korea
heedong.yang.kr@gmail.com

Manhee Lee
Department of Computer Engineering, Hannam University
Daejeon, Republic of Korea
manheelee@hnu.kr

ABSTRACT

Recently, sensitive information such as financial data and electronic payment systems have been stored in mobile devices. To protect important data, TEE technology has emerged, a trusty and safe execution environment. In particular, ARM TrustZone technology, which is mainly used in mobile, divides one physical processor into Normal World and Secure World to provide a safer execution environment. Many manufacturers have started using TrustZone technology, but existing commercial TEEs have limitations in conducting security research using TrustZone. Therefore, this paper shows how to use OP-TEE which is an open source project for implementing ARM TrustZone technology and TEE Client API that communicates with Trusted Application of TrustZone Secure World. To demystify TEE Client API, we implemented a simple trusted application for communication between Normal World and Secure World in OP-TEE OS using QEMU emulator.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; **Embedded hardware**.

KEYWORDS

ARM Processor, TrustZone, Trusted Execution Environment, OP-TEE

ACM Reference Format:
Heedong Yang and Manhee Lee. 2020. Demystifying ARM TrustZone TEE Client API using OP-TEE. In *The 9th International Conference on Smart Media and Applications (SMA 2020), September 17–19, 2020, Jeju, Republic of Korea.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3426020.3426113

1 INTRODUCTION

The advances in mobile technology and the spread of smartphones have made smartphones important in our lives. As smartphone stores very sensitive data due to services such as Samsung Pay, Apple Pay, and Mobile Banking, there are increasing attacks to steal them. According to SKYBOX Security's Vulerability and Threat Trends Report, vulnerabilities and exploits continue to increase. In particular, Google Android accounts for 35% of all vulnerabilities and is the highest in top 10 product list[7]. Thus, TEE(Trusted Execution Environment) technology was proposed to store and execute important data safely in mobile environments. TEE is a hardware security technology that isolates the execution of secure programs from other programs[10]. Among many embedded devices using ARM TrustZone technology[6], Samsung Knox which is used to secure Samsung Pay and Samsung smartphones is the representative TrustZone technology platform[11].

ARM TrustZone has support for Global Platform TEE Client API and TEE Internal Core API. TEE Client API is used to communicate with Trusted Application. However, recently a new attack appeared targeting Trusted Applications, so security study is necessary[3]. This paper will help to implement the TEE Client API used to communicate with Trusted Application for TrustZone security study. Commercial TEEs have limitations in conducting security research using TrustZone. Therefore, we implemented TrustZone environment using OP-TEE(Open Portable Trusted Execution Environment), an open source project[8]. This paper is composed as follows. Section 2 introduces ARM TrustZone and open source project OP-TEE for ARM TrustZone Implement. Section 3 introduces TEE Client API. In addition we directly implement TrustZone environment using OP-TEE and trusted application using TEE Client API. Concluding remarks are included in Section 4.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SMA 2020, September 17–19, 2020, Jeju, Republic of Korea
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8925-9/20/09…$15.00
https://doi.org/10.1145/3426020.3426113

2 BACKGROUND

2.1 Structure of ARM TrustZone

ARM Processors use ARM TrustZone technology to implement the TEE environment. ARM TrustZone is a hardware security technology that divides one physical processor into Normal World and Secure World to run software safely. Both worlds are divided

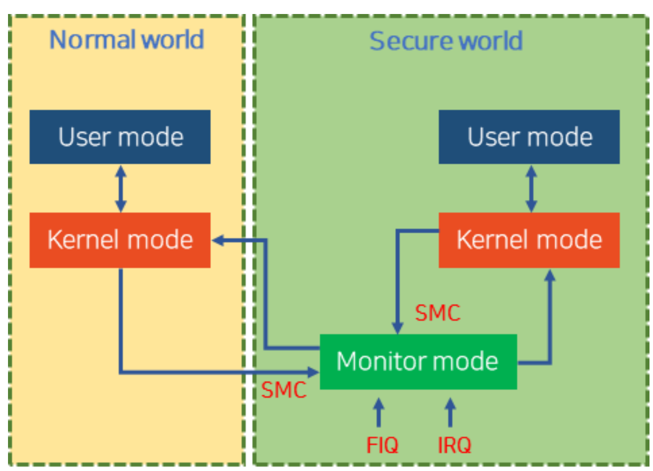


Figure 1: Stucture of ARM Trustzone[6]

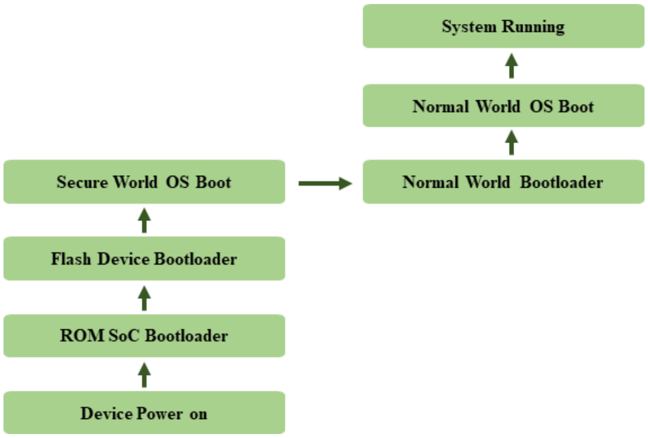


Figure 2: TrustZone Secure boot Process[6]

into user mode and privileged mode according to Exception Level. However, since the two worlds do not run at the same time, a switch is required to change from the normal world to the secure world. For this, Monitor Mode was added in existing ARM operation modes(User, Abort, FIQ, Undefined, IRQ, System, and SVC Mode). Monitor Mode operates only when SMC(Secure Monitor Call), IRQ, or FIQ occurs. Therefore, Normal World applications running in the user mode are unable to switch between the worlds. In particular, jobs that require security such as TIMA(TrustZone-based Integrity Measurement Architecture), mobile payment services, and DRM are executed only through Secure World. Normal World applications request execution from Secure World through SMC[6]. ARM TrustZone provides security process to ensure platform integrity through a boot process called Secure Boot as shown in Fig. 2. The Secure Boot process begins with running the ROM SoC Bootloader to initialize peripherals such as memory controller. Next, the Secure World OS boots from the flash device and then the Normal World OS boots. Boot starts first in Secure World for a security check before Normal World's application might have a chance to modify the system[6].

2.2 OP-TEE(Open Portable Trusted Execution Environment)

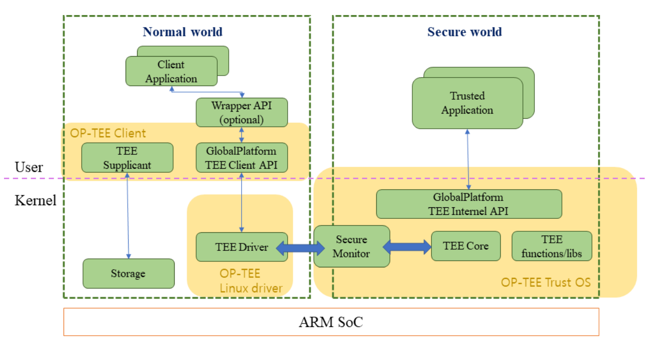


Figure 3: OP-TEE Software Architecture[1]

OP-TEE is an open source TEE that implements TrustZone technology. OP-TEE project is managed and deployed by Linaro, non-profit organization that aims to open source on Linux based on ARM. It is designed to use ARM TrustZone technology and is implemented according to Global Platform's TEE Client API 1.0 and Global Platform TEE Internal API 1.0[9]. OP-TEE consists of three components: OP-TEE Client, OP-TEE Linux driver, and OP-TEE Trusted OS shown in Fig. 3. It also ensures platform integrity with TrustZone Secure boot. The OP-TEE project provides the ability to implement a TrustZone environment on a real device or virtual environment, and officially supports multiple platforms[9].

3 COMMUNICATES WITH TRUSTED APPLICATION

3.1 TEE Client API

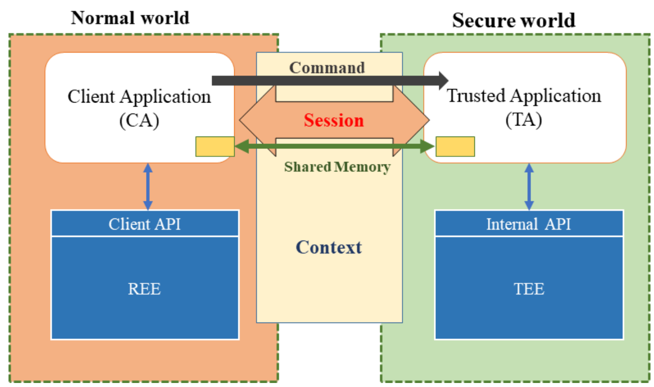


Figure 4: TEE Client API Concepts

ARM TrustZone supports TEE Client API to communicate with TEE Trusted Application in Secure World. Global Platform specifies TEE Client API concepts as shown in Fig. 4 so that CA(Client Application) in REE(Rich Execution Environment) can communicate with TA(Trusted Application) in TEE[5]. REE is a modern operating system such as Windows, Linux, Mac OS, Android or iOS. TEE is

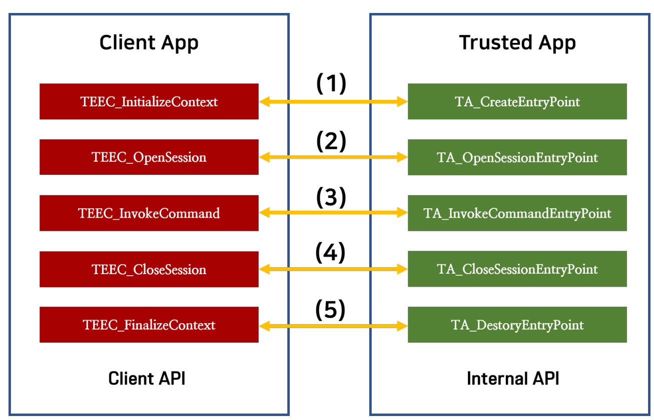


Figure 5: Communication process Using TEE Client API

an environment that can be executed separately from REE, such as QSEE[2], OP-TEE, Kinibi, Trustcore or TEEGRIS[4].

Table 1: Data Types of Client API[5]

Struct	Description
TEEC_Result	Defined to contain the return code that is the result of calling a TEE Client API function
TEEC_UUID	Contains a UUID type defined in RFC4122 and used to identify the TA
TEEC_Context	Logical container that associates the CA with a specific TEE
TEEC_Session	Logical container linking a CA with a particular TA
TEEC_Shared Memory	CA memory blocks registered or allocated to shared memory
TEEC_Temp Memory Reference	Shared memory temporarily created by TA service request
TEEC_Registered Memory Reference	Define a memory reference to use some of TEEC_SharedMemory for TA service requests
TEEC_Value	32-bit unsigned integer that does not refer to shared memory but is passed by value
TEEC_Parameter	Define the parameters of TEEC_Operation
TEEC_Operation	Define TEEC_Parameter and Data Delivery Direction

Generally, the application of TrustZone is divided into CA running in REE and TA running in TEE. CA and TA cannot be connected directly because they run in a separate environment, but when it calls TEE Client API and Internal API functions in pairs like Fig. 5., it can communicate as shown in Fig. 4[5].

CA needs to connect to Secure World (TEE) before connecting to TA. Logical connection between CA and Secure World is called Context. Context is initialized in CA through the *TEEC_InitializeContext* function. When the instance of the TA is created, TA calls *TA_CreateEntryPoint* function as shown in Fig. 5 (1). Additionally, one CA may have multiple contexts.

The connection between CA and TA after the initialization of context is called Session. TEE Session is the logical container linking a CA with a particular TA. TEE Session is created in CA through *TEEC_OpenSession* function. TA connects by calling the *TA_OpenSessionEntryPoint* function as shown in Fig. 5 (2). UUID (Universally Unique Resource Identifier) information is required to open a session. At this time, one CA can open multiple sessions for a several of TAs that can know UUID. But, please note that CA cannot connect to all TAs and only can connect to a specific TA. As shown in Fig. 4, communication between CA and TA uses shared memory, and CA and TA call Command in specified Session through *TEEC_InvokeCommand* and *TA_InvokeCommandEntryPoint* as shown in Fig. 5(3).

Finally, to complete the communication, the Session is closed and the context is finalized. Therefore, CA and TA call functions that

Table 2: CA Operation function of Client API[5]

Function	Description
TEEC_Initialize Context	Create a context, a logical connection between TEE and CA
TEEC_Finalize Context	Release the logical connection stored in the context
TEEC_Open Session	Create session by connecting TA and CA specified by UUID
TEEC_Invoke Command	Service request by function or service ID of TA connected to Session
TEEC_Close Session	Terminate TA connection with CA stored in Session
TEEC_Register SharedMemory	Register CA's memory block in shared memory in context scope
TEEC_Allocate SharedMemory	Allocate CA memory block to shared memory in context scope
TEEC_Release SharedMemory	Deallocate the block of memory from shared memory
TEEC_Request Cancellation	Create Session or Stop TA Service
TEEC_PARAM _TYPES	Set parameter directionality of TEEC_Operation

Table 3: TA Interface function of internal API[5]

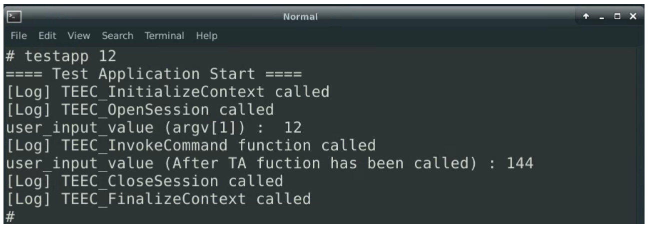
Function	Description
TA_CreateEntry Point	Run the first time the CA to TA connection is run
TA_OpenSession EntryPoint	Run the first time the CA to TA connection is run
TA_Invoke Command EntryPoint	Paired with TEEC_InvokeCommand to provide the service according to the function or service ID of the TA
TA_CloseSession EntryPoint	Paired with TEEC_CloseSession to disconnect CA and TA
TA_Destroy EntryPoint	Run when CA to TA is completely terminated

TEEC_CloseSession and *TA_CloseSessionEntryPoint*, *TEEC_Finalize Context* and *TA_DestoryEntryPoint* as shown in Fig. 5(4), Fig. 5(5).

Client API is defined in header file named “*tee_client_api.h*”. The C Data type used by the Cilent API is defined as shown in Table 1. And functions of TEE Client API and TEE Internel API are shown in Table 2, Table 3[5].

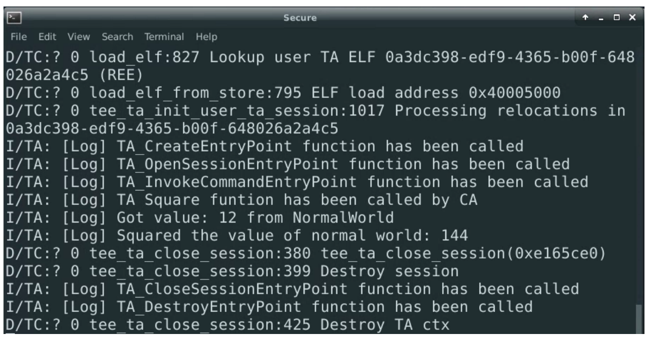
3.2 TEE Client API on OP-TEE

Commercial TEEs are limited in implementing and experimenting with read devices Trusted Application for security research. Thus, we developed Trusted Application using TEE Client API in OP-TEE. We built QEMUv8 according to the instructions in the OP-TEE documentation for ARMv8 64-bit TrustZone virtual environment[9]. After a successful build with no errors and the QEMU emulator running, jobs of Normal World and Secure World can be monitored through two terminal windows. In order to test communication process between CA and TA, we developed a simple test application using TEE Client API. If it works, it calls the TA function that returns the squared integer through the TEEC_InvokeCommand function.



```
Normal
File Edit View Search Terminal Help
# testapp 12
==== Test Application Start ====
[Log] TEEC_InitializeContext called
[Log] TEEC_OpenSession called
user_input_value (argv[1]) :  12
[Log] TEEC_InvokeCommand function called
user_input_value (After TA fuction has been called) : 144
[Log] TEEC_CloseSession called
[Log] TEEC_FinalizeContext called
#
```

Figure 6: TEST APP running on Normal World



```
Secure
File Edit View Search Terminal Help
D/TC:? 0 load_elf:827 Lookup user TA ELF 0a3dc398-edf9-4365-b00f-648
026a2a4c5 (REE)
D/TC:? 0 load_elf_from_store:795 ELF load address 0x40005000
D/TC:? 0 tee_ta_init_user_ta_session:1017 Processing relocations in
0a3dc398-edf9-4365-b00f-648026a2a4c5
I/TA: [Log] TA_CreateEntryPoint function has been called
I/TA: [Log] TA_OpenSessionEntryPoint function has been called
I/TA: [Log] TA_InvokeCommandEntryPoint function has been called
I/TA: [Log] TA Square funtion has been called by CA
I/TA: [Log] Got value: 12 from NormalWorld
I/TA: [Log] Squared the value of normal world: 144
D/TC:? 0 tee_ta_close_session:380 tee_ta_close_session(0xe165ce0)
D/TC:? 0 tee_ta_close_session:399 Destroy session
I/TA: [Log] TA_CloseSessionEntryPoint function has been called
I/TA: [Log] TA_DestroyEntryPoint function has been called
D/TC:? 0 tee_ta_close_session:425 Destroy TA ctx
```

Figure 7: TEST APP running on Secure World

Fig. 6 and Fig. 7 show the test application running in the normal world and secure world. Since the debug mode was used to log each function, it was confirmed that the TEE Client API function was called with a CA and TA pair as shown in Fig. 5. In addition, OP-TEE project provides source code for testing TEE. OP-TEE sanity testsuite, called xtest, has 8 TAs for the kernel, internal APIs, client APIs, cryptographic tests, and includes 62 test commands.

4 CONCLUSIONS

ARM TrustZone, used in numerous mobile devices for trusted execution, have become important in embedded security. Attacks targeting TrustZone have also increased, and research on TrustZone is required[3]. In this paper, we showed how to use TEE Client API used to communicate with Trusted Application for TrustZone security study. In addition, we implemented a simple example of TEE Client API for communication between Normal World and Secure World in OP-TEE OS. We built a TrustZone environment on a virtual ARM v8 system using QEMU, but the OP-TEE project allows us to use the TrustZone environment on other real devices. As a future study, we will conduct a TrustZone security vulnerability study by building an ARM TrustZone environment on real devices.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government(MSIT) (NRF-2018R1A4A1025632)

REFERENCES

[1] Joakim Bech. 2014. OP-TEE, open-source security for the mass-market. (2014). https://www.linaro.org/blog/op-tee-open-source-security-mass-market
[2] Liang Cai. 2019. Guard Your Data with the Qualcomm® Snapdragon™ Mobile Platform. (2019).
[3] David Cerdeira, Nuno Santos, Pedro Fonseca, and Sandro Pinto. 2020. SoK: Understanding the prevailing security vulnerabilities in TrustZone-assisted TEE systems. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P), San Francisco, CA, USA*. 18–20.
[4] Samsung Developers. [n. d.]. SAMSUNG TEEGRIS. ([n. d.]). https://developer.samsung.com/teegris/overview.html
[5] GlobalPlatform. 2010. *GlobalPlatform Device Technology TEE Client API Specification Version 1.0.* Technical Report.
[6] ARM Holding. 2009. ARM Security Technology, Building a Secure System using TrustZone Technology. (2009).
[7] Skybox® Research Lab. 2019. *2019 vulnerability and threat trends.* Technical Report.
[8] Linaro. [n. d.]. Open Portable Trusted Execution Environment. ([n. d.]). https://www.op-tee.org/
[9] Linaro. 2020. OP-TEE Documentation. (2020). https://buildmedia.readthedocs.org/media/pdf/optee/latest/optee.pdf
[10] Global Platform. February 2011. The trusted execution environment: Delivering enhanced security at a lower cost to the mobile market. *White Paper* (February 2011).
[11] Samsung Electronics Co., Ltd. 2017. Whitepaper: Samsung Knox Security Solution. (2017).