D3.1
TEE Security Study

Project number:	883156
Project acronym:	EXFILES
Project title:	Extract Forensic Information for LEAs from Encrypted SmartPhones
Project Start Date:	1st July 2020
Duration:	36 months
Programme:	H2020-SU-SEC-2019, Technologies to enhance the fight against crime and terrorism
Deliverable Type:	Report
Reference Number:	SU-FCT02-883156 / D3.1 / V1.0
Workpackage:	WP 3
Due Date:	December 2020 – M06
Actual Submission Date:	15th January 2021
Responsible Organisation:	Royal Holloway, University of London (RHUL)
Editor:	Carlton Shepherd (RHUL)
Dissemination Level:	Public
Revision:	V1.0
Abstract:	This report describes mobile TEEs and their common security mechanisms. An open-source, white-box testbed is then presented for conducting security research on TEE-equipped mobile systems.
Keywords:	Trusted Execution Environments, Mobile Devices

The project EXFILES has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 883156.

Editor

Carlton Shepherd (RHUL)

Contributors (ordered according to beneficiary numbers)

Carlton Shepherd (RHUL)
Konstantinos Markantonakis (RHUL)
Technikon
IRCGN
BKA
CNI
CSIC
NFI
RISCURE
Cyber I
NCIS
Synacktiv

Disclaimer

Executive Summary

The security mechanisms deployed on modern mobile handsets significantly hinder forensic analysts seeking to recover device data. This has particular importance to law enforcement agencies (LEAs) where valuable evidence is protected by such mechanisms that could be used to facilitate criminal investigations. On today's devices, secure boot sequences prevent the introduction of unauthorised firmware, while TEEs provide hardware-assisted isolation of cryptographic keys and applications that may be used to conceal data of interest.

This report details a white-box model for LEAs and forensic analysts to develop forensic extraction methods for TEE-equipped mobile devices. The developed platform supports a variety of common mechanisms, including secure boot, a TEE based on the ARM TrustZone architecture, and encrypted secure storage. The platform conforms to the GlobalPlatform TEE specifications that govern today's mobile device TEEs. Moreover, the platform is built from open-source components that can be readily inspected, modified and redeployed in order to explore potential weaknesses and prototype extraction techniques. At its core, the platform uses the OP-TEE framework originally developed by ST-Ericsson and maintained by the Trusted Firmware project, comprising major industry players, such as ARM, Google, NXP, and STMicroelectronics. Our testbed platform is deployed on a development board based on the ARM architecture, used by most of today's mobile devices, and specifically uses a system-on-chip used by several Huawei smartphone models.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Mobile digital forensics is concerned with the recovery of device data from mobile devices, such as smartphones and tablets, to assist in forensic investigations. However, data extraction methods have substantially increased in difficulty due to the growing complexity of mobile platforms. In today's world, mobile devices contain a range of security mechanisms based on a combination of cryptographic techniques, e.g., encryption and digital signature verification, and hardware-enforced access control. These mechanisms continue to significantly impede the efforts of law enforcement agencies (LEA) when attempting to recover valuable evidence for serious criminal investigations [1, 18, 73].

A significant hindrance has been the widespread deployment of trusted execution environments (TEEs) that provide hardware-assisted isolation of data and applications. TEEs underpin a range of security-related mobile services, including cryptographic key management, full-disk encryption, user authentication, and encrypted secure storage of arbitrary data objects. TEEs aim to defend against privileged software adversaries in the native operating system (OS). Consequently, even gaining root access to the native OS cannot subvert these security mechanisms, which can be used to conceal evidence from LEAs.

This report presents a white-box platform to assist with the research and development of forensic extraction techniques on today's mobile devices. The platform, which closely replicates a modern mobile architecture and its security mechanisms, is built to facilitate research on TEEs and its applications. The software components of our platform are open-source: they can be inspected, modified and redeployed to analyse weaknesses in security mechanisms and develop new extraction techniques. The platform is built with ARM-based system-on-chips (SoCs) in mind that, as established in the D1.1 market study [21], are used ubiquitously by the major smartphone vendors at the time of writing. It implements a variety of common security mechanisms, including authenticated secure boot, an ARM TrustZone-based TEE using an open-source security kernel from the OP-TEE project, and TEE-backed secure storage and key management. Moreover, the test-bed conforms to the GlobalPlatform specifications that govern the use of device TEEs used by original equipment manufacturers (OEMs). From a firmware perspective, the open-source Trusted Firmware project is used—a widely used reference implementation of the secure boot chain and the TEE secure monitor, which is developed by ARM. We have deployed the platform on an ARM development board that replicates a modern mobile platform and uses a SoC found on several Huawei smartphone models.

1.2 Report Organisation

The coming chapters describe ARM-based TEEs and their specifications, before presenting technical details regarding the features and limitations of the proposed research platform. Specifically, Chapter 2 provides preliminary information about system-on-chips, TEEs, their governing specifications and initialisation through an authenticated secure boot procedure. Chapter 3 then examines common security mechanisms found on modern mobile devices, including those that are enhanced by TEEs, such as the Android Keystore, remote attestation, and full-disk encryption. Following this, Chapter 4 presents the design and development of a white-box model for conducting research and prototyping forensic extraction techniques against a platform that closely resembles a modern mobile device. Finally, Chapter 5 summarises and concludes this report.

Chapter 2

Mobile Trusted Execution Environments

This chapter introduces mobile TEEs in greater technical detail, their governing specifications and typical platform configurations. It begins with a recap of system-on-chips (SoCs) first introduced in D1.1 (§3.3, [21]), before discussing ARM TrustZone, the GlobalPlatform specifications, and leading proprietary TEE-based systems developed by major mobile OEMs.

2.1 System-on-Chips (SoCs): An Overview

A system-on-chip (SoC) is an integrated circuit comprising most components of a computing system, such as the multiple application processor cores, memory units (RAM, ROM, and flash memory), system buses, timers, analog-to-digital and digital-to-analog converters (ADCs and DACs), and integrated graphics processing units (GPUs). Because of their integrated nature, current mobile devices are underpinned by SoCs because of their power efficiency and small physical footprint. In general, a SoC's precise components and capabilities varies between vendors. At the time of writing, leading SoC vendors include Apple, Samsung, Qualcomm, MediaTek, Broadcom, Texas Instruments, and Huawei. The reader is referred to Table 9 in D1.1 [21] for a comprehensive list of SoCs used by modern mobile devices.

The hardware design of modern mobile device SoCs is based heavily around the configuration of reusable semiconductor building blocks. These blocks, known formally as intellectual propriety (IP) blocks or IP cores, are built directly by the silicon vendor and/or licensed from a third party, such as ARM. By selecting and configuring particular IP blocks, certain features can be provided by the SoC, such as high-performance video processors, security-aware memory controllers, and cryptographic co-processors. The configuration and implementation details of modern mobile SoC IP cores, e.g. Verilog or VHDL code, lies primarily outside the public domain. The closed nature of IP cores and modern hardware design in general has catalysed open-source movements, such as the RISC-V initiative [59]. However, mobile devices with open-source hardware are not readily available on the mass market.

2.2 ARM TrustZone

ARM TrustZone (TZ) is a set of security extensions that are built from a suite of hardware IP cores and firmware components. TrustZone is the foundation for building TEEs on ARM-based SoCs.

2.2.1 High-Level Overview

TrustZone divides system execution into two worlds: the 'secure' and 'non-secure' worlds[1] that host security-sensitive and non-sensitive applications and services respectively. Fundamentally, the ARM architecture uses a 'non-secure' (NS) bit as a control signal to denote the current world of execution. Each logical core of modern ARM Cortex-A application processors (APs) is associated with two virtual cores for secure and non-secure world execution, which operate in a time-sliced fashion.

During the execution of a non-secure program, A, which communicates with a secure world application, B, the NS bit will be changed accordingly when transitions are made from A to B and vice-versa. The NS bit is propagated through the SoC bus transactions to other components, such as memory controllers, which can behave differently depending on the state of the bit. A TrustZone-enabled SoC is thus aware of any unauthorised accesses of secure world-only resources from non-secure world programs. When detected, such accesses are prevented by returning the appropriate error code depending on the target component. Components that behave differently between non-secure and secure worlds of execution are known as being 'TrustZone-aware'. This is shown in Figure 2.1.

Overall, the secure creation and maintenance of a TrustZone system is underpinned by the physical and software configuration of the SoC bus components. A TrustZone-enabled SoC contains a multitude of IP blocks for enforcing access control to secure world-only peripherals, security co-processors, static (SRAM) and dynamic RAM (DRAM) modules, and more. A sample ARM-based SoC is shown in Figure 2.2.

2.2.2 ARM Exception Model and Secure Monitor Mode

The ARMv8-A architecture defines three software privilege exception levels for standard, non-secure world use: EL0, EL1, and EL2 in order of increasing privilege, described as follows:

- **EL0**: user applications executing in the non-trusted world, equivalent to user mode on X86-64 systems.
- **EL1**: higher privilege services within the non-trusted world, e.g. native operating system, equivalent to kernel mode.
- **EL2**: hypervisor mode for managing virtualised operating systems.

Programs executing at lower levels are not able to access higher-level applications and services except via tightly controlled interfaces, e.g., the system call interface (SCI) between EL0 and EL1. This concept is expanded by ARM TrustZone to provide three additional exception levels—S-EL0, S-EL1, and EL3—for use with the secure world. These are described below in increasing order of privilege for ARMv8.3 and earlier AP cores:

- **S-EL0**: the lowest privilege level for applications executing in the secure world.
- **S-EL1**: secure world equivalent of kernel mode; the TEE OS kernel operates here.
- **EL3**: secure monitor mode for mediating the switches between the worlds of execution.

[1] Also known as 'trusted' and 'untrusted' worlds.

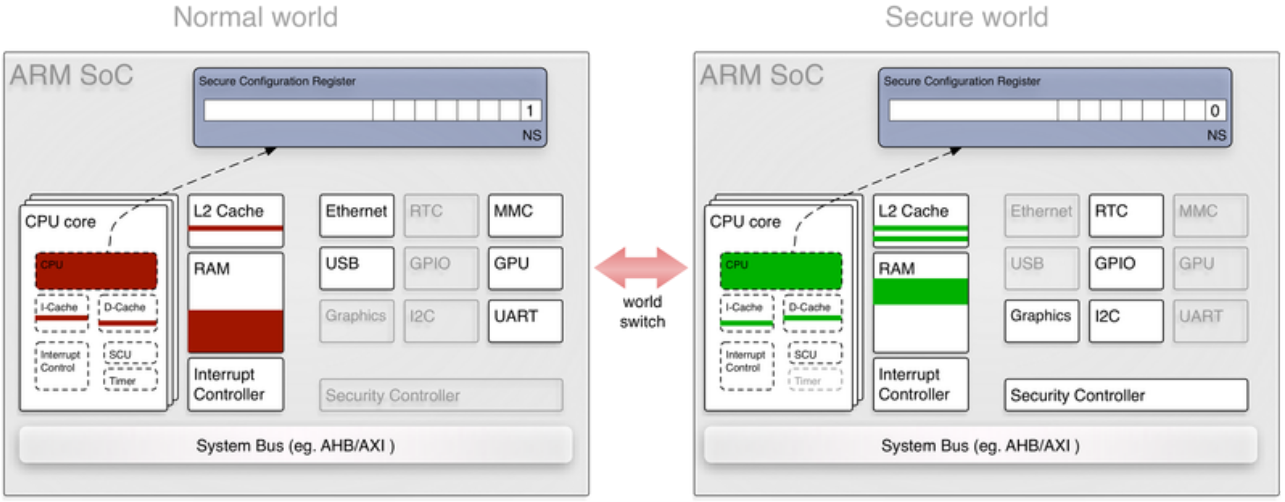

Figure 2.1: A TrustZone-enabled ARM SoC in non-secure and secure world modes [22].

The cornerstone of the TrustZone security model is the use of the latter exception mode—secure monitor mode at EL3—to manage world context switches for ARM Cortex-A APs[2]. Monitor mode is entered using exception-generating secure monitor call (`smc`) instructions, which are called from the non-secure or secure world kernels at EL1 and S-EL1 respectively. The secure monitor performs the saving and restoration of secure and destination world contexts, such as register states, and writes the NS bit.

For ARM AP IP cores using ARMv8.4 and beyond (2017—today), a further secure world exception level (S-EL2) was added to provide the ability for multiple TEE OSs to be hosted in an isolated fashion. The full exception model for ARMv8.4-A APs is shown in Figure 2.3. A reference implementation of the secure monitor is provided as part of the Trusted Firmware project [43], whose membership includes leading device manufacturers and SoC vendors, such as Google, ARM, STMicroelectronics, NXP, Cypress (now part of Infineon), and Texas Instruments. The project aims to provide SoC vendors and OEMs with a trusted, open-source and ARM-compliant code base upon which secure world services can be built[3].

2.2.3 TrustZone Protection Controllers

Several mechanisms are used to protect off-CPU memory and peripheral controllers used by the secure world from non-secure world accesses. These controllers can partition RAM into secure and non-secure regions, effectively serving as a memory firewall, and prevent unauthorised non-secure world accesses to peripherals intended only for the secure world. Zero, one, or more of these units might be used depending on the vendor's security goals.

[2]TrustZone is also available for microcontrollers, known as TrustZone-M. However, this architecture is substantially different to TrustZone found on modern smartphone SoCs, which is the focus of this work. The reader is referred to [53] and [63] for a comprehensive survey of TrustZone-A and -M, and TEEs generally.

[3]The Trusted Firmware project serves as a *reference* code base, which will serve as only part of the TEE firmware deployed on SoCs in the wild. Manufacturers may develop proprietary firmware in addition to this code base.

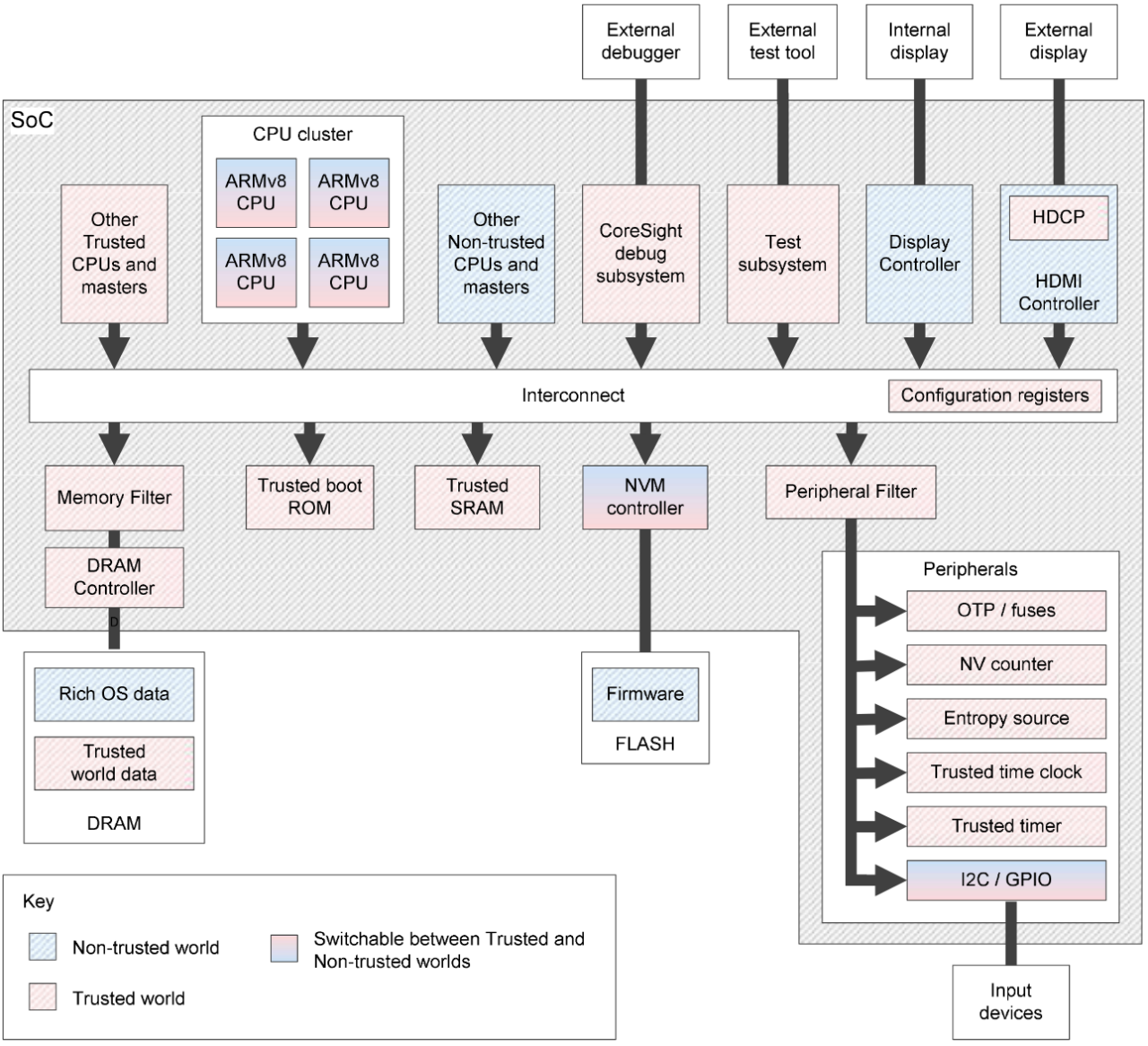

Figure 2.2: A sample ARM-based system-on-chip with TrustZone awareness [11]. The trusted world components are accessible only to the secure world, i.e. when the NS-bit=0, while non-trusted world components can be accessed by the non-secure world. Certain components can be configured to provide access to both worlds, but data is only returned to the calling world based on the NS bit that are propagated in bus transactions over the SoC interconnect.

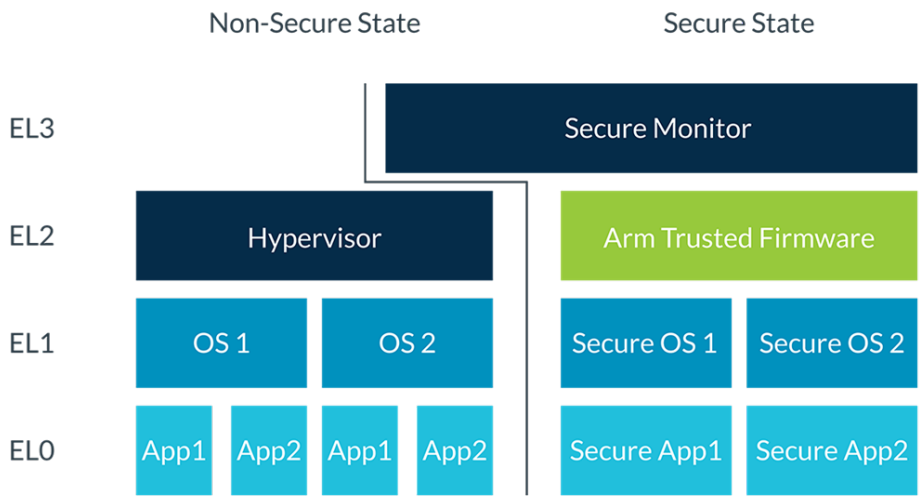

Figure 2.3: ARM exception model for v8.4-A and beyond [10].

The TrustZone Address Space Controller (TZASC) is a programmable memory security unit, also known as a memory firewall, for partitioning DRAM into secure and non-secure memory address spaces. During execution, the TZASC prevents bus transactions with NS-bit=1 from accessing secure world-only address regions. Secondly, the TrustZone Peripheral Controller (TZPC) is used to assign protection bits to peripherals, such as those connected over UART or GPIO, to mark them as being accessible to only the secure world, non-secure world, or both. This unit is used with the ARM AXI-to-APB bus bridge to deny illegal accesses to peripherals intended for the secure world when the NS bit is set (NS-bit=1). Lastly, the TrustZone Memory Adapter (TZMA) is a configurable block for protecting static memory modules, such as secure SRAM and ROM, intended for the secure world from unauthorised non-secure accesses using the NS bit.

For on-CPU protection, ARM Cortex-A APs provide two virtual MMUs, each for the secure and non-secure worlds. This allows each world's kernel to maintain local translation tables for virtual-to-physical address mappings. The ARM L1 processor cache (the lowest) is extended to tag each cache line as belonging to the non-secure or secure world. Translation look-aside buffer (TLB) entries are also tagged similarly. Accesses to the cache lines or TLB entries are accepted or denied based on the current world execution mode [13].

2.2.4 Secure Boot

Securely initialising the secure world and its requisite on-chip units is key to maintaining the security of a TrustZone-enabled SoC. The aim of the secure boot process is to ensure the authenticity of bootloader components when the device is first powered. This way, only authenticated components, which are used to configure TrustZone-aware units and load the TEE kernel and its applications, are loaded and used by the device.

The ARM secure boot procedure is underpinned by a hardware-based root of trust (RoT) that is inherently considered to be trusted. At the heart of a RoT are the security functions it supports; these functions provide the facility while offering a strong protection of an SoC's product lifecycle management during all operation phases, such as power off, power up, run time operations and communications with external entities. The RoT supports functions as secure monitoring, secure

validation/authentication, storage protection, secure communication and key management. The RoT is typically implemented at the SoC level in read-only memory, which bootstraps the boot process [44]. Modern boot sequences comprise loading a number of boot loading stages: after receiving a reset signal, the SoC uses the boot ROM to authenticate and the other boot loader stages in a chain of trust. These stages, shown in Figure 2.4, are as follows:

1. **Boot stage 1 (BL1)**: resides in read-only memory, performs basic on-board processes, such as power-on self-test (POST) and control register setup, and loads the second-level image (BL2).
2. **Boot stage 2 (BL2)**: used to configure non-volatile storage, S-EL1 page translation, and loads all third-level boot images (BL3).
3. **Boot stage 3-1 (BL3-1)**: loads the EL3 and secure monitor firmware, including interrupt handlers, power management, TrustZone memory controllers, and the SMC interface for world switching. Executes at EL3.
4. **Boot stage 3-2 (BL3-2)**: loads and configures the S-EL1 payload including the secure world kernel/OS. Executes at S-EL1.
5. **Boot stage 3-3 (BL3-3)**: loads and passes control to the non-trusted firmware image, e.g., UEFI, at EL1.

On mobile devices, the bootloaders are signed by the OEM. During the boot sequence, each bootloader veries the subsequent loader by measuring it, i.e., calculating its SHA-256, and verifying this against an OEM-signed measurement using the OEM's public key. This begins with the boot ROM (BL1), which measures and authenticates BL2, and so on until control is finally passed to the untrusted world kernel. As such, any unauthorised bootloaders will fail signature verification and the component will not be loaded. This persists while the offending component exists on the device, manifesting itself as a device bootloop when it is powered on. The authenticated boot procedure is depicted in Figure 2.5.

A reference implementation of the secure boot process is provided by the Trusted Firmware project [43], which vendors can tailor to particular SoCs and devices.

2.2.5 World Setups and Inter-World Communication

Following the boot process, each world is configured with its own operating system and an independent set of applications. The non-secure OS is often called the *rich* or *untrusted* OS, while the secure OS is sometimes known as the *trusted OS*. The most common rich OSs for smartphones, which execute at EL1, are Android [7] and Apple iOS [8], as discussed in D1.1 [21]. Secure world OSs, which execute at S-EL1, are specialist security kernels with a reduced footprint in order to limit the scope for potential vulnerabilities. These hardened kernels are evaluated using the Common Criteria framework against the GlobalPlatform TEE protection profile, described next in §2.3. Commercially available secure world OSs include Trustonic Kinibi [71], Huawei iTrustee [34], Qualcomm QTEE [56], and Samsung TEEGRIS [61]. Access to secure world OSs is, for the most part, closed source (see §4.1). Access to documentation, SDKs, and third-party application development is typically limited to authorised personnel with an established business relationship.

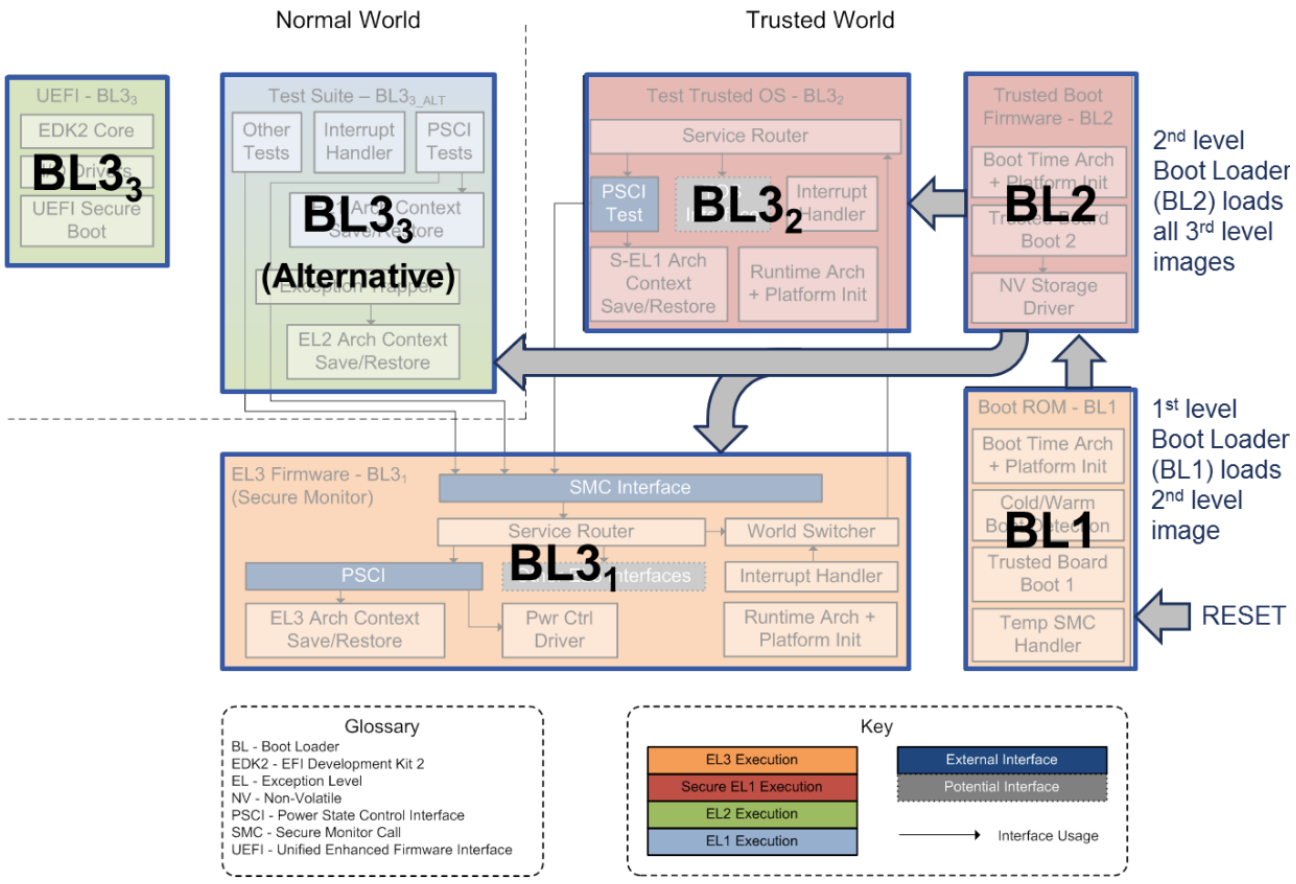

Figure 2.4: High-level ARM boot process [11].

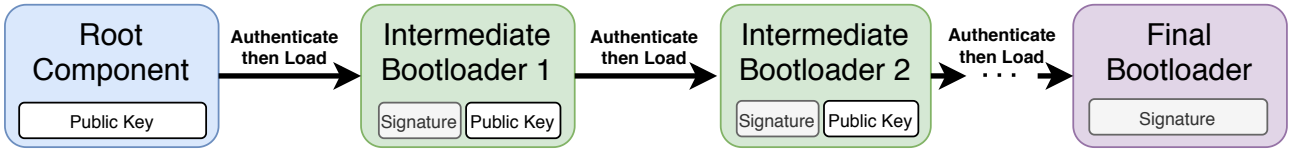

Figure 2.5: Authenticated secure boot process.

The secure world OS hosts a number of trusted applications (TAs) that execute at the lowest security exception level (S-EL0). These are used for implementing security-critical services, some of which can be used by the non-secure OS and its applications. Common services provided by TAs on today's mobile devices are biometric authentication and digital rights management (DRM) enforcement, such as WideVine and PlayReady for video playback protection [42].

Non-secure applications (EL0) communicate with TAs through a system library, typically provided by the secure OS provider. This library communicates to a corresponding secure OS driver in the non-secure OS that enters secure monitor mode over the SMC interface. If the SMC call is permissible, then the secure monitor firmware passes control to the secure OS that routes the request to the intended TA. Control flow is returned to non-secure world applications using the reverse process. This process is depicted in Figure 2.6.

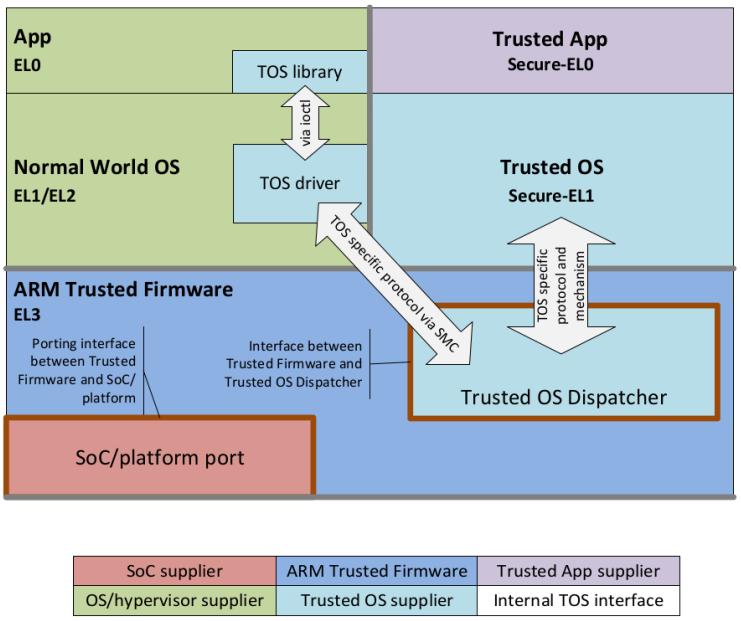

Figure 2.6: High-level inter-world communication flow for TrustZone-based systems [9].

2.3 GlobalPlatform TEE

The GlobalPlatform TEE (GP TEE) is a family of specifications that governs the use and management of TEEs, including their system architecture [27]; communication interfaces between the non-secure and secure worlds [24]; interfaces between TAs and secure OS [28]; APIs for communication between the TEE and networked devices [26]; the use of external security hardware, such as secure elements [25]; and the protection scope that a compliant TEE must meet [30].

It is important to note that the GP TEE is a suite of security design specifications. For example, it does not contain precise details for implementing rich OS (REE) and TEE partitioning using any particular vendors or technologies, although TrustZone is currently the predominant method for instantiating a GlobalPlatform TEE [67]. The salient specifications are briefly described as follows; the reader is referred to the GlobalPlatform TEE Technology Document Library for accessing the specifications in full [29].

- **GP System Architecture**: defines the high-level system architecture for a device that hosts a GP-compliant TEE. It specifies its high-level hardware architecture and its security requirements, roots of trust, multiple TEE architectures on the same device, and the scope of each specification in the GlobalPlatform TEE family.
- **GP Internal Core API**: defines the interfaces and function definitions between trusted applications (TAs) and the TEE. It covers function definitions and return types for TAs to access secure storage functionality, arithmetic and cryptographic operations, and sources of time.
- **GP Client API**: specifies the interfaces and function definitions by which non-secure world applications should communicate with the TEE. It specifies data types and function definitions for resource initialisation, TA connection interfaces, multi-threading, and establishing shared secure memory regions.

- **GP Protection Profile**: used for assessing GP TEE-based targets of evaluation (TOE) under the Common Criteria framework.
- **GP Sockets API**: specifies how TEEs open socket connections directly from the secure world over UDP/IP and TCP/IP including TLS. The APIs define return types, function definitions, and supported algorithms for secure socket connections.
- **GP TA Debug API**: defines data types, function interfaces and data structures for debugging TAs, including post-mortem reporting and debug log messages.
- **GP TEE Management Framework**: defines mechanisms, such as security protection domains, by which the TEE and TAs can be securely hosted and managed by distinct providers and stakeholders.

2.3.1 Architecture

Like TrustZone, the GP TEE divides execution into trusted and untrusted worlds. The trusted world may host one or more TEEs, which themselves host an independent set of secure OSs and TAs. The untrusted world hosts a non-secure operating system, e.g., as Android, with its own set of untrusted applications. TAs communicate with the TEE OS using the GP Internal API, while two methods exist for enabling data transfers between a TEE TA and a REE client application. The first method is using the GP Client API, which defines interfaces for a REE application to access a TEE TA by passing messages over a hardware-assisted secure monitor, such as the ARM TrustZone secure monitor interface. The secure world kernel then securely routes the messages to the target TA. Secondly, TAs may establish shared memory buffers directly with a REE application.

The specifications stipulate that the TEE must be protected by a secure boot process, as described in §2.2.4, where the TEE image and its bootloaders are authenticated starting with an immutable root of trust, such as SoC ROM. Moreover, the TEE must implement trusted storage with device binding, where TEE assets can be stored securely from an untrusted REE, and cannot be accessible if the TEE is migrated to another device. This protection method must be *"at least equal to that of the TEE environment"* [27]. The specifications provide support for encrypting TEE data to an untrusted filesystem using authenticated encryption under a TEE file encryption key derived from a hardware unique key. It also supports the use of an external secure element that only the TEE can access in order to store data, e.g., cryptographic keys. The high-level architecture of the GP TEE is shown in Figure 2.7.

2.3.2 Threat Model and Protection Scope Overview

The GP TEE aims to protect against sophisticated software adversaries that originate from the REE. The GP TEE Protection Profile (GP TEE PP), which is used for a target of evaluation(TOE) based on the GP TEE *"targets threats to the TEE assets that arise during the end-usage phase and can be achieved by software means...focuses on non destructive software attacks that can be easily widespread...and constitute a privileged vector for getting undue access to TEE assets without damaging the device itself"* [30].

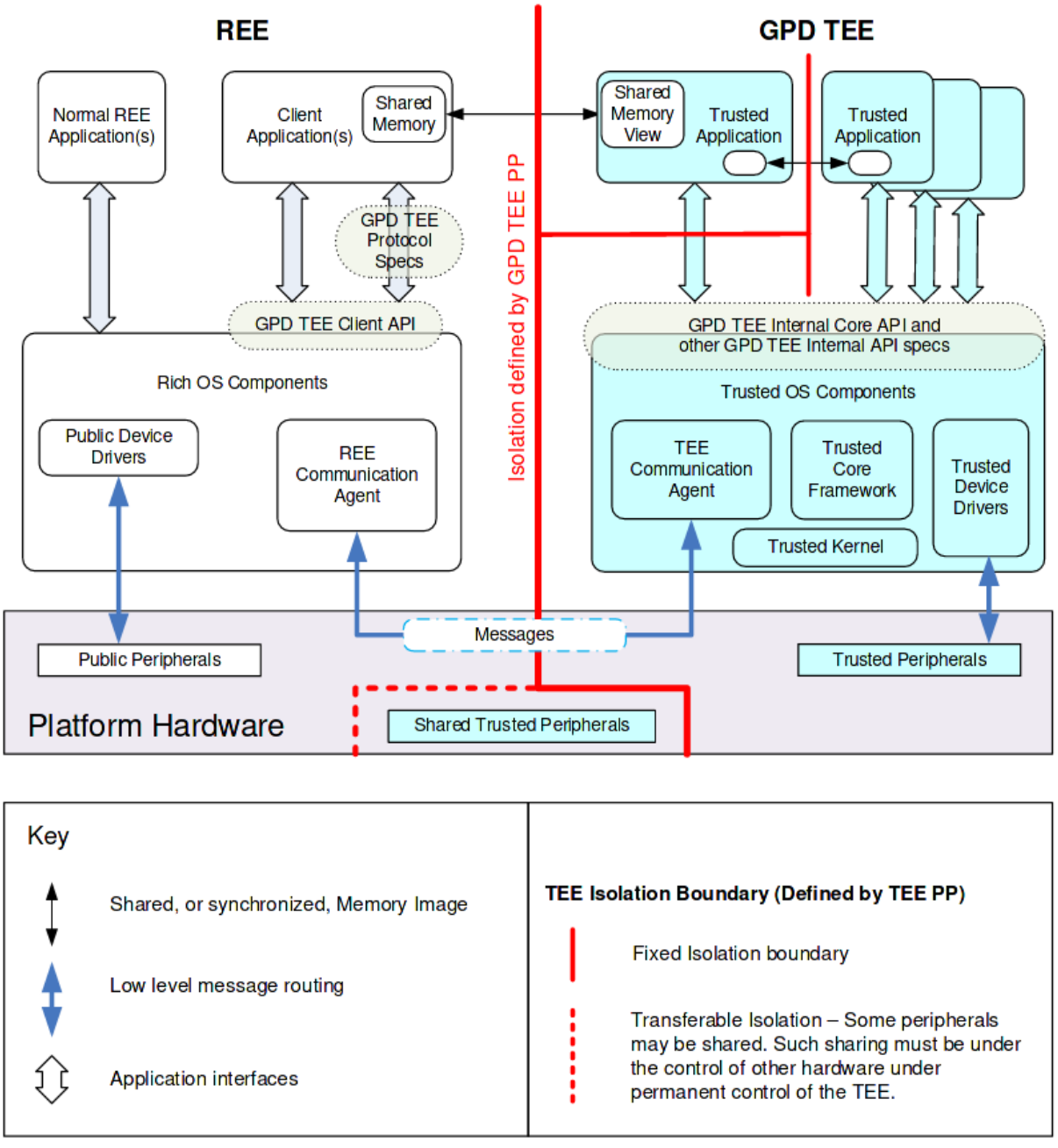

Figure 2.7: System architecture of a REE and GP-defined TEE (GPD TEE) [27].

In hardware, a GP TEE should have access to a secure clock, cryptographic accelerators, and volatile and non-volatile memory. The GP TEE must be able to access resources without trusting the REE. More generally, the GP TEE should be self-contained and be dependent on any software or firmware components in the non-secure world for secure use. In software terms, the root(s) of trust, including the initial bootloader code; the TEE kernel; and the TAs are considered trusted. Errors in any of these components, such as the functions TAs expose to non-secure client applications, may compromise the intended services that the TEE and its TAs aims to provide.

The GP TEE PP provides an explicit list of GP TEE assets and the security properties that should be upheld. The security matrix of requirements and TEE assets is reproduced from [30] and [63] in Table 2.1. The GP TEE PP defines an on-device and remote attacker that should be defended against as a minimum requirement. In general, the TEE protects against low-cost, low-expertise physical attacks; the protections *"will be at a lower level than that provided to dedicated tamper resistant technology"* [27]. The defined attacker models are reproduced as follows:

- **Basic remote attacker**: *"Performs the attack on a remotely-controlled device or alternatively makes a downloadable tool that is very convenient to end-users. The attacker retrieves details of the vulnerability identified in the identification phase and [...] makes a remote tool or malware and uses techniques such as phishing to have it downloaded and executed by a victim [in the untrusted world]."*
- **Basic on-device attacker**: *"Has physical access to the target device; it is the end-user or someone on his behalf. The attacker is able to retrieve exploit code, guidelines written on the internet on how to perform the attack, and downloads and uses tools to jailbreak/root/reflash the device in order to get privileged access to the REE allowing the execution of the exploit. The attacker may be a layman or have some level of expertise but the attacks do not require any specific equipment."*

At present, the minimum assurance level for GP TEE compliance is CC EAL2+. This is a lower level of assurance than other secure execution platforms, such as smart cards and secure elements. These latter platforms are typically evaluated to CC EAL4+ and against expert adversaries possessing specialist testing equipment.

Table 2.1: Minimum security requirements of GP TEE assets, from [30, 63].

	Property									
Asset	**C**	**I**	**AU**	**U**	**UP**	**AT**	**DB**	**M**	**CO**	**IM**
TEE Identifier				✓						✓
RNG					✓					
TA Code			✓						✓	
TA Data and Keys	✓	✓	✓			✓	✓		✓	
TA Instance Time								✓		
TA Run-time Data	✓	✓							✓	
TA Persistent Data	✓	✓	✓				✓		✓	
TEE Firmware		✓	✓							
TEE Init. Code and Data		✓								
TEE Storage RoT	✓	✓								
TA Persistent Time								✓		
Rollback Detection Data		✓								
TEE Debug Auth. Key	✓	✓								

C: Confidentiality, **I**: Integrity, **AU**: Authenticity, **U**: Uniqueness, **UP**: Unpredictability, **AT**: Atomicity, **DB**: Device Binding, **M**: Monotonicity, **CO**: Consistency, **IM**: Immutability.

2.4 Proprietary Systems

Proprietary TEE systems have been developed by some OEMs that offer orthogonal services to the GlobalPlatform TEE. Two common systems found on Samsung and Apple devices are Samsung KNOX and the Apple Secure Enclave Processor (SEP) respectively.

2.4.1 Apple Secure Enclave Processor (SEP)

Apple's Secure Enclave Processor (SEP) is a security co-processor on Apple devices. Very few publicly available details exist about the SEP besides high-level features provided in Apple marketing materials. However, some reverse engineering methods have been made by independent security researchers [45]. The SEP shares similarities with ARM TrustZone and is directly initialised during the secure boot process by the boot ROM, the device's root-of-trust. It provides access to hardware-backed cryptographic operations, such as encryption and signing services. The SEP uses memory encryption between internal memory and external RAM to protect data in transit between itself and and main memory. It can also access its own set of peripherals, a bank of one-time programmable ROM (eFuses), and widely used input/output interfaces, such as GPIO and SPI. The SEP has been shown to support iOS key management and the Apple iOS biometric authentication systems, including Touch ID and Face ID.

2.4.2 Samsung KNOX

Samsung KNOX is a security platform found on leading Samsung handsets built from a TrustZone-based TEE. The platform provides containers, known as KNOX Workspaces, that are encrypted using keys held in the TEE. The workspaces provide secure locations in which documents,

videos, photographs and other data can be stored and encrypted automatically. As such, data cannot be extracted from these workspaces without co-operation from the TEE to decrypt them. At boot time, Samsung KNOX extends the secure boot process on Samsung devices to set an eFuse known as the 'warranty bit' if any unauthorised boot components are loaded. If the eFuse is set, new KNOX containers cannot be created and existing ones cannot be decrypted. The state of this eFuse is permanent and persists across reboots and resets [20]. Another additional security mechanism is the use of remote attestation to allow remote entities to validate the state of the target device. That is, the remote verifier can learn, using a secure channel, if the device has been booted with any unauthorised bootloaders by inspecting the state of the eFuse. These attestation messages are signed using a device-specific attestation key accessible only to the TEE, the public key of which is used by the remote verifier to authenticate the response messages [20].

Chapter 3

Security Mechanisms Using Mobile TEEs

This chapter summarises common device security mechanisms that are provided wholly, or enhanced in part, by mobile trusted execution environments.

3.1 Trusted Boot

Trusted boot is an adaptation of secure boot (see §2.2.4) that enables services to be aware of the platform's configuration state. Trusted boot is the implementation which makes use of a hardware module to verify the boot sequence. During the launch process, boot components are measured using SHA-2 or another cryptographically secure one-way hash function and stored securely, as shown in Figure 3.1. Alternatively, the device may securely set an eFuse when unauthorised boot components are detected, as with the Samsung KNOX platform. Either way, both the measuring and storing procedures relating to platform measurement values are performed using the TEE. The TEE can then use these values at run-time to limit access to sensitive assets, such as cryptographic keys. The deviation of measurements from a set of known values indicates the presence of unauthorised boot components that could undermine the security of the TEE.

On non-mobile devices, trusted boot is often realised using a Trusted Platform Module (TPM). Here, the TPM measures each stage in the boot sequence using a hash function, where it is appended to the previously measured hash value. This creates a hash chain using the Platform Configuration Registers (PCRs), which can be used for various purposes. For example, it can be used to decrypt data only when the machine reaches a specific stage in the boot sequence (sealing) or to verify that the system is in a state that is trusted (remote attestation) [31].

3.2 Remote Attestation

Remote attestation dates back to the security abstractions provided by the Trusted Platform Module (TPM) specified by the Trusted Computing Group (TCG) [37]. It is a challenge-response protocol between a remote verifying authority and a proving device, such as a mobile handset. The function of RA is for the device to authenticate its hardware and software configuration to a remote verifier, which may control access to sensitive assets based on that configuration. Traditionally, these hardware and software measurements are taken at boot time as part of a trusted boot sequence; however, some non-mobile TEE platforms support taking on-demand measurements of trusted applications as part of the device's run-time environment.

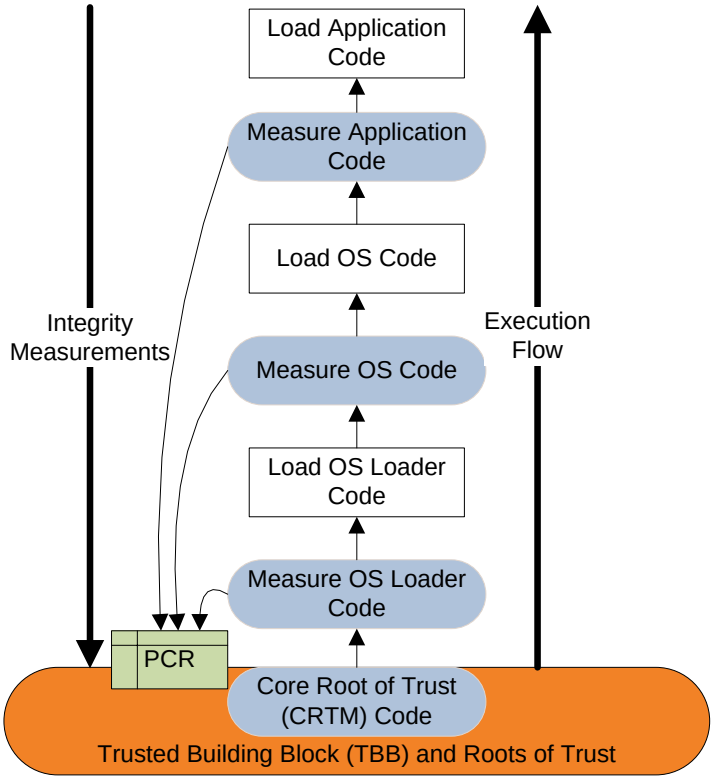

Figure 3.1: Generic trusted boot measurement procedure [67].

Whether the measurements are taken at run-time or boot-time, a critical step is that the attestation response is signed by a device-specific signing key whose public key is known to the verifier. The attestation signing key can be provisioned, or 'burned', into eFuses by the manufacturer or provisioned over a secure channel after the TEE has first launched. It is also critical that the TEE measures the hardware and/or software components of interest and signs the responses, also known as attestation 'quotes'. After returning the quote, the remote party verifies its signature using the attestation public key and inspects the measurements for any deviations from the expected values.

While not formalised as part of the GlobalPlatform TEE specifications, remote attestation protocols have been widely deployed by a small number of mobile and non-mobile vendors. Samsung KNOX supports remote attestation to assist with enterprise bring your own device (BYOD) policies. The KNOX platform provides a mobile device manager (MDM) for enterprise system administrators to assess the trustworthiness of employee handsets. The MDM can perform remote attestation with an enrolled device; the device then signs and returns the bootloader measurements, and the state of the warranty bit using a device-specific and TEE-bound attestation. The attestation responses are authenticated at the MDM's backend using the Samsung Attestation Server, which possesses the corresponding public keys [60].

Remote attestation is also used by Intel SGX on Intel-based X86-64 systems. Each Intel CPU contains a hardware-bound attestation key that is used to sign quotes containing the measurement of SGX enclave applications [19]. The signed quote is then transmitted to the remote verifying authority over a secure channel. The SGX attestation protocol uses a group signature scheme, which uses a single public key to verify all private keys such that users' individual CPU signing keys cannot be identified. Remote attestation is also used to verify the target enclave's

state in the EPID protocol, first proposed by Brickell et al. [16]. After this, EPID can bootstrap a secure channel to allow the provisioning of secrets into enclaves after their deployment.

3.3 Key Management

Mobile TEEs can provide secure persistent storage of sensitive material. At a specification level, the GlobalPlatform TEE provide functions via the Internal Core API [28] for device data 'binding'. This generates encryption keys from a hardware unique key (HUK) using a key derivation function (KDF) within the TEE, which are used to encrypt TA and TEE key material and data. TAs can also use these keys to encrypt non-secure world data if needed. The keys must never leave the TEE and, consequently, encrypted material cannot be decrypted on another device. The GlobalPlatform TEE specifications support the storage of keys and data objects to external security hardware, such as a secure element. If used, this external hardware must be accessible only to the TEE [27].

One widely deployed TEE key management system is the Android Keystore, which lets (user-space) application developers generate and containerise cryptographic keys within a TEE [5]. The key material is generated using a TEE-based secret and never exposed outside the secure world, even if the OS is compromised. Android OS provides abstract interfaces to developers in order to operate upon this material, such as encryption, decryption, and signature verification and signing, but direct access to these keys is not given to developers. A similar system, the Android Gatekeeper [4], performs user authentication of inputted passwords in the TEE. The Gatekeeper enrolls user passwords by applying an HMAC to the password, identification value, and a hardware secret key, which is stored in the TEE. Subsequent authentication attempts involve regenerating this value and checking their equivalence. If successful, the Gatekeeper uses a TEE-derived secret to sign and transmit an authentication message to the Keystore TA. This notifies the Keystore TA that the user authentication secret was entered correctly and TEE-resident keys can be used. The Gatekeeper service limits the number of failed verification attempts and prevents further attempts using a timeout and counter.

Another application of TEE key management is for digital rights management (DRM) enforcement, as used by Microsoft PlayReady and Google Widevine frameworks [69, 32, 47]. Widevine—a widely used DRM platform used by leading video providers, including Netflix and Amazon Prime Video—supports TEE-based decryption and processing of video material. Widevine provides OEMs with a TA that contains decryption key for recovering encrypted video material sent by media providers. The decrypted media then undergoes video processing directly by the TEE without REE co-operation (WideVine Level 1 security) or passed to the REE for processing but with the decryption occurring in the TEE (WideVine Level 2).

3.4 Trusted Paths for Input/Output (I/O) Devices

Enforcing access control to security-critical peripherals is a key design principle of many of today's security systems. Allowing only the TEE to access these peripherals prohibits an untrusted native OS from reading sensitive data that could compromise the long-term security of device users. If supported, a SoC's TrustZone Protection Controller (TZPC)—see §2.2.3—can prevent non-secure accesses to secure world-only I/O peripheral devices, such as General Purpose I/O

(GPIO) and the I2C and SPI serial interfaces.

Biometric authentication on Android devices follows this approach, where the TEE acquires and processes facial and finger data using the camera and fingerprint reader peripherals respectively. The Android fingerprint authentication system, shown in Figure 3.2, gives only the TEE the ability to access the reader, which then executes the authentication algorithm and exposes the results to the untrusted world over a restricted interface. If the user is successfully authenticated, i.e. the fingerprint image corresponds to the enrolled image, then the TEE also notifies the Keystore TA to allow any TEE-resident cryptographic keys to be used. A similar approach is also used for facial authentication where the TEE has direct access to the camera peripheral without untrusted world co-operation. It is critical to understand that the untrusted world—OS services or third-party user applications—cannot access the peripherals for biometric data acquisition, or the authentication algorithm itself. This is to prevent a privileged adversary from simply disabling the authentication stage or harvesting users' fingerprint images, which would pose major long-term security and privacy issues.

In the DRM domain, the Google Widevine framework supports trusted path video output. Here, the WideVine TA decrypts the video stream from the provider, as described above in §3.3, and supports outputting video contents directly to a video buffer in the TEE. This buffer is outputted to the screen such that the REE never observes the decrypted video material.

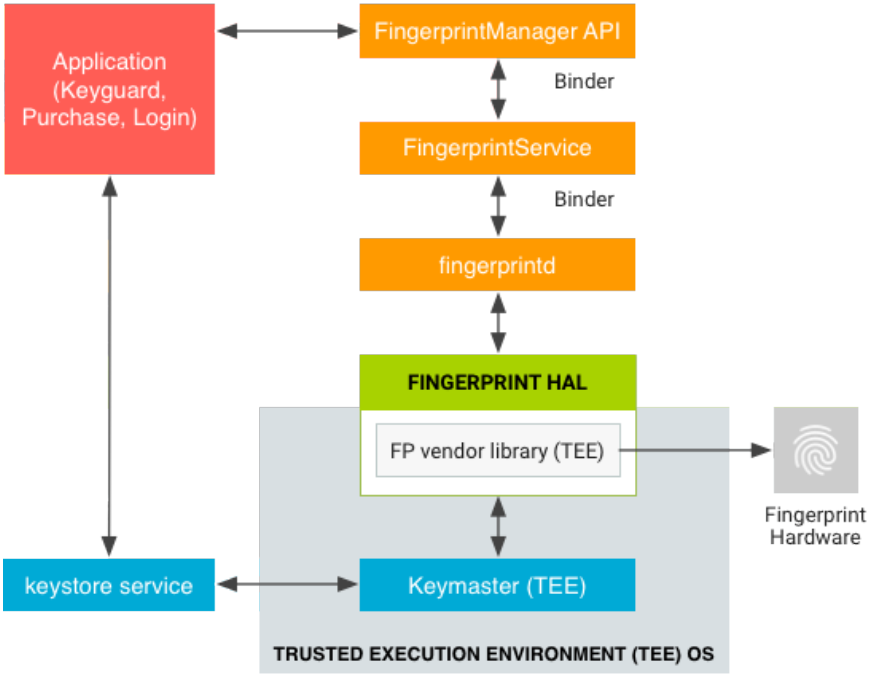

Figure 3.2: Android fingerprint authentication information flow [2].

3.5 Full Disk Encryption

Full disk encryption (FDE) is used to encrypt all device user data such that it cannot be readable by an attacker who has not decrypted its contents. FDE implementations use symmetric encryption, usually AES, to encrypt data at the block level. User data is encrypted before saving to it

to disk, such as a photograph or audio file; when the data is read, the decryption procedure is performed before returning it to the parent process. Android's FDE implementation operates as follows: a master key (up to 256 bits) is used to encrypt/decrypt data, which is generated and hashed with a default password and salt value when the device is booted for the first time. This hash is signed by the TEE; the signature of this hash is used to encrypt the master key. When the user sets their PIN or password, the 128-bit key is re-encrypted and stored. The AES-128 data encryption key (DEK) is encrypted with an AES-128 key encryption key (KEK), which is in turn derived from the user PIN though the password-based key derivation function 2 (PBKDF2) [72]. Using two different keys, namely the DEK and the KEK, renders re-encryption in the case of PIN changes unnecessary. Only with successful user authentication is the FDE key released such that it can be used to decrypt the contents of user data. Without successful authentication, the OS is not able to decrypt and read the protected data [3].

3.6 File-based Encryption

File-based encryption (FBE) refers to the encryption of data at a filesystem level, rather than a block or volume level as with FDE. This can allow different user profiles to exist under separate security policies; for example, a dedicated work profile that encrypts corporate files. FBE typically uses AES on commercial devices. On Apple devices, a 256-bit per-file encryption key (FEK) is generated each time a file on the data volume is created by Apple's proprietary Data Protection module, which uses the SEP for key management. The FEK is wrapped using a class key generated by the SEP and given to a hardware AES engine, which encrypts the file under the FEK as it is written to flash memory. When a file is read, it is decrypted by the AES engine as it is transmitted from memory. The encryption uses AES-128 in XTS mode, using 128 bits of the 256-bit key as the cipher key and the other 128 bits as the tweak.

FBE is also supported by Android implementations from version 7.0, which uses the Direct Boot feature to boot encrypted devices to the lock screen; only after successful user authentication are encrypted files decrypted. On Qualcomm-based SoCs (Snapdragon 855 and over) this works as follows [55]: FBE Credential Encrypted (CE) class keys are generated that are protected by a secret derived from a device unique key and a synthetic password. The synthetic password is generated by Android for each user, which is protected by the user credential, e.g. password, and (if applicable) an escrow token for providing access to system administrators. When the device is unlocked, the FBE CE key is decrypted and set in the Linux kernel key-ring. When the file system driver reads or writes a protected file, the key is retrieved from the kernel key-ring and set in the proprietary Qualcomm Inline Crypto Engine (ICE) for encrypting/decrypting files to and from memory using AES [57].

3.7 Future Directions

Security practitioners should also be aware of emerging methods and applications of using mobile TEEs, a multitude of which have been published in recent research. Examples include TEE-based mobile deep learning libraries [38] that allow multi-layer neural networks to execute aboard a mobile TEE, thus paving the way for secure image classification and speech recognition. Authenticating mobile adverts from advertising networks within the TEE was proposed by Li et al. [39] to protect users from malicious advertisements introduced, for example, using man-in-the-browser attacks. Processing REE system logs in TEEs has also been proposed for both Intel SGX [36]

and the GlobalPlatform TEE [64] to preserve logs used for audit and forensic investigations. Furthermore, TEE-to-TEE communication and credential management has also been proposed to aid data migrations, backups, and backups from TEEs with remote attestation [65, 66]. In other work, TEEs have been proposed for securing health-related data [62], confidential image processing [17], and executing cryptocurrency wallet operations [23].

Chapter 4

White-Box Mobile TEE System

This chapter describes the design choices and rationale, features, and limitations of our white-box mobile TEE system.

4.1 Commercial and Open-Source TEEs

As discussed in Chapters 2 and 3, TrustZone is the principle way by which TEEs are established on ARM-based SoCs, which are found ubiquitously on today's mobile devices and their security mechanisms [21]. However, TrustZone comprises the hardware components—SoC and CPU extensions—for creating a TEE. Additional modules are required to instantiate a full TEE, particularly a secure kernel that conforms to the GlobalPlatform TEE specifications. The secure kernel implements the TEE's *software* functionality; it is the kernel and its hosted applications that implement the TEE-driven security mechanisms on modern mobile devices. We have identified 11 commercial and open-source TrustZone-based TEEs that can be used to provide a fully functional TEE:

- **Huawei iTrustee** [34]: a closed-source TEE from Huawei for Hisilicon SoCs with ARM TrustZone. It is used to protect fingerprint screen unlocking, fingerprint payments, USB keys, Skytone, and the Huawei Wallet on modern Huawei devices. iTrustee is known to be compatible with the Kirin 980 SoC used by the Huawei P30, Mate 20, Nova 5T, Mate X, Mediapad M6, Honor 20, and View 20 models [33].
- **ProvenCore** [54]: a closed-source TEE developed by Prove & Run for TrustZone-based SoCs using Cortex-A and Cortex-M CPUs, and supports the RISC-V architecture. ProvenCore is backed by formal methods and has been certified to Common Criteria EAL7, above the minimum EAL2+ required for GlobalPlatform TEE certification.
- **Qualcomm Trusted Execution Environment (QTEE)** [56]: a closed-source TEE OS developed by Qualcomm for TrustZone-based SoCs. QTEE is known to be GlobalPlatform-compliant and is currently used on LG, Xiaomi, Sony, HTC, and OnePlus mobile devices. It is also publicly known to support Qualcomm-based SoC security functions such as cryptographic accelerators, random number generators, and eFuses.
- **Samsung TEEGRIS** [61]: a closed-source TEE developed by Samsung for TrustZone SoCs. TEEGRIS allows authorised developers to provision their own applications and services, and complies with the GlobalPlatform TEE specifications. TEEGRIS is currently the TEE on Samsung devices from the Samsung Galaxy S10 [58].

- **SierraTEE** [68]: a closed-source TEE that supports MIPS and ARM TrustZone. It is known to provide a comprehensive implementation of the GlobalPlatform APIs, with support for the ARM11, ARM Cortex-A8/A9/A15/A53/A55/A75, and the MIPS P5600 architectures.
- **TrustKernel T6** [70]: a closed-source TEE developed for ARM TrustZone SoCs with GlobalPlatform certification. It is currently used by over 200 OEMs with selected partners, principally based in China, including Alipay, Tencent, Baidu, ChinaDRM, ZTE and Skyworth.
- **Trustonic Kinibi** [71]: a closed-source TEE for ARM TrustZone-based SoCs. Kinibi is GlobalPlatform compliant and has been deployed on two billion devices worldwide. Kinibi was the main TEE for Samsung devices from the release of the Galaxy S6 to the Galaxy S9, after which it was replaced with TEEGRIS [58].
- **Nvidia TLK** [48]: the Nvidia Trusted Little Kernel (TLK) is an open-source TEE built for Nvidia Tegra SoCs. It is a fork of the Little Kernel (LK) open-source embedded kernel project [35], supports the GlobalPlatform TEE specifications, and provides limited but publicly available documentation. However, the last code commit was in 2015.
- **OP-TEE** [42]: the Open Portable TEE (OP-TEE) is an open-source platform first developed by ST-Ericsson. It is currently maintained by the Trusted Firmware project, which develops reference implementations of ARM firmware that includes Google, STMicroelectronics, ARM and NXP. It is actively maintained with a large base of documentation and supports popular non-secure world OSs, e.g. Android and Debian Linux, and secure world application development. OP-TEE is maintained for several ARM-based development kits, including the HiKey Board (HiSilicon Kirin 620 SoC), Raspberry Pi 3, Atmel ATSAMA5D2-XULT, ARM Juno, and the QEMU hardware virtualisation platform.
- **Open-TEE** [46]: an open-source virtual TEE developed by Aalto University and the Intel Collaborative Research Institute for Secure Computing. Open-TEE supports the GlobalPlatform TEE specifications, enables the development of third-party applications and services, and provides low-level access to firmware components. It also provides build support for Android OS.
- **Trusty**: an open-source TEE developed by Google for Android OS. It is derived from the LK project [35] and provides a non-secure world kernel driver and a user-space Android library for interfacing with the TEE. However, Trusty is not GlobalPlatform-compliant and uses Google-specified interfaces for inter- and intra-world communications.

4.1.1 Evaluation Criteria

To compare these technologies, the following set of evaluation criteria are used to summarise and contrast the features provided by each system:

- **C1**: *Open-source*. The system's source code is publicly available and supports the development of new features, applications and services, and the modification of existing ones. For our goals, an open-source system is critical to enabling researchers and analysts to inspect, modify and redeploy critical TEE components in a white-box fashion.
- **C2**: *GlobalPlatform-compliant*. The system complies to the GlobalPlatform TEE specifications. This provides confidence that new attacks and security mechanisms are developed using a standardised TEE architecture that will be encountered in the wild.

- **C3**: *Actively maintained*. The TEE is receiving active maintenance in the form of bug fixes, new features and new platform support.
- **C4**: *Public documentation*. The system has publicly available documentation about its low- and high-level features. This is indispensable in order to fully characterise the TEE without entering into a commercial relationship or requiring extensive reverse engineering.
- **C5**: *Firmware access*. The TEE's firmware is open-source and can be modified. While the security *operating system* might be open-source, it is critical for the underlying firmware to be open-source, which may be supplied by a third-party. This is important when developing new unsupported security mechanisms and developing and understanding proof-of-concept attacks against low-level components.
- **C6**: *Implements common security mechanisms*. The system supports common security mechanisms, such as secure encrypted storage and secure boot, against which potential attacks could be developed.
- **C7**: *Deployable on commercial handsets*. The TEE can be deployed on commercially available, off-the-shelf mobile handsets.
- **C8**: *Deployable on commercial development kits*. The TEE can be deployed on commercially available, off-the-shelf development boards that closely resemble, but do not directly reflect, a complete commercial mobile platform.

4.1.2 Analysis and Discussion

A comparison of commercial and open-source TEEs is presented in Table 4.1. Notably, only a minority of TEEs have been published in the open-source domain; the majority are closed-source systems that require an established business relationship with the provider. The TEEGRIS, for example, *"is only available to strategic partners with strategic partnership agreements"* with Samsung [61]. Certain TEEs limit support to SoCs by the same manufacturer. Qualcomm's QTEE provides a hardware abstraction layer (HAL) for Qualcomm-based SoCs, such as the Snapdragon series; Samsung TEEGRIS is deployed only on Samsung-based SoCs at present; and Huawei's iTrustee is known to be certified only against Hisilicon SoCs, a Huawei-owned semiconductor subsidiary [33, 56, 61]. We are currently not aware of widely available mobile smartphones or tablets currently using the ProvenCore or Sierra TEEs, while very few details are publicly known about the T6 kernel's deployment.

Table 4.1: Comparison of commercial and open-source TEEs.

	Evaluation Criteria							
TEE	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**
Huawei iTrustee [34]	✗							
ProvenCore [54]	✗							
Qualcomm TEE [56]	✗							
Samsung TEEGRIS [61]	✗							
Sierra TEE [68]	✗							
TrustKernel T6 [70]	✗							
Trustonic Kinibi [71]	✗							
Nvidia TLK [48]	✓	✗	✗	✓	✓	✓	✗	✓
OP-TEE [42]	✓	✓	✓	✓	✓	✓	✗	✓
Open-TEE [46]	✓	✳	✳	✓	✓	✓	✗	✗
Trusty [6]	✓	✗	✓	✓	✓	✓	✗	✓

✓: Satisfies, ✗: Does not satisfy, ✳: Partially satisfies. Gray regions cannot be verified due to proprietary restrictions.

Among the open-source TEEs, Nvidia's TLK is a promising secure world OS stack, which is released as FOSS[1] under the MIT license. The code base is relatively small (23k lines of C), thus limiting operational complexity, and supports the Tegra SoC with TrustZone extensions. However, TLK's is not actively maintained[2] and implements its own REE client and TEE internal APIs that are not GlobalPlatform-compliant. Complying to the GlobalPlatform TEE specifications is an important feature when developing a research platform to develop attacks on applications and services. It provides a level of assurance that research is applicable to the standard behaviour of mobile TEEs, which are evaluated to these specifications under the Common Criteria framework. Trusty shares some similarities with Nvidia's TLK: both are built from the LK open-source embedded kernel project and support TrustZone-based SoCs. However, Trusty is an actively developed TEE that is maintained as part of the Android Open Source Project. Unfortunately, like TLK, Trusty implements client and internal APIs that are not faithful to the GlobalPlatform specifications. In contrast, Open-TEE is a dedicated platform for conducting TEE-based research, which is built around remaining faithful to the GlobalPlatform specifications:

> *"Our primary motivation for the virtual TEE is to use it as a tool for developers of Trusted Applications and researchers interested in using TEEs or building new protocols and systems on top of it. Although hardware-based TEEs are ubiquitous in smartphones and tablets ordinary developers and researchers do not have access to it. While the emerging GlobalPlatform specifications may change this situation in the future, a fully functional virtual TEE can help developers and researchers right away."*
> — Open-TEE project [52]

Open-TEE was developed in 2015 and remains under semi-active development. It provides publicly available documentation and supports Android as the non-secure world OS. Open-TEE also has integrated support for GDB[3] and the QtCreator C/C++ IDE. One major downside is that

[1] FOSS: Free open-source software.
[2] TLK's last code commit is 5+ years ago.
[3] GDB: GNU Debugger.

the project is a *virtualised* TEE; it is not supported for deployment on physical development kits or handsets. This prohibits researchers from analysing physical and micro-architectural attacks based on hardware idiosyncrasies, such as Rowhammer-type attacks and cache-based side-channel attacks. In the last of the open-source TEEs, OP-TEE is an actively development TEE framework maintained by the Trusted Firmware industry consortium. Like Open-TEE, OP-TEE is GlobalPlatform-compliant, supports Android as the REE OS, provides debugging functionality and has well-developed, publicly available documentation. OP-TEE supports several commercially available TrustZone development boards and the QEMU virtualisation platform. It supports common security mechanisms defined in the GlobalPlatform TEE specifications, including encrypted secure storage and secure booting of the secure world. A downside is that we are not aware of any commercial handsets that are currently using OP-TEE. Similar to the Open-TEE, its aim is to facilitate research and prototype development on a GlobalPlatform-compliant TEE.

We note that no open-source TEE can be readily developed and deployed on commercial handsets. This is because the secure boot process requires OEM-signed binaries of the secure world and its firmware; new firmware, applications and services cannot simply be redeployed without ascertaining corresponding signatures from the OEM. We are not aware of any mobile device OEMs that offer free and open access to their TEE platform.

In conclusion, we have selected OP-TEE from which to build a testbed platform for mobile TEE research. In summary, this is based on its faithful compliance with the GlobalPlatform specifications; wide documentation and community support; support for common security mechanisms and popular non-secure world OSs, e.g. Android and Debian Linux; and support for physical ARM TrustZone development kits. The following sections in this chapter describe the construction of this testbed.

4.2 Proposed Test-Bed Platform

This section summarises the proposed white-box TEE platform, its secure world functionality, cryptographic implementation details and its key management.

4.2.1 Overview

Our test-bed platform is underpinned by the OP-TEE project—an open-source, ARM TrustZone-based TEE that was first developed by ST-Ericsson. STMicroelectronics maintained OP-TEE following the split of ST-Ericsson in 2013, then by the Linaro Group in 2015–2019, and is currently maintained by the Trusted Firmware project. It is built as a secure world counterpart to a non-secure world Linux kernel running on the ARM architecture with a Cortex-A CPU. The system follows the GlobalPlatform TEE system architecture and implements both the Internal Core and Client APIs for intra- and inter-world communication respectively. The reader is referred to [49, 41, 42] for detailed documentation regarding its supported functionality and limitations.

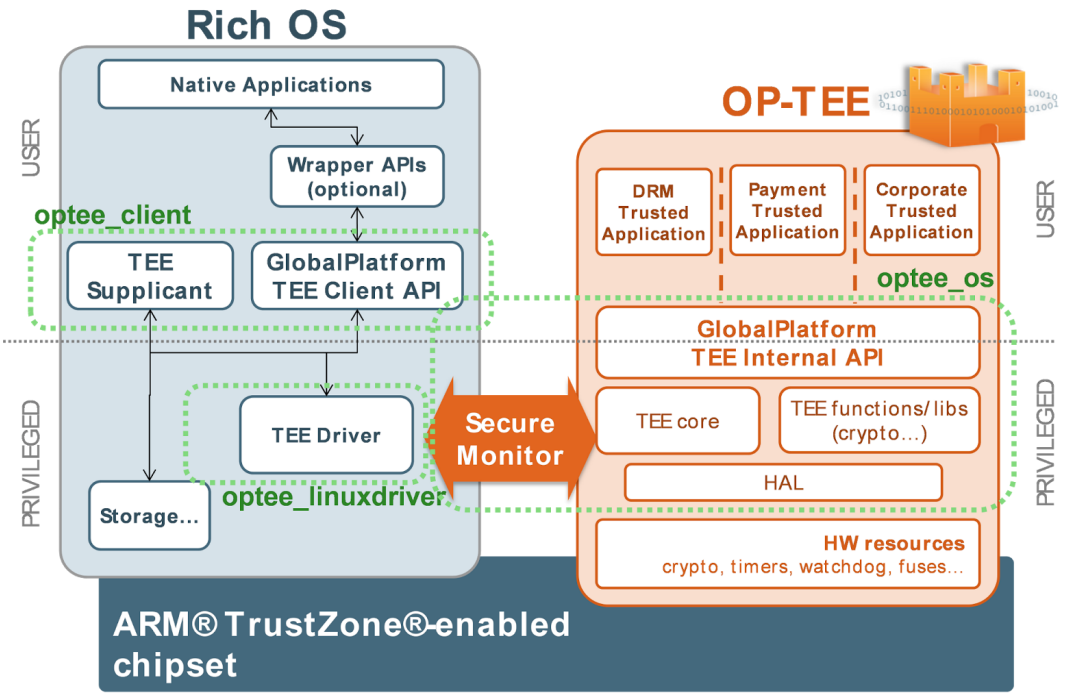

Figure 4.1: OP-TEE high-level system architecture [41].

The platform mirrors the OP-TEE architecture, illustrated in Figure 4.1, which comprises the following components:

- A secure world operating system that executes at EL-1 (see §2.2.2).
- Secure world user-space libraries for TEE TAs to interact with the secure world OS.
- A non-secure world Linux kernel TEE framework and driver.
- Non-secure world user-space libraries that comply with the GlobalPlatform TEE Client APIs.
- Non-secure world daemon (`tee-supplicant`) for facilitating auxiliary TEE operations, such as TA performance profiling using `gprof`.
- Test suite for regression and specification testing of GlobalPlatform API implementations.
- Example non-secure and TEE applications.
- Debugging tools and build scripts for over 50 supported ARM development boards (see Chapter 7).

Additionally, the test-bed uses the ARM Trusted Firmware framework [14]—a reference firmware developed by ARM—for implementing the secure boot sequence and the secure monitor code for Cortex-A processors.

4.2.2 Secure World Core Functionality

The primary component is the secure world kernel that complies with the GlobalPlatform specifications. We now discuss the main features of the secure world OS, including world context switching, memory management, and trusted applications.

World Switching and Interrupt Handling

The secure OS executes in the secure world at EL-1 under the ARMv8 architecture, with world switches performed by the secure monitor (at EL-3). Switching between the worlds occurs when the REE executes a secure monitor call (SMC) instruction. The SMC is trapped by the monitor code that switches execution mode—saves the current context, general purpose and non-banked system registers, and sets the NS bit—if the target service is in the TEE. Similarly, when OP-TEE returns execution to the normal world, the OP-TEE OS executes an SMC instruction that is trapped by the monitor, which restores the REE context and switches the execution mode back to the normal world.

Each world owns and handles its own exception vector tables, except when secure world interrupts are handled during non-secure execution and vice-versa. If secure interrupts are signalled while the processor is in secure mode, then the secure OS handles it from its exception vector. If secure interrupts are raised while the CPU is in non-secure mode, the secure monitor handles the exception and invokes the secure OS to serve the interrupt. When non-secure world interrupts are raised, the normal world handles it in its own interrupt exception vector. Lastly, if the system is in secure execution mode and a non-secure interrupt is raised, OP-TEE OS will transiently return to the REE to allow it to handle the non-secure interrupt.

Memory Management

At present, the secure OS requires 256kB RAM as a minimum requirement. Both dynamic and static RAM is supported, the latter of which is supported using memory paging to overcome situations where the kernel does not fit into SRAM. For virtual memory management, the secure OS uses several translation tables for mapping virtual memory to physical memory: one of size 4GB and two or more smaller tables of size 32MB. The 4GB table handles kernel model mapping, while the smaller tables are assigned per thread and cover the virtual memory mapping per TA context. We note that the secure OS assigns a (trusted) thread for each SMC call that handles the execution context of the requested TEE service.

The secure OS can also support the shared memory between the TEE and the REE in line with the GlobalPlatform TEE, which can be used to transfer data between each world. The shared memory is allocated and managed by the non-secure world by the Linux OP-TEE driver. Both contiguous and non-contiguous shared memory buffers are supported.

At boot-time, the test-bed uses the open-source Trusted Firmware framework. This implements the secure boot procedure and, if available, configures the TrustZone Address Space Controller (TZASC). The TZASC allocates secure/non-secure memory regions and enforces access control of secure/non-secure accesses to those regions at run-time. The reader is referred back to §2.2.3 for a description of TZASPC and the controllers used to protect other memory units.

Trusted Applications

Two methods are supported for implementing trusted applications (TAs): user-mode TAs, which conform to the GlobalPlatform TEE specifications and run at a lower exception level than the TEE kernel, and pseudo-TAs. User-space TAs are designed for typical TEE application, such as for implementing biometric authentication, keystores, and more. OP-TEE provides the `libutee` library that enables user-space TAs to interact with the OP-TEE Core services using the GlobalPlatform Internal Core APIs. The non-secure world invokes a TA by calling its universally unique identifier (UUID) using the Linux library, after which the OP-TEE OS memory maps the TA into the secure world. OP-TEE provides support for signing and encrypting secure world TAs under a signing key that is used for the OP-TEE OS blob. As such, TAs can be stored in untrusted persistent memory, e.g., flash memory, prior to being used.

Pseudo-TAs are statically built into the OP-TEE binary image execute at the same exception level as the kernel. They are designed to act as interfaces to the OP-TEE core to facilitate the development of higher privilege components, rather than needing to directly modify the OP-TEE core. This is useful when building novel secure OS services or low-level tests that require a higher exception level.

4.2.3 Cryptographic Implementation Details

The secure OS implements the cryptographic operations specified by the GlobalPlatform TEE Internal Core API [28]. This includes signature signing and verification, including elliptic curve-based operations; asymmetric and symmetric algorithms, e.g., AES in various modes of operation, such as Cipher Block Chaining (CBC); message authentication codes (MACs); authenticated encryption; cryptographic one-way hash functions; and key derivation functions. The reader is referred to the GP Internal Core API [28] for a full specification of supported algorithms and modes of operation.

The secure OS implements cryptographic operations using the open-source LibTomCrypt [40] library and, where available, ARM Cryptography Extensions for providing instruction-level AES and SHA. The functions it exposes to TAs, however, are far removed from the actual implementations, as per Listing 1. The TA calls the GlobalPlatform Internal Core API functions, which are handled using the `libutee` user-space library. This library uses the system call interface to call the corresponding cryptographic implementation in LibTomCrypt built into the secure OS core. The secure OS also provides an abstraction layer for supporting additional cryptographic operations. This could be useful, for example, when implementing novel mechanisms based on post-quantum cryptography that are not yet supported by the GlobalPlatform specifications.

Listing 1 OP-TEE GlobalPlatform Internal API call stack [50].

```
1: ta_function() {Arbitrary TA function – User space}
2: → TEE_*() {Provided in libutee.a}
3:    → utee_*() {System call interface}
4:       → tee_svc_*() {Kernel space}
5:          → crypto_*() {libtomcrypt.a and crypto.c}
6:           /* LibTomCrypt */ {libtomcrypt.a; algorithm implementations}
```

4.2.4 Security Mechanisms

We now describe the key security mechanisms provided by our test-bed platform.

Secure Boot

The platform supports an authenticated secure boot process implemented by the Trusted Firmware Framework for Cortex-A CPUs. This process mirrors that outlined in §2.2.4 of Chapter 2. It uses a chain of trust starting with a trusted component: this is the first bootloader image (BL1) and a SHA-256 hash of the root of trust public key. The BL1 first loads and verifies the BL2 certificate. The hash of this key is calculated and compared with the root of trust SHA-256 hash; if successful, the hash of the BL2 image is read from the certificate. BL1 loads BL2, computes its hash, and verifies it against that contained in the certificate. If successfully verified, control is passed to BL2. This process repeats with BL2 to load the trusted and non-secure worlds with their public keys. At each stage, the hash of the public key is checked against that in the root of trust. OP-TEE uses the open-source mbedTLS [12] embedded cryptography library to implement this process. During the build process, the developer can specify an RSA certificate as the root of trust. All of the components in the boot chain are self-signed.

Secure Storage

The secure OS implements the Trusted Storage abstraction defined by the GlobalPlatform TEE. This assumes one of two implementations:

- Using the filesystem controlled by REE. As per the GlobalPlatform TEE specifications, this is acceptable as long as TEE data is protected using a method *"as strong as the means used to protect the TEE code and data itself"* [27].
- Using the Replay Protected Memory Block (RPMB) partition of an eMMC device.

Using the first method, a TEE TA calls a function using the GlobalPlatform Internal Core API to write data to a persistent object. This triggers a system call implemented by the secure OS that encrypts the data and sends to the REE over the secure monitor. The TEE supplicant (see Figure 4.1) receives the message and stores the encrypted data to a Linux-based filesystem. Reading files invokes the reverse operations, with the object being decrypted in the final step. The encryption and decryption operations are performed using AES in GCM mode and keyed using a file encryption key (FEK), which is derived using a four-tier hierarchy (from root to lowest):

1. **Hardware unique key (HUK)**: a unique, per-device key that is provisioned by the device's manufacturer into read-only memory. Currently, the secure OS uses a static key for the HUK; its concrete implementation is left to the manufacturer.
2. **Secure storage key (SSK)**: a per-device key that is generated by the secure OS on first boot. It is derived using an HMAC from the HUK as follows, where $||$ is the concatenation operation:
$$SSK = HMAC_{SHA256}(HUK, ChipID \ || \ static\_string)$$
3. **TA storage key (TSK)**: a per-TA key that is used to encrypt and decrypt the file encryption key (FEK). It is derived as follows:
$$TSK = HMAC_{SHA256}(SSK, UUID_{TA})$$

4. **File encryption key (FEK)**: generated every time the TA produces a persistent object file from a pseudo-random number generator (PRNG). The FEK is saved in a meta-file after being encrypted under the TSK.

The second secure storage abstraction uses the device's RPMB. This is a small, authenticated area controlled by the secure world. It uses a shared key between the controller and the OP-TEE OS to authenticate all read/write operations that access the secure area, which is configured at launch time, to prevent accesses from the REE.

Additional Mechanisms

We have also deployed TAs proposed in related research in [64] for protecting non-secure world system logs in the TEE, and [65] for performing TEE-to-TEE remote attestation. Both of these comply with and use functions from the GlobalPlatform Client and Internal Core APIs. Furthermore, we have successfully developed skeleton pseudo-TAs, which execute at a higher privilege level, for developing novel TEE security mechanisms and for conducting security tests on conventional user-space TAs.

4.2.5 Supported Development Platforms

The platform currently supports 51 commercially and non-comercially available ARM development boards with TrustZone extensions. These include the HiKey 620 and 960 boards, based off the HiSilicon Kirin 620 and 960 SoCs respectively; the Raspberry Pi 3; ARM Juno and Foundation FVP; NXP i.MX7Dual SabreSD board; Atmel ATSAMA5D2-XULT board; the Xilinx Zynq 7000; and STMicroelectronics STM32MP1. The platform also supports the QEMU hardware virtualisation platform for the ARMv7 and ARMv8 architectures. This is useful when rapidly prototyping test services and applications without re-initialising and re-flashing physical development boards. A comprehensive list of the supported platforms is provided in Appendix A in Chapter 7.

For our test-bed, we have successfully deployed the infrastructure on the HiKey 620 ARM development, pictured in Figure 4.2. This board was chosen due to its commercial availability and closely resembles a typical architecture of a modern mobile device. In particular, the HiSilicon Kirin 620 is same SoC used by a large range of Huawei smartphones, including the Huawei Honor 5A, 4C and 4X, the Honor Holly 3, Huawei Y6, P8 and P8 Lite, and G Play Mini. The HiKey boards are also the only boards that OP-TEE provides documented support for Android as the non-secure world OS. It is important to note that the infrastructure does not support the most recent mobile handsets; development platform for such devices, e.g., Apple iPhone 12 or Samsung Galaxy S20, are not commercially available, nor are they supported by the OP-TEE project.

The specifications of the HiKey 620 board are as follows:

- **SoC**: HiSilicon Kirin 620.
- **CPU**: 64-bit, eight-core ARM Cortex-A53 at 1.2GHz (ARMv8 architecture).
- **GPU**: ARM Mali 450-MP4 with OpenGL ES and OpenVG support.
- **RAM**: 2GB LPDDR3 SDRAM at 800MHz.

- **Storage**: 8GB eMMC with a MicroSD card slot.
- **Network connectivity**: Ethernet (via USB2.0), IEEE 802.11 b/g/n Wi-Fi (2.4GHz), Bluetooth and Bluetooth Low Energy support.
- **Display**: full-size HDMI 1.4 output and serial MIPI/DSI HDMI output.

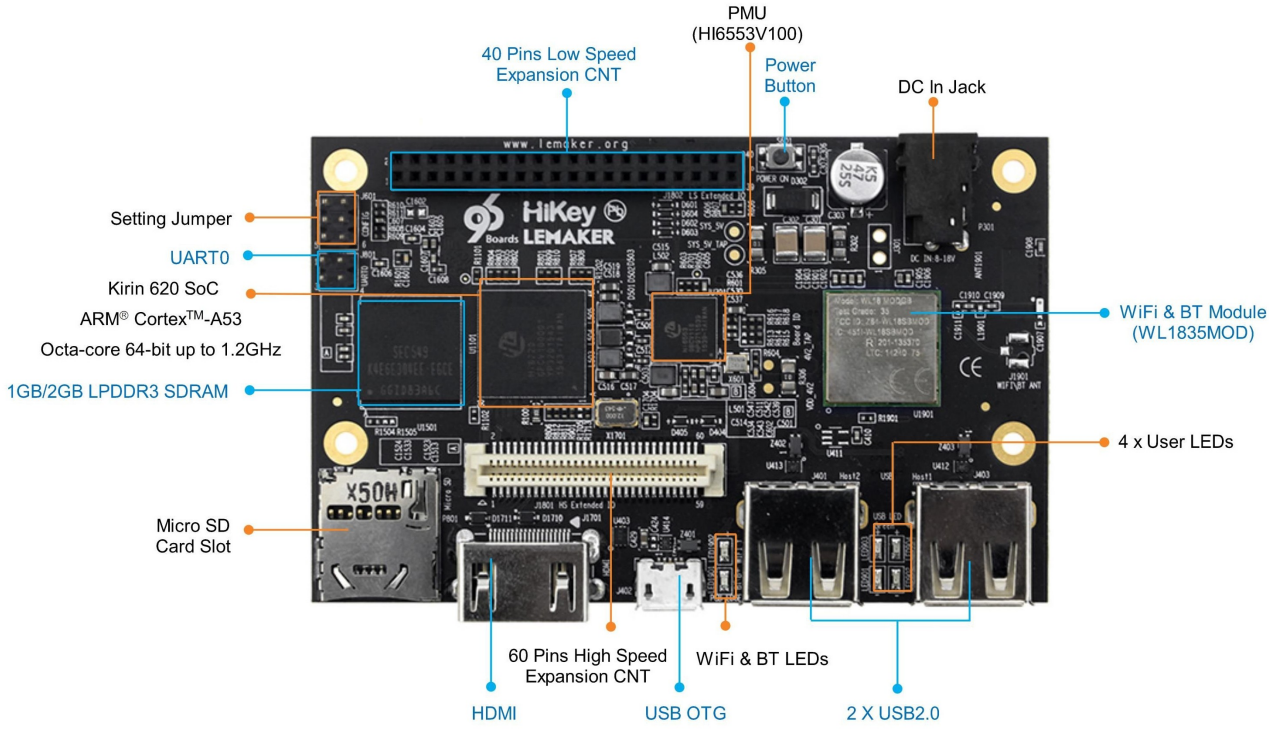

Figure 4.2: HiKey 620 ARM development board with a HiSilicon Kirin 620 SoC [15].

The OP-TEE infrastructure, including the ARM Trusted Firmware framework, is used for the secure world OS, secure monitor code, TA libraries (including GlobalPlatform API implementations), and non-secure world drivers. It is used to implement the architecture shown previously in Figure 4.1. For the non-secure world, we have successfully trialled both Debian Linux and Android OS (version P), the latter of which closely mimics the setup on a commercial handset. We have also deployed the infrastructure using the QEMU hardware virtualisation platform for rapid prototyping.

4.2.6 Limitations

Our test-bed platform contains several limitations. Firstly, retrieval of the hardware unique key (HUK) is not supported by the secure world OS. According to the OP-TEE project, this is because *"information about how to retrieve key data from the SoC is considered sensitive by the vendors"* and thus not publicly available [50]. The secure world OS uses a constant value as the HUK; encrypted data can therefore be decrypted elsewhere with the same key, i.e., there is no hardware-software device binding of encrypted data. However, the cryptographic operations that are performed using the synthetic HUK are the same as platforms that provide direct access to the HUK; the difference is in (hardware) key accessibility only.

Secondly, there is currently limited support for secure off-board peripherals using the secure OS, such as fingerprint readers and sensor hubs. While this is theoretically implementable—for example, developing kernel-mode drivers in the secure OS to interface with a peripheral over the SPI or I2C connectors of the board's 40 pin expansion slot—we note that no support is provided out of the box. This increases the complexity of conducting research against mechanisms that protect secure world-only peripherals.

Thirdly, the chosen board does not contain eFuse banks that are used for implementing one-time programmable (OTP) memory. We are not aware of any commercially available boards that implement secure world eFuses and the remaining architecture in order to test specific attacks, e.g., physical fault injections that temporarily change the fuse state.

Another limitation is that the OP-TEE infrastructure does not support multiple TEEs. This is an optional possibility in the GlobalPlatform TEE specifications, whereby multiple TEEs with independent secure OSs and sets of TAs are maintained [27]. The envisaged scenario is to allow OEMs to provision TEEs from competing business stakeholders without needing to bundle them into a single environment. However, the conventional setup of modern mobile devices is to use a single TEE with a single set of TAs at the time of writing [33]. A lack of support for multiple TEEs is, therefore, not significantly disruptive when designing attacks and new security mechanisms for today's devices.

Lastly, our platform is not Common Criteria-certified. As far as we are aware, no OP-TEE boards have been Common Criteria-certified against the GlobalPlatform TEE protection profile. This is to be expected: the aim of OP-TEE and the development boards are to be used a reference implementation with which to conduct research and for OEMs to prototype functionality. It is likely the case that CC-certified devices may resist a larger set of attacks than those against the prototype research platform. However, we are not aware of any commercially available CC-certified platforms with the same feature set as our test-bed platform.

Chapter 5

Summary and Conclusion

The difficulty of forensic data extraction from modern mobile platforms has increased significantly because of their growing complexity and increasing focus on security. Today, these devices contain a range of security mechanisms, such as authenticated boot chains from a hardware-based root of trust, full-disk encryption (FDE), and trusted execution environments (TEEs) that provide hardware-assisted isolation of cryptographic keys and security-critical applications. It is necessary for law enforcement agencies to retrieve forensic evidence protected by these mechanisms for serious crime investigations.

To this end, this report presented a white-box platform that can be used for security research against modern mobile platforms using TEEs. It began with a detailed description of mobile TEEs and common mobile security mechanisms, including ARM TrustZone, remote attestation, secure and trusted boot, FDE and TEE-backed key managers. It also included a detailed description of the standards that govern mobile TEEs and their and security requirements, namely the GlobalPlatform TEE specification suite. This was followed by an examination of commercial and open-source frameworks for implementing TEEs on ARM-based devices, which were compared against a common base of evaluation criteria.

Our white-box system is built from open-source components that can be inspected, modified and redeployed as needed in order to develop novel extraction methods. The platform implements a wide variety of open-source common security mechanisms and conforms to the GlobalPlatform specifications. It uses a GlobalPlatform-compliant architecture for creating a TEE using ARM TrustZone, as used by most of today's mobile devices. This is built using the open-source OP-TEE project, first developed by ST-Ericsson and currently maintained by the Trusted Firmware project—a consortium comprising major industry players, including Google, STMicroelectronics, ARM and NXP. The model also supports an authenticated secure boot sequence based on the Trusted Firmware Framework, which is a widely used open-source reference implementation developed by ARM. Secure storage abstractions are also provided for protecting TEE data in line with the GlobalPlatform TEE specifications.

In summary, the proposed white-box platform closely replicates the architecture of modern mobile devices, which we have deployed on an ARM development board using a system-on-chip found on several Huawei smartphone models. It is hoped that this report provides valuable details to forensic analysts looking to develop a platform for evaluating extraction methods on devices that use modern security mechanisms.

Chapter 6

List of Abbreviations

Abbreviation	Translation
AES	Advanced Encryption Standard
ADC	Analog-to-Digital Converter
API	Application Programming Interface
BL	Boot Loader
CPU	Central Processing Unit
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
EL	Exception Level
EL	Enhanced Privacy ID protocol
FBE	File-Based Encryption
FDE	Full-Disk Encryption
FEK	File Encryption Key
FPGA	Field-Programmable Gate Array
GP	GlobalPlatform
GPIO	General Purpose Input/Output
GPU	Graphics Processing Unit
HMAC	Hash-based Message Authentication Code
HUK	Hardware Unique Key
I2C	Inter-Integrated Circuit (communication bus)
LEA	Law Enforcement Agency
MMU	Memory Management Unit
NFC	Near-Field Communication
OEM	Original Equipment Manufacturer
OS	Operating System
POST	Power-On Self-Test
RAM	Random Access Memory
REE	Rich Execution Environment

Abbreviation	Translation
ROM	Read-Only Memory
RoT	Root of Trust
RSA	Rivest–Shamir–Adleman
SCA	Side-Channel Attack
SoC	System-On-Chip
SPI	Serial Peripheral Interface
SSK	Secure Storage Key
TA	Trusted Application
TEE	Trusted Execution Environment
TSK	TA Storage Key
TZ	TrustZone
TZASC	TrustZone Address Space Controller
TZPC	TrustZone Peripheral Controller
TZMA	TrustZone Memory Adapter
USB	Universal Serial Bus

Chapter 7

Appendix A: OP-TEE Supported Platforms

Table 7.1: OP-TEE supported ARM platforms, reproduced from [51].

Platform	Publicly Available?	Maintained?
ARM Juno Board	✓	✓
Atmel ATSAMA5D2-XULT Board	✓	✓
Broadcom ns3	✗	✓
DeveloperBox (Socionext Synquacer SC2A11)	✓	✓
FSL ls1021a	✓	✓
NXP ls1043ardb	✓	✓
NXP ls1046ardb	✓	✓
NXP ls1012ardb	✓	✓
NXP ls1028ardb	✓	✓
NXP ls1088ardb	✓	✓
NXP ls2088ardb	✓	✓
NXP ls1012afrwy	✓	✓
FSL i.MX6 Quad SABRE Lite Board	✓	✓
FSL i.MX6 Quad SABRE SD Board	✓	✓
SolidRun i.MX6 Quad Hummingboard Edge	✓	✓
SolidRun i.MX6 Dual Hummingboard Edge	✓	✓
SolidRun i.MX6 Dual Lite Hummingboard Edge	✓	✓
SolidRun i.MX6 Solo Hummingboard Edge	✓	✓
FSL i.MX6 UltraLite EVK Board	✓	✓
NXP i.MX7Dual SabreSD Board	✓	✓
NXP i.MX7Solo WaRP7 Board	✓	✓
NXP i.MX8MQEVK Board	✓	✓
NXP i.MX8MMEVK Board	✓	✓
ARM Foundation FVP	✓	✓

Table 7.2: OP-TEE supported ARM platforms, reproduced from [51] (continued).

Platform	Publicly Available?	Maintained?
HiSilicon D02	✗	✓
HiSilicon Hi3519AV100 Demo Board	✗	✓
HiKey Board (HiSilicon Kirin 620)	✓	✓
HiKey960 Board (HiSilicon Kirin 960)	✓	✓
Marvell ARMADA 7K Family	✓	✓
Marvell ARMADA 8K Family	✓	✓
Marvell ARMADA 3700 Family	✓	✓
MediaTek MT8173 EVB Board	✗	✓
Poplar Board (HiSilicon Hi3798C V200)	✓	✓
QEMU	✓	✓
QEMUv8	✓	✓
Raspberry Pi 3	✓	✓
Renesas RCAR	✗	✓
Rockchip PX30	✗	✓
Rockchip RK322X	✗	✓
Rockchip RK3399	✓	✓
STMicroelectronics b2260 - h410 (96boards fmt)	✗	✓
STMicroelectronics b2120 - h310 / h410	✗	✓
STMicroelectronics STM32MP1 series	✓	✓
Allwinner A64 Pine64 Board	✓	✓
Texas Instruments AM65x	✓	✓
Texas Instruments DRA7xx	✓	✓
Texas Instruments AM57xx	✓	✓
Texas Instruments AM43xx	✓	✓
Xilinx Zynq 7000 ZC702	✓	✗
Xilinx Zynq UltraScale+ MPSOC	✓	✗
Spreadtrum SC9860	✗	✗

Chapter 8

Appendix B: Related Projects

This appendix references related projects funded under the same topic in 2019/2020.

- **RISEN**: Real-time on-site forensic trace qualification.
  - Coordinator: Agenzia Nazionale per le Nuove Tecnologie, l'Energia e lo Sviluppo Economico Sostenibile (ENEA).
  - https://www.risen-h2020.eu/
  - https://cordis.europa.eu/project/id/883116
- **GRACE**: Global response against child exploitation.
  - Coordinator: Fundacion Centro de Tecnologias de Interaccion Visual y Comunicaciones Vicomtech.
  - https://www.grace-fct.eu/
  - https://cordis.europa.eu/project/id/883341
- **CREST**: Fighting crime and terrorism with an IoT-enabled autonomous platform based on an ecosystem of advanced intelligence, operations, and investigation technologies.
  - Coordinator: Serviciul de Protectie si Paza.
  - https://project-crest.eu/
  - https://cordis.europa.eu/project/id/833464
- **FORMOBILE**: From mobile phones to court — a complete forensic investigation chain targeting mobile devices.
  - Coordinator: Hochschule Mittweida (FH).
  - https://formobile-project.eu/
  - https://cordis.europa.eu/project/id/832800
- **INSPECTr**: Intelligence network and secure platform for evidence correlation and transfer.
  - Coordinator: University College Dublin, National University of Ireland, Dublin.
  - https://inspectr-project.eu/
  - https://cordis.europa.eu/project/id/833276

- **LOCARD**: Lawful evidence collecting and continuity platform development.
  - Coordinator: Athina-erevnitiko Kentro Kainotomias Stis Technologies Tis Pliroforias, Ton Epikoinonion Kai Tis Gnosis.
  - https://locard.eu/
  - https://cordis.europa.eu/project/id/832735
- **ROXANNE**: Real time network, text, and speaker analytics for combating organized crime.
  - Coordinator: Fondation de l'Institut de Recherche Idiap.
  - https://roxanne-euproject.org/
  - https://cordis.europa.eu/project/id/833635
- **DARLENE**: Deep AR law enforcement ecosystem.
  - Coordinator: Ethniko Kentro Erevnas Kai Technologikis Anaptyxis.
  - https://www.darleneproject.eu/
  - https://cordis.europa.eu/project/id/883297
- **INFINITY**: Immerse. Interact. Investigate.
  - Coordinator: Airbus Defence and Space SAS.
  - https://cordis.europa.eu/project/id/883293

Bibliography

[1] M. Al Fahdi, Nathan L. Clarke, and Steven M. Furnell. Challenges to digital forensics: A survey of researchers & practitioners attitudes and opinions. In *Information Security for South Africa*, pages 1–8. IEEE, 2013.

[2] Android. Fingerprint HIDL, 2020. https://source.android.com/security/authentication/fingerprint-hal.

[3] Android. Full-disk encryption, 2020. https://source.android.com/security/encryption/full-disk.

[4] Android. Gatekeeper, 2020. https://source.android.com/security/authentication/gatekeeper.

[5] Android. Keystore, 2020. https://developer.android.com/training/articles/keystore.

[6] Android. Trusty TEE, 2020. https://source.android.com/security/trusty.

[7] Android Open Source Project (AOSP). Android, 2020. https://www.android.com/.

[8] Apple, Inc. Apple iOS, 2020. https://www.apple.com/uk/ios/ios-14/.

[9] ARM Holdings. ARM Trusted Firmware: Roadmap and progress, 2015. https://www.youtube.com/watch?v=je0_-yYgKdc.

[10] ARM Holdings. Introducing 2017's extensions to the ARM architecture, 2017. https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/introducing-2017s-extensions-to-the-arm-architecture.

[11] ARM Holdings. ARMv8-A Trusted Board Boot requirements, 2018. https://static.docs.arm.com/den0006/d/DEN0006D_Trusted_Board_Boot_Requirements.pdf.

[12] ARM Holdings. mbedTLS, 2020. https://github.com/ARMmbed/mbedtls.

[13] ARM Holdings. Securing the level one memory system, 2020. https://developer.arm.com/documentation/PRD29-GENC-009492/c/TrustZone-Hardware-Architecture/Processor-architecture/Securing-the-level-one-memory-system.

[14] ARM Holdings. Trusted Firmware-A, 2020. https://github.com/ARM-software/arm-trusted-firmware.

[15] 96 Boards. HiKey LeMaker specifications, 2018. http://www.lemaker.org/product-hikey-specification.html.

[16] Ernie Brickell and Jiangtao Li. Enhanced privacy ID from bilinear pairing for hardware authentication and attestation. *International Journal of Information Privacy, Security and Integrity 2*, 1(1):3–33, 2011.

[17] Tiago Brito, Nuno O Duarte, and Nuno Santos. ARM TrustZone for secure image processing on the cloud. In *IEEE 35th Symposium on Reliable Distributed Systems Workshops*, pages 37–42. IEEE, 2016.

[18] Maxim Chernyshev, Sherali Zeadally, Zubair Baig, and Andrew Woodward. Mobile forensics: Advances, challenges, and research opportunities. *IEEE Security & Privacy*, 15(6):42–51, 2017.

[19] Victor Costan and Srinivas Devadas. Intel SGX explained. *IACR Cryptol. ePrint Arch.*, 2016(86):1–118, 2016.

[20] Munkhzorig Dorjmyagmar, MinChang Kim, and Hyoungshick Kim. Security analysis of Samsung KNOX. In *19th International Conference on Advanced Communication Technology (ICACT)*, pages 550–553. IEEE, 2017.

[21] European Commission. *D1.1. – State of the Art in Mobile Forensics*. The European Commission, 2020.

[22] Genode Operating System Framework. An exploration of ARM TrustZone technology, 2018. https://genode.org/documentation/articles/trustzone.

[23] Miraje Gentilal, Paulo Martins, and Leonel Sousa. TrustZone-backed Bitcoin wallet. In *Proceedings of the Fourth Workshop on Cryptography and Security in Computing Systems*, pages 25–28, 2017.

[24] GlobalPlatform. TEE Client API (v1.0), 2010.

[25] GlobalPlatform. TEE Secure Element API (v1.1.1), 2016.

[26] GlobalPlatform. TEE Sockets API (v1.0.1), 2017.

[27] GlobalPlatform. TEE System Architecture (v1.2), 2017.

[28] GlobalPlatform. TEE Internal Core API (v1.2.1), 2019.

[29] GlobalPlatform. Technology Document Library, 2020. http://globalplatform.org/specs-library/?filter-committee=tee.

[30] GlobalPlatform. TEE Protection Profile (v1.3), 2020.

[31] Javier González, Michael Hölzl, Peter Riedl, Philippe Bonnet, and René Mayrhofer. A practical hardware-assisted approach to customize trusted boot for mobile devices. In *International Conference on Information Security*, pages 542–554. Springer, 2014.

[32] Google. Widevine DRM, 2020. https://www.widevine.com/solutions/widevine-drm.

[33] Huawei Technologies. Huawei iTrustee V3.0 on Kirin 980 Security Target, 2019. https://www.commoncriteriaportal.org/files/epfiles/anssi-cible-cc-2020_67en.pdf.

[34] Huawei Technologies. iTrustee, 2020. https://www.huawei.com/en/sustainability/stable-secure-network/privacy-protection.

[35] Gurjant Kalsi, Christopher Anderson, Brian Swetland, and Travis Geiselbrecht. Little Kernel (LK), 2020. https://github.com/littlekernel/lk.

[36] Vishal Karande, Erick Bauman, Zhiqiang Lin, and Latifur Khan. SGX-Log: Securing system logs with SGX. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 19–30, 2017.

[37] Steven L Kinney. *Trusted platform module basics: using TPM in embedded systems*. Elsevier, 2006.

[38] Roland Kunkel, Do Le Quoc, Franz Gregor, Sergei Arnautov, Pramod Bhatotia, and Christof Fetzer. TensorSCONE: A secure TensorFlow framework using Intel SGX. *arXiv preprint arXiv:1902.04413*, 2019.

[39] Wenhao Li, Haibo Li, Haibo Chen, and Yubin Xia. Adattester: Secure online mobile advertisement attestation using TrustZone. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 75–88, 2015.

[40] LibTom. LibTomCrypt, 2020. https://github.com/libtom/libtomcrypt.

[41] Linaro. A gentle introduction to trusted execution and OP-TEE, 2016. https://connect.linaro.org/resources/bkk16/bkk16-110/.

[42] Linaro. Frequently asked questions, 2020. https://optee.readthedocs.io/en/latest/faq/faq.html.

[43] Linaro Ltd. Trusted Firmware Open Governance Project, 2020. https://www.trustedfirmware.org/.

[44] Hans Löhr, Ahmad-Reza Sadeghi, and Marcel Winandy. Patterns for secure boot and secure storage in computer systems. In *International Conference on Availability, Reliability and Security*, pages 569–573. IEEE, 2010.

[45] Tarjei Mandt, Mathew Solnik, and David Wang. Demystifying the secure enclave processor. *Black Hat Conference*, 2016.

[46] Brian McGillion, Tanel Dettenborn, Thomas Nyman, and N Asokan. Open-TEE – an open virtual trusted execution environment. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 400–407. IEEE, 2015.

[47] Microsoft. Microsoft PlayReady enhanced content protection, 2020. https://www.microsoft.com/playready/features/enhancedcontentprotection.aspx.

[48] Nvidia. TLK reference, 2020. http://nv-tegra.nvidia.com/gitweb/?p=3rdparty/ote_partner/tlk.git;a=blob_plain;f=documentation/Tegra_BSP_for_Android_TLK_FOSS_Reference.pdf.

[49] OP-TEE. Core documentation, 2020. https://optee.readthedocs.io/en/latest/architecture/core.html.

[50] OP-TEE. Cryptographic implementation, 2020. https://optee.readthedocs.io/en/latest/architecture/crypto.html.

[51] OP-TEE. Platforms supported, 2020. https://optee.readthedocs.io/en/latest/general/platforms.html.

[52] Open-TEE. Open-TEE, 2020. https://open-tee.github.io/.

[53] Sandro Pinto and Nuno Santos. Demystifying ARM TrustZone: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 51(6):1–36, 2019.

[54] Prove & Run. ProvenCore, 2020. https://www.provenrun.com/products/provencore/.

[55] Qualcomm. File based encryption, 2019. https://www.qualcomm.com/media/documents/files/file-based-encryption.pdf.

[56] Qualcomm. Guard Your Data with the Qualcomm Snapdragon Mobile Platform, 2020. https://www.qualcomm.com/media/documents/files/guard-your-data-with-the-qualcomm-snapdragon-mobile-platform.pdf.

[57] Qualcomm. Inline crypto engine (ufs), 2020. https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/security-policies/140sp3124.pdf.

[58] Quarkslab. A deep device into Samsung's TrustZone, 2019. https://blog.quarkslab.com/a-deep-dive-into-samsungs-trustzone-part-1.html.

[59] RISC-V International. RISC-V, 2020. https://riscv.org/.

[60] Samsung. KNOX white paper: Device health attestation, 2020. https://docs.samsungknox.com/admin/whitepaper/kpe/attestation.htm.

[61] Samsung. Samsung TEEGRIS, 2020. https://developer.samsung.com/teegris/overview.html.

[62] Carlos Segarra, Ricard Delgado-Gonzalo, Mathieu Lemay, Pierre-Louis Aublin, Peter Pietzuch, and Valerio Schiavoni. Using trusted execution environments for secure stream processing of medical data. In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 91–107. Springer, 2019.

[63] Carlton Shepherd. *Techniques for Establishing Trust in Modern Constrained Sensing Platforms with Trusted Execution Environments*. PhD thesis, Royal Holloway, University of London, 2019.

[64] Carlton Shepherd, Raja Naeem Akram, and Konstantinos Markantonakis. EmLog: tamper-resistant system logging for constrained devices with TEEs. In *IFIP International Conference on Information Security Theory and Practice*, pages 75–92. Springer, 2017.

[65] Carlton Shepherd, Raja Naeem Akram, and Konstantinos Markantonakis. Establishing mutually trusted channels for remote sensing devices with trusted execution environments. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, pages 1–10. ACM, 2017.

[66] Carlton Shepherd, Raja Naeem Akram, and Konstantinos Markantonakis. Remote credential management with mutual attestation for trusted execution environments. In *IFIP International Conference on Information Security Theory and Practice*, pages 157–173. Springer, 2018.

[67] Carlton Shepherd, Ghada Arfaoui, Iakovos Gurulian, Robert P Lee, Konstantinos Markantonakis, Raja Naeem Akram, Damien Sauveron, and Emmanuel Conchon. Secure and trusted execution: Past, present, and future-a critical review in the context of the internet of things and cyber-physical systems. In *IEEE TrustCom*, pages 168–177. IEEE, 2016.

[68] Sierraware. SierraTEE trusted execution environment, 2020. https://www.sierraware.com.

[69] SMART Wireless Computing Inc. Protecting your premium HD content with Widevine digital rights management (drm) on Inforce platforms, 2020. https://www.inforcecomputing.com/protecting_with_premium_hd_content/.

[70] TrustKernel. T6 – Secure OS and TEE, 2020. https://www.trustkernel.com/en/products/tee/t6.html.

[71] Trustonic. Trustonic TEEs, 2020. https://www.trustonic.com/technology/.

[72] Meltem Turan, Elaine Barker, William Burr, and Lily Chen. Special publication 800-132: Recommendation for password-based key derivation. *NIST, Computer Security Division, Information Technology Laboratory, Tech. Rep*, 2010.

[73] Eva A Vincze. Challenges in digital forensics. *Police Practice and Research*, 17(2):183–194, 2016.