Qualcomm

Qualcomm Technologies, Inc.

Guard Your Data with the Qualcomm® Snapdragon™ Mobile Platform

Hardware-Backed Mobile Secure Storage

Liang Cai

April 27, 2019

Disclaimer

Qualcomm Technologies, Inc.

Qualcomm Snapdragon, Qualcomm Trusted Execution Environment, Qualcomm Secure Storage Solutions, Qualcomm Secure Processing Unit, Qualcomm Secure File System, and Qualcomm Fast Trusted Storage are products of Qualcomm Technologies, Inc. and/or its subsidiaries.

Qualcomm and Snapdragon are trademarks of Qualcomm Incorporated, registered in the United States and other countries. Other products and brand names may be trademarks or registered trademarks of their respective owners. The contents of this document are provided on an “as-is” basis without warranty of any kind. Qualcomm Technologies, Inc. specifically disclaims the implied warranties of merchantability and fitness for a particular purpose.

Qualcomm Technologies, Inc.

5775 Morehouse Drive

San Diego, CA 92121

U.S.A.

Contents

Introduction

Overview

Data protection is a well-recognized security requirement for mobile devices, feature designers and mobile app developers. As mobile devices evolve into the most important personal computing platform, they have become the major producer, consumer and storage of user's personal data. The demand to prevent data stored on mobile devices from unauthorized access arises not only from users' expectations of privacy, but also from legal requirements and the reputation of related technology providers. In addition to user data, sensitive information stored on mobile devices can also be from device manufacturers, app developers, cloud service providers, network carriers, financial institutions and so on. It is of the business interest of the whole mobile ecosystem to provide secure storage on mobile devices.

Designing a secure storage scheme is often challenging and error prone. Home-brew secure storage solutions, especially those purely based on software mechanisms, are often vulnerable to attacks. To help address this issue, Qualcomm Technologies, Inc. (QTI) has integrated several hardware-backed secure storage solutions in its mobile SoC products. Each of these solutions are designed with different purposes.

- Secure File System (SFS). SFS provides an encrypted file system for trusted apps in Qualcomm Trusted Execution Environment to store data in the flash storage. Data stored in SFS is only decrypted in memory that Qualcomm Trusted Execution Environment software can access. It provides even stronger data protection than Keymaster-based encryption because the Android system cannot access the plaintext of the SFS data in any state. SFS is useful in protecting highly security sensitive information such as user's biometric data and financial accounts. Keys and data used to protect intellectual property are often stored in SFS as well.
- Fast Trusted Storage (FTS). FTS provides trusted apps in Qualcomm Trusted Execution Environment with similar encryption service. Compared to SFS, it trades random file access for higher data throughput to the secure storage. FTS does not support rollback protection, hence data that is vulnerable to replay attacks should be stored in SFS.
- Qualcomm Trusted Execution Environment RPMB driver (Qualcomm Trusted Execution Environment STOR). RPMB (Rollback Protection Memory Block) is a write protected region on certain flash devices such as eMMC and UFS. Once initialized, this region can only be accessed by trusted apps in Qualcomm Trusted Execution Environment through the Qualcomm Trusted Execution Environment RPMB driver. It is mainly used for storing counters for detecting replay attacks.

- Android Keymaster[1] for cryptographic keys management. Android applications and system modules can encrypt the data using keys managed by Keymaster or utilize Android file encryption features. Keymaster enforces access control rules on the use of keys even when the Android system is compromised. For example, if an application chooses to protect its data with a key that is only available when the user has unlocked the device, an attacker should not be able to decrypt the data when the device is in locked state even if she has the root privilege of the Android system.

In the following sections, we show the limitations of software-based secure storage solutions. After brief descriptions of the hardware building blocks used by these solutions, we introduce the architecture and main features of the first three. Android Keymaster deserves a separate deep dive, which will be publishing in the near future.

Table 1 in the summary section outlines their differences in usage and features.

Acronyms

AES	Advanced Encryption Standard
eMMC	embedded Multimedia Card
FTS	Fast Trusted Storage
GPCE	General Purpose Crypto Engine
HAL	Hardware Abstraction Layer
HMAC	Hashed Message Authentication Code
KDF	Key Derivation Function
ICE	Inline Crypto Engine
IMEM	Internal Memory
PRNG	Pseudo Random Number Generator
REE	Rich Execution Environment
RPMB	Rollback Protection Memory Block
SFS	Secure File System
SHA	Secure Hash Algorithm
SoC	System on Chip
SPU	Secure Processor Unit
TA	Trusted App
TEE	Trusted Execution Environment
UFS	Universal Flash Storage

Hardware support for secure storage

As mentioned in the introduction, secure storage solutions purely based on software have certain limitations that we address through our hardware-based features. In this section we provide an overview of limitations of software-based secure storage solutions. We also discuss hardware building blocks necessary to support secure storage on our SoCs.

Limitation of pure software-based solutions

Secure storage systems that are based on pure software mechanisms lack important hardware security features and, therefore, expose the data to a broader range of threats.

Runtime protection

One common attack on encryption based secure storage is to exploit software vulnerabilities and access keys or decrypted data directly from the memory. Alternatively, an attacker can inject malicious code or change the execution flow to circumvent access control policies. Either attack allows unauthorized access to keys or data in the memory. In many use cases, the device designer or an app developer does not want certain data to be leaked even to a legitimate device user, who might root the device to gain full control.

To address the above problems, the runtime of the secure storage system needs to be protected from both attackers and users. This is typically achieved by moving the secure storage software to a hardware supported Trusted Execution Environment (TEE). The solutions described in this white paper are built using such hardware-backed TEEs.

Key storage

Another challenge for encryption based secure storage systems is the question where to store the cryptographic keys. No matter how many times the data encryption key is wrapped by other keys, it is vulnerable to reverse engineering attacks if the outmost key is hard-coded. If the key originates from a cloud service, there is a similar issue of where to store the secret required to bootstrap the device-cloud communication. Encryption with user credential-derived key is often vulnerable to brute-force or dictionary attacks and does not protect data which should be invisible to the users.

The problem is easier to solve if hardware support is available. One solution is to derive the wrapping key from a hardware key that is unique per device. The key derivation occurs in TEE such that the plaintext of the key doesn't leak.

Rollback prevention

Data encryption software cannot prevent roll-back attacks, which allow an attacker to reinstate a compromised key, downgrade to an older vulnerable software image, or disable access control by resetting the state of the storage. We need to utilize hardware to store versions or state values securely.

Device binding

Another related feature that requires hardware support is device binding. It is a critical security feature for secure storage systems, which protects data from a compromised "global" key. Device binding is based on a per-device unique identity which is baked into the hardware and cannot be altered by software after the devices are initialized.

All solutions described in this paper are based on device-bound hardware keys.

Random number generation

Software-based pseudo random number generators (PRNG) are dependent on the quality of external entropy source. On their own, they have no entropy and, therefore, lower the effort for cryptoanalysis attacks.

Hardware building blocks

The secure storage solutions described in this paper are built on top of the following hardware components: Trusted Execution Environment, Hardware Crypto Engine, and Anti-Replay Protection.

Trusted Execution Environment

Mobile operating systems, such as Android, offer a Rich Execution Environment (REE), providing a hugely extensive and versatile runtime environment. While bringing flexibility and capability, REE leaves devices vulnerable to a wide range of security threats. The TEE is designed to reside alongside the REE and provide a safe area on the device to protect assets and to execute trusted code.

Qualcomm Trusted Execution Environment

The TEE on Qualcomm Technologies SoC is based on ARM TrustZone technology. TrustZone is a set of security extensions on ARM architecture processors providing a secure virtual processor backed by hardware-based access control. This secure virtual processor is often referred to as the "secure world", in comparison to the "non-secure world", where REE resides. Software running on TrustZone consists of the Qualcomm Trusted Execution Environment platform (Qualcomm Trusted Execution Environment kernel, libraries and services) and Trusted apps. Qualcomm Trusted Execution Environment provides software support for chipset security and exposes hardware abstraction layer (HAL) APIs for chipset security functions such as crypto, RNG and fuse blocks. Qualcomm Trusted Execution Environment is also responsible for initializing the system security environment for software and hardware. Qualcomm Trusted Execution Environment provides security services, such as binary loading, authentication, crypto and logging to

secure software modules, called Trusted Apps (TAs), which are dynamically loaded and executed in TrustZone. As shown in Figure 1, transition of execution from the non-secure world to the secure world must be handled by a component that runs in secure monitor mode. The monitor component guarantees the context of the secure world to be segregated from that of the non-secure world.

The software running in Qualcomm Trusted Execution Environment is exposed to lower security risks. As part of the security goals, anyone who does not have the device hardware keys must not be able to access Qualcomm Trusted Execution Environment data and services unless they are intentionally exposed. This is supported by various countermeasures:

1. Qualcomm Trusted Execution Environment is a software framework with a much smaller footprint, and so is the attack surface.
2. TAs running in Qualcomm Trusted Execution Environment must be signed and authenticated when they are loaded. The Qualcomm Trusted Execution Environment platform (including Qualcomm Trusted Execution Environment kernel and Qualcomm Trusted Execution Environment libraries/services) itself is signed and loaded by the boot loader during the initial device bootup process.
3. Separation between TEE and REE is enforced by hardware-based access control. For example, sensitive data of Qualcomm Trusted Execution Environment and TAs are located either on internal memory or an encrypted, integrity protected memory region called Pseudo-IMEM (PIMEM).
4. Debug support for Qualcomm Trusted Execution Environment software is separated from that for non-secure software.

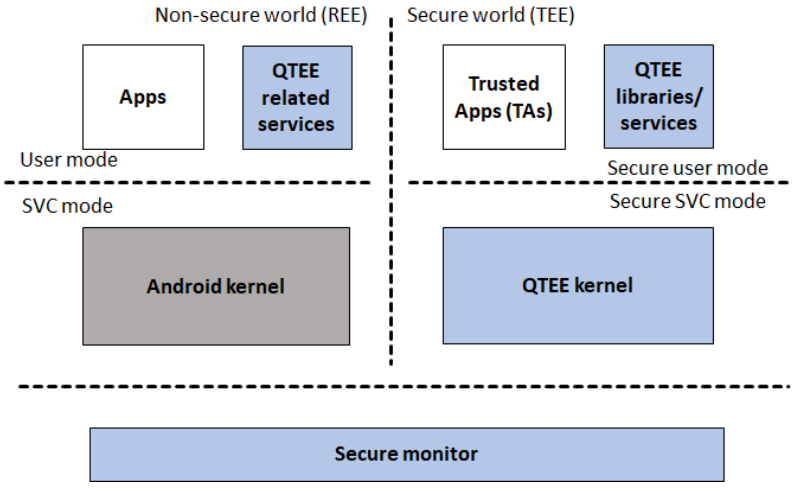

Figure 1: ARM v8 Software architecture

SPU

Qualcomm® Secure Processing Unit provides a physically segregated TEE based on a secure processor integrated into some Qualcomm Technologies SoCs. The Secure Processing Unit (SPU) hardware is a dedicated subsystem that is engineered to provide an independent boot-loader and boot chain, dedicated clocks, hardware-based anti-replay protection, a key management unit, and a crypto management unit with inline crypto accelerators. Operating condition sensors are integrated into the subsystem to help prevent power attacks. Its side-channel resistant crypto includes masking and blinding. SPU is Common Criteria EAL 4+ certifiable and designed to meet the security requirements of the Android P Strongbox feature.

Hardware Crypto Engine

There are two type of hardware crypto engines in Qualcomm Technologies SoCs that contribute to the secure storage solutions.

GPCE

QTI General Purpose Crypto Engine (GPCE), also known as the Crypto core, is a FIPS 140-2 certified coprocessor design that provides hardware acceleration of standard cryptographic algorithms, such as AES and SHA. TrustZone can program the Crypto core to provide crypto services to software running both in REE and TEE.

One major security gain by using GPCE is that cryptographic operations can use hardware keys from fuse blocks that never need to be accessed by software. This is necessary to enable device bound encryption as the device unique key can only be used on the device. In addition to hardware keys, software key set can be stored in the internal memory of GPCE. Access to these keys is restricted by access control rules.

ICE

While GPCE provides an acceptable performance, it falls short as the general storage throughput increases. To overcome performance degradation, another FIPS 140-2 certified hardware crypto engines — the inline crypto engine (ICE) — was introduced on recent Qualcomm Technologies Mobile SoCs to help achieve high throughput cryptographic encryption of storage data. ICE is mainly used by the Android file and disk encryption features.

Anti-Replay Protection

We use hardware mechanisms, such as RPMB, to help prevent replay attacks in the secure storage solutions.

RPMB

The Replay Protected Memory Block (RPMB) is a separated partition on eMMC or UFS devices designed for secure data storage. Since each access to RPMB must be authenticated, this region is engineered to be write protected from software entities without the authentication key. This allows the device to defend against rollback or replay attack by storing versions or counts in this region.

Access to RPMB is authenticated by a message authentication code (MAC), which is a hash value generated in Qualcomm Trusted Execution Environment from a 256-bit authentication key. This key must be provisioned to the eMMC or UFS device in Qualcomm Trusted Execution Environment before any access to RPMB.

Others

In addition to the hardware building blocks described above, Qualcomm® Secure Storage Solutions also utilize several other hardware components such as fuse blocks, the hardware key derivation function (KDF), and the random number generator (RNG).

In the following chapters we will describe hardware-backed Qualcomm Secure Storage Solutions, which address the shortcomings of software-based approaches and use the above-mentioned hardware building blocks.

Qualcomm Trusted Execution Environment Secure File System

Overview

On Qualcomm Technologies SoCs, runtime protection of sensitive data is provided by Qualcomm Trusted Execution Environment Secure File System (SFS). It is a file encryption service only available to TAs in Qualcomm Trusted Execution Environment. SFS can also be used for securing crypto keys, which are designed to be protected from a compromised or rooted Android.

In Qualcomm Trusted Execution Environment SFS, data is engineered to be encrypted with AES256-CTR-mode and integrity protected with HMAC-SHA256. Figure 2 shows the software architecture of Qualcomm Trusted Execution Environment SFS. The SFS API, also known as Persistent Object API, is compliant with GlobalPlatform Device Technology TEE Internal Core API Specification. It allows Tas to create, enumerate and delete persistent data objects, as well as perform data stream access such as read, write, truncate and seek. Persistent objects are structured into three parts and stored in Android File system and the RPMB partition:

- object files which contains sectioned TA data, version, IV for encryption and the HMAC of the file;
- a per TA Index file which contains metadata (file name and version) of each file;
- RPMB data which contains SHA256 hash of all indexed data.

Since Qualcomm Trusted Execution Environment has no direct access to external storage, there are two Qualcomm Trusted Execution Environment listeners running in REE to proxy SFS access to the physical storage. Listeners are daemons or services running in Android. They receive requests from Qualcomm Trusted Execution Environment, perform operations and return the results back to Qualcomm Trusted Execution Environment.

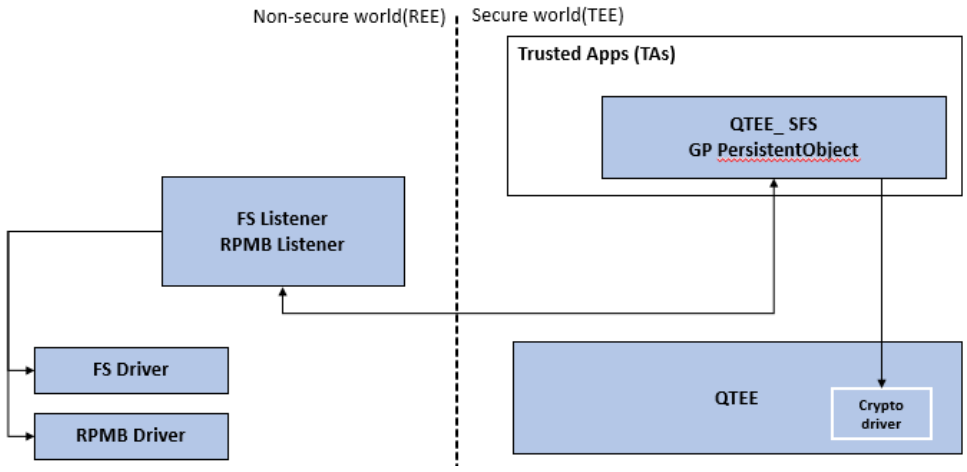

Figure 2: Qualcomm Trusted Execution Environment Secure File System software architecture

Features

Qualcomm Trusted Execution Environment SFS has the following security features:

Client Sandboxing

SFS is designed to ensure no TA can decrypt another TA's files. Keys used for encrypting data of each TA are derived from the hardware key using the TA identifier as the input. The TA identifier is cryptographically bound to the TA binary and stored in Qualcomm Trusted Execution Environment platform when the TA binary is authenticated. No TA can spoof another TA by crafting the TA identifier.

Rollback Protection:

A data object has an associated version which can only increase from creation at each modification. These versions are stored within both the data file and the index file. The version and HMAC value of the index file is stored in the RPMB. If an attacker tries to replace a file with an older version, SFS is engineered to detect the version mismatch in the decrypted data segment and the index file. If the attacker tries to also replace the index file with an old version, SFS can detect the mismatch between the HMAC of the index file and HMAC stored in RPMB.

Atomicity

Any modification to an SFS file is guaranteed to be an atomic operation. The old data file and index record will not be deleted until the operation is complete, A dirty flag is set in RPMB during the operation so that SFS can return to a valid state in the event of attacks or a power loss.

Metadata encryption

Data files are stored in segments with fixed length, the file name, version and actual file size can only be found in the index file, which is also encrypted. Such information cannot be inferred from the Android file system.

Qualcomm Trusted Execution Environment Fast Trusted Storage

Overview

Qualcomm Trusted Execution Environment Fast Trusted Storage (FTS) is an alternative file encryption solution available to TAs that require higher throughput. It is based on the same Qualcomm Trusted Execution Environment File IO API and crypto API as those supporting SFS (Secure File System). FTS does not provide a transparent file system interface, so the clients need to call the `FileIO` API to access the file system and the crypto API to encrypt/decrypt the data separately. The software architecture of FTS is shown in Figure 3. It shares SFS's design in many aspects:

- using the same crypto parameters as SFS: AES256 CTR mode and HMAC SHA256;
- using the same per TA keys;
- accesses the file system through the same listener service.

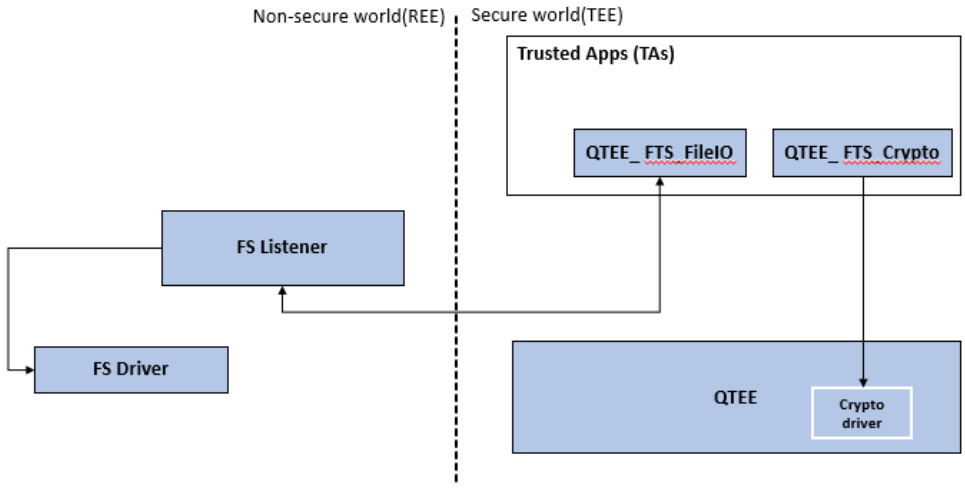

Figure 3: Qualcomm Trusted Execution Environment FTS software architecture

Features and Limitations

FTS is mainly designed for high file throughput with baseline security features. FTS provides similar protection on confidentiality and integrity to TA data, as well as the client sandboxing. But it does not provide replay attack protection and metadata encryption. It also does not support random file access. The data on the file system must be operated as a whole file.

Qualcomm Trusted Execution Environment RPMB Driver

Overview

None of the above solutions have the capacity to prevent encrypted files from being deleted. However, erasure protection is useful in certain scenarios such as device factory reset. Qualcomm Trusted Execution Environment RPMB Driver provides an interface to TAs in Qualcomm Trusted Execution Environment to store data to the RPMB region of the storage. As we discussed earlier, since all accesses to RPMB must be authenticated and the MAC (message authentication code) can only be calculated in Qualcomm Trusted Execution Environment, any attempt to read or write any data in RPMB from the non-secure world will fail.

The software architecture of the Qualcomm Trusted Execution Environment RPMB driver, also known as QTEE_STOR, is shown in Figure 4. QTEE_STOR is part of the Qualcomm Trusted Execution Environment SDK and it provides TAs an API to create, read and write RPMB partitions. SD Manager is a Qualcomm Trusted Execution Environment service which communicates with the RPMB Listener service in the non-secure world and deals with the RPMB specifics. The RPMB Content Generator component receives RPMB access requests from SD Manager and creates the corresponding RPMB packets. The MAC of each RPMB packet is calculated by the RPMB Content Generator using the GPCE (General Purpose Crypto Engine). When a response frame from the storage device is received, SD Manager sends it to RPMB Content Generator for authentication and data extraction.

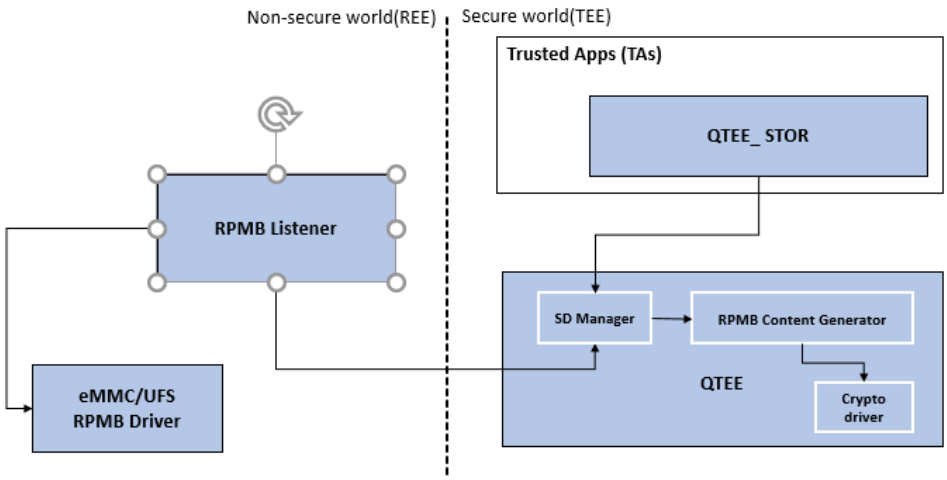

Figure 4: Qualcomm Trusted Execution Environment RPMB Driver software architecture

Features and Limitations

Client Sandboxing

SD Manager extracts an 8-byte secure application name ID (SANID) from the identifier of the calling TA and associates it with a partition when it is created. Only the TA with a matching SANID is allowed to access the partition. Note that the TA identifier is cryptographically bound to the device and TA binary, so TA's can't spoof other TA's by crafting the SANID.

Provisioning

To use the RPMB feature, a key must be provisioned to the storage device. This key is randomly generated per device. It cannot be altered or deleted. Also, several fuses must be blown to set the device state appropriately.

Limitation

The capacity of RPMB is decided by the manufacturer of the storage device. It is often not big enough for storing large data objects. Neither should it be opened to all TAs. The data stored in the RPMB is not encrypted so it could be leaked by a compromised REE or tapping on the bus between the storage device and the SoC. Therefore, TAs using RPMB for storing confidential data such as keys must encrypt it before storing in RPMB.

Summary

We introduced several hardware based secure storage solutions available on Qualcomm Technologies mobile platforms, each designed for different scenarios:

Secure File System (SFS)

Qualcomm® Secure File System provides a transparent encrypted file system to all trusted apps in Qualcomm Trusted Execution Environment. The API exposed by Secure File System (SFS) is compliant to the `PersistentObject` interface in Global Platform TEE Internal Core API Specification[2].

Fast Trusted Storage (FTS)

Qualcomm® Fast Trusted Storage is an alternative secure storage solution available in Qualcomm Trusted Execution Environment for use cases requiring higher file access throughput. However, FTS does not provide a standard File IO interface. Neither does it support random file access, metadata encryption or rollback prevention.

RPMB driver (QTEE_STOR)

RPMB driver, also known as QTEE _STOR, allows trusted apps in Qualcomm Trusted Execution Environment to access the Replay Protected Memory Block (RPMB), which is a separated partition on eMMC or UFS devices designed for secure data storage. RPMB is the only currently-available secure storage solution with erasure protection. Due to its limited size, RPMB should be limited to specific use cases.

Table 1 shows a brief comparison of features and potential clients of the above secure storage solutions. Note that all information in this white paper is based on Qualcomm® Snapdragon™ 855 mobile platform, unless otherwise indicated.

Solution	Clients	Features
Qualcomm Trusted Execution Environment SFS	• Trusted Apps in Qualcomm Trusted Execution Environment	• Device binding • Data Confidentiality and integrity • Rollback protection • Complete file system interface • Metadata encryption • Client sandboxing • Runtime protection on data in memory • Runtime protection on keys in memory
Qualcomm Trusted Execution Environment FTS	• Trusted Apps in Qualcomm Trusted Execution Environment (not vulnerable to replay attacks)	• Device binding • Data Confidentiality and integrity • Client sandboxing • Runtime protection on data in memory • Runtime protection on keys in memory • High throughput
Qualcomm Trusted Execution Environment RPMB Driver	• Replay prevention • Limited Trusted Apps in Qualcomm Trusted Execution Environment	• Device binding • Client sandboxing • Write/erasure protection • Runtime protection on data in memory

Table 1: Comparison of three secure storage solutions

References

[1] “Hardware-backed Keystore”, Android Open Source Project.

[2] “TEE Internal Core API Specification v1.1.2” , Global Platform specification.