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. Recently, various manufacturers have started using TrustZone technology. but existing commercial TEEs have limitations in conducting security research using TrustZone. Therefore, this paper introduces 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 also 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

1 INTRODUCTION

The advances in mobile technology and the spread of smartphones have made smartphones important in our lives. In particular, smartphone was stored sensitive data due to services such as Samsung Pay, Apple Pay, and Mobile Banking and was increased 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[1]. Thus, TEE(Trusted Execution Environment) technology has made to store and safely execute important data in mobile environments. TEE is a hardware security technology that isolates the execution of secure programs from the another of the program[2]. Embeded devices mostly use ARM TrustZone technology[3]. Samsung Knox which are used to secure Samsung Pay and Samsung smartphones is the representative TrustZone technology platforms[4].

ARM TrustZone has support for GlobalPlatform TEE Client API and TEE Internal Core API. TEE Client API is used to communicate with Trusted Application. However, recently attacks have appeared targeting Trusted Applications and security study is required[5]. The paper will help to demystify and 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 implement TrustZone environment using OP-TEE(Open Portable Trusted Execution Environment), an open source project[6]. The paper is composed as follows. Section 2 introduces ARM TrustZone and open source project OP-TEE for ARM TrustZone Implement. Senction 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.

2 BACKGROUND

2.1 Structure of ARM TrustZone

ARM Processor uses 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 into user mode and privileged mode according to Exception Level. However, since the two worlds do not run at the same time, a

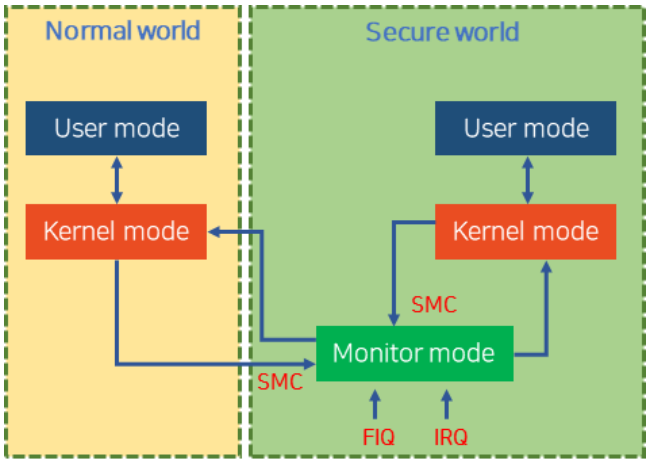

Figure 1: Stucture of ARM Trustzone[3]

SMA 2020, Jeju, Republic of Korea,

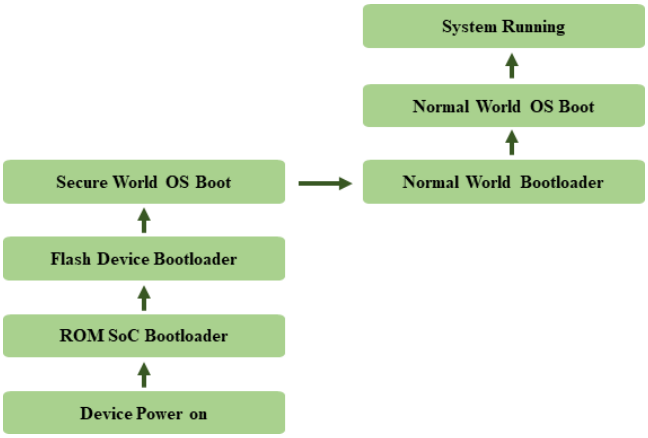

Figure 2: TrustZone Secure boot Process[3]

switch is required to change from the normal world to the secure world. Therefore, Monitor Mode was added in existing ARM operation modes(User Mode, Abort Mode, FIQ Mode, Undefined Mode, IRQ Mode, System Mode, SVC Mode). Monitor Mode operate only when SMC(Secure Monitor Call), IRQ, and FIQ occur. Therefore, Normal application running in Usermode is unable to switch between 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[3]. 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 is initiated by running the ROM SoC Bootloader to initialize peripherals such as memory controller. Next, boot the Secure World OS from the flash device and then boot the Normal World OS. Boot starts first in Secure World, allowing you to run a security check before Normal World's application has a chance to modify your system[3].

2.2 OP-TEE(Open Portable Trusted Execution Environment)

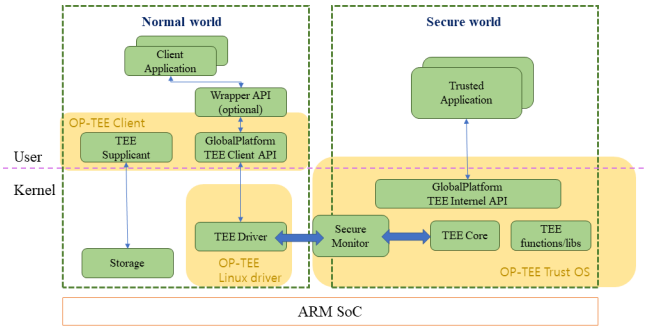

Figure 3: OP-TEE Software Architecture[7]

OP-TEE is an open source TEE that implements TrustZone technology. OP-TEE Project is managed and deployed by Linaro, with engineer, 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 GlobalPlatform TEE Internal API 1.0[8]. OP-TEE consists of three components, OP-TEE Client, OP-TEE Linux driver, and OP-TEE Trusted OS, as shown in Fig. 3. It is alsoensures 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[8].

3 COMMUNICATES WITH TRUSTED APPLICATION

3.1 TEE Client API

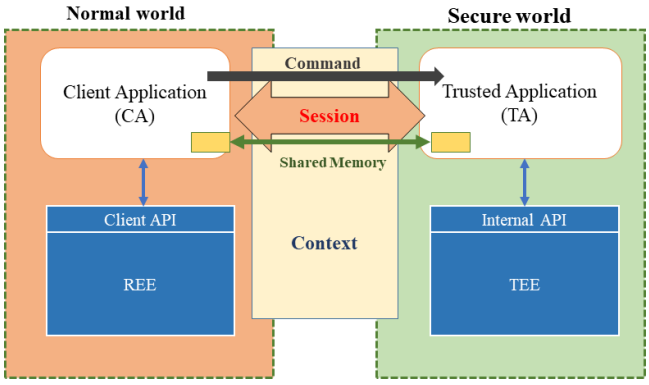

Figure 4: TEE Client API Concepts

ARM TrustZone supports TEE Client API to communicate with TEE Trusted Application in Secure World. GlobalPlatform 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[9]. REE is a modern operating system such as Windows, Linux, Mac OS, Android or iOS. TEE is an environment that can be executed separately from REE, such as QSEE[10], OP-TEE, Kinibi, Trustcore or TEEGRIS[11].

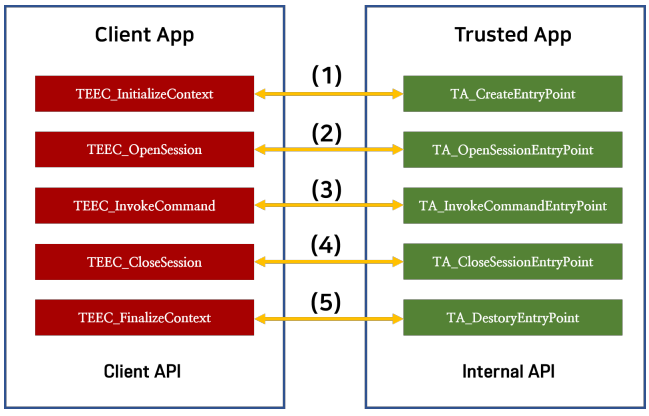

Figure 5: Communication process Using TEE Client API

Table 1: Data Types of Client API[9]

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

Table 2: CA Operation function of Client API[9]

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[9]

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

Generally, the application of TrustZone is divided into CA running in REE and TA running in TEE. CA and TA can not be connected directly because these run in a separate environment, but if it called TEE Client API and Internal API functions in pairs like Fig. 5., it can communicate as shown in Fig. 4[9].

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

ClientAPI 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[9].

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. Building QEMUv8 according to the instructions in the OP-TEE documentation[8]builds the ARMv8 64-bit TrustZone virtual environment and runs the QEMU emulator. After a successful build with no errors and the QEMU emulator running, jobs of Normal World and Secure World can be monitored into 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.

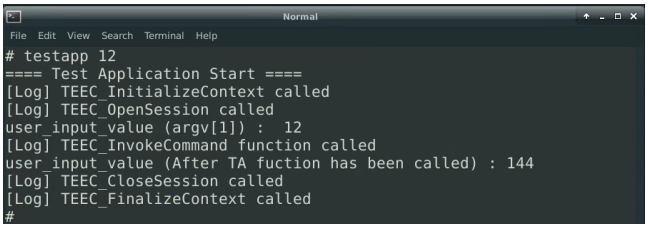

Figure 6: TEST APP running on Normal World

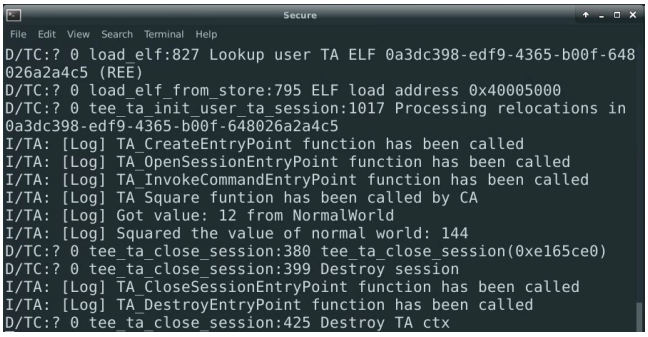

Figure 7: TEST APP running on Secure World

Fig. 6 and Fig. 7 shows 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[5]. In this paper, we introduced 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] Skybox® Research Lab. 2019. 2019 vulnerability and threat trends. Technical report.
[2] Global Platform. February 2011. The trusted execution environment: delivering enhanced security at a lower cost to the mobile market. *White Paper*.
[3] ARM Holding. 2009. Arm security technology, building a secure system using trustzone technology. (2009).
[4] Samsung Electronics Co., Ltd. 2017. Whitepaper: samsung knox security solution. (2017).
[5] 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.
[6] Linaro. [n. d.] Open portable trusted execution environment. (). https://www.op-tee.org/.
[7] 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.
[8] Linaro. 2020. Op-tee documentation. (2020). https://buildmedia.readthedocs.org/media/pdf/optee/latest/optee.pdf.
[9] GlobalPlatform. 2010. GlobalPlatform Device Technology TEE Client API Specification Version 1.0. Technical report.
[10] Liang Cai. 2019. Guard your data with the qualcomm® snapdragon™ mobile platform. (2019).
[11] Samsung Developers. [n. d.] Samsung teegris. . https://developer.samsung.com/teegris/overview.html.