Qualcomm Technologies, Inc.

SDM670/SDM710 TrustZone and Security Overview

80-PD126-3 Rev. C

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision History

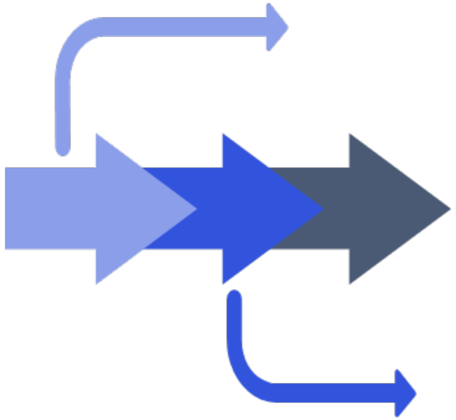
Revision	Date	Description
A	August 2017	Initial release
B	September 2017	Updated Slide 10, description for Iris secure camera
C	April 2018	Updated the title and slide 11 to include the SDM710 chipset

Contents

- Objective
- Product Features
- TrustZone
- Qualcomm Access Control
- pIMEM
- Crypto
- HLOS Security
- Authentication Frameworks
- Content Protection
- References
- Questions?

Objective

- This document serves as an entry point for security for OEMs to understand the architectural and project planning information
- Provides information on new features, product improvements, and helps in understanding of:
  - Chipset and SDM chipsets-level hardware architecture
  - Software architecture
  - Hardware and software migration details to help plan for the development effort
  - High-level software design
  - Reference documents that provide more details



Product Features

Security Summary

Software feature	Description
Secure boot	Three certificate chain model: ▪ SHA256 and SHA384 signing with RSA-PSS padding ▪ OEM customizable root key hash ▪ Xbl_sec boot loader image for secure access control setup
Qualcomm® Trusted Execution Environment (QTEE) (formerly known as Qualcomm Secure Execution Environment (QSEE))	Secure OS that runs in the ARMv8-A hardware-based TrustZone (TZ) security environment
Trusted applications (TA)	Dynamically loadable secure applications that run on QTEE
Pseudo-IMEM (pIMEM)	Encrypted DDR region to execute TZ secure OS and trusted applications
Keymaster (KM)	Keystore services that support public and private key generation, management, signing, and verification
Secure file system (SFS)	Encrypted file storage for TZ and modem
Access control	▪ Slave side access control for register address (fixed or dynamic) memory regions ▪ Master side access control via Stage-2 memory management unit (MMU)
Debug security (formerly known as secure debug)	Re-enable debug on fuse blown devices
Content protection	PlayReady, Widevine, HDCP 2.x, and ISDB-T full segment

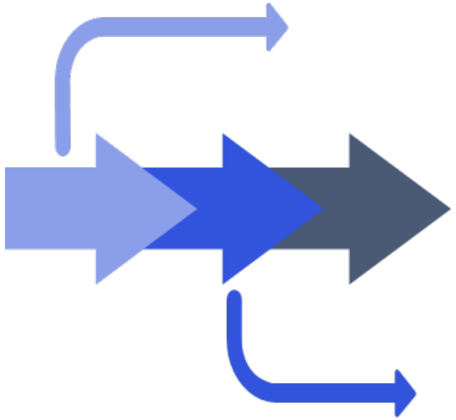
Security Summary (cont.)

Software feature	Description
Device encryption	Full disk encryption, file-based encryption
Authentication frameworks	▪ Trusted UI (formerly known as Secure UI) ▪ Secure Payment ▪ Camera Security (formerly known as Secure Camera) ▪ Qualcomm® Malware Detection (formerly known as Qualcomm Snapdragon Smart Protect)
Android verified boot	Cryptographic authentication of high-level operating system (HLOS) images

Hardware feature	Description
Crypto core	AES, Triple-DES, SHA cryptographic algorithms implemented in device hardware
Inline crypto engine (ICE)	Hardware encryption for storage devices
Fuse memory	▪ One-time programmable QFPROM fuse memory ▪ Automatic programming via sec partition

Security Migration Guide

Change	Description	Migration	Impact
Key management: Keymaster 3.0	Format change for Keymaster 3.0	▪ Keys generated with Keymaster v1.5 or v2 are not compatible with Keymaster v3 ▪ Wipe user data to upgrade	High
TZ: Trusted application improvements	▪ Improved dynamic linking with Commonlib ▪ Heap and stack overflow check with addition of guard pages	build files	High
Biometrics: Iris secure camera	End to end secure camera solution with Qualcomm Spectra™ ISP hardware support	▪ Minor changes to enable and test this feature ▪ Porting guide is provided	Low

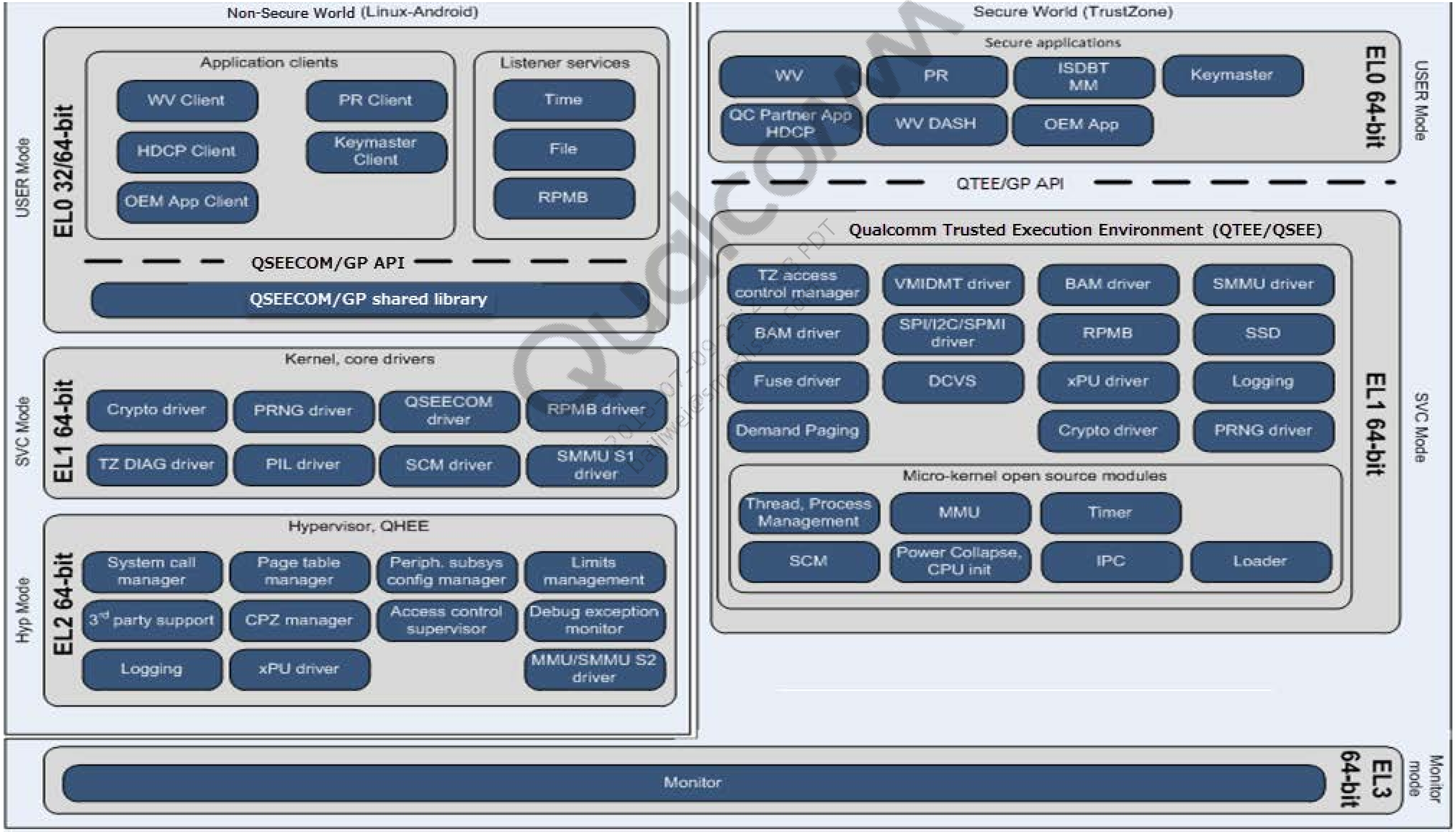


TZ

TZ Overview

- The SDM670/SDM710 chipsets provide a 64-bit ARMv8-A processor system, including Qualcomm® Kryo™ CPU with hardware virtualization
- TZ is a hardware-based security environment through a Secure mode of the ARM processor
  - Operating system runs in Non-secure EL1
  - Transition from Non-secure to Secure mode occurs via a Secure Monitor mode
- TZ software consists of TZ board support package (TZBSP) and QTEE components
  - TZBSP
    - Provides software support for chipset security
    - Exposes hardware abstraction layer (HAL) APIs for chipset security functions such as crypto, fuse block, and PRNG
    - Initializes the system security environment for software and hardware during bootup and wake-up from power collapse
    - Provides memory and other subsystem protection and services during runtime
  - QTEE provides security services, such as image loading, authentication, cache management, crypto, logging, and QFPROM to TZ secure applications
- During the initial device bootup, the XBL loader loads the TZ software and the XBL SEC authenticates it
- Pseudo-IMEM (PIMEM) provides a 64 MB encrypted double date rate (DDR) region to execute TZ and trusted applications

QTEE Framework Block Diagram



Non-Secure World (Linux-Android)

USER Mode

EL0 32/64-bit

Application clients

WV Client

PR Client

HDCP Client

Keymaster Client

OEM App Client

Listener services

Time

File

RPMB

QSEECOM/GP API

QSEECOM/GP shared library

SVC Mode

EL1 64-bit

Kernel, core drivers

Crypto driver

PRNG driver

QSEECOM driver

RPMB driver

TZ DIAG driver

PIL driver

SCM driver

SMMU S1 driver

Hyp Mode

EL2 64-bit

Hypervisor, QHEE

System call manager

Page table manager

Periph. subsys config manager

Limits management

3rd party support

CPZ manager

Access control supervisor

Debug exception monitor

Logging

xPU driver

MMU/SMMU S2 driver

Secure World (TrustZone)

Secure applications

WV

PR

ISDBT MM

Keymaster

QC Partner App HDCP

WV DASH

OEM App

EL0 64-bit

USER Mode

QTEE/GP API

Qualcomm Trusted Execution Environment (QTEE/QSEE)

TZ access control manager

VMIDMT driver

BAM driver

SMMU driver

BAM driver

SPI/I2C/SPMI driver

RPMB

SSD

Fuse driver

DCVS

xPU driver

Logging

Demand Paging

Crypto driver

PRNG driver

Micro-kernel open source modules

Thread, Process Management

MMU

Timer

SCM

Power Collapse, CPU init

IPC

Loader

EL1 64-bit

SVC Mode

Monitor

EL3 64-bit

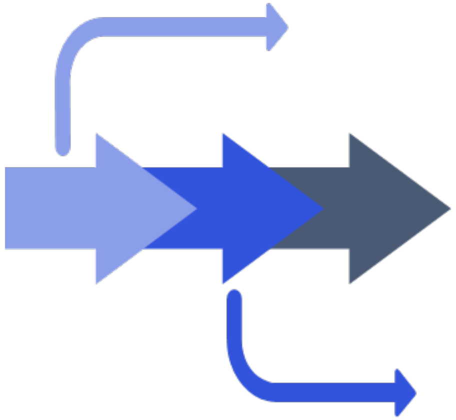
Monitor mode

Trusted Applications

- Supports trusted applications that run in secure world EL0
- Provides services used by clients on the non-secure HLOS side
- QTEE creates guard pages for heap and stack overflow check
- As a template and reference to develop new trusted applications, sampleapp provided is:
  - Used to provision RPMB and test QTEE APIs
  - Invoked by its HLOS counterpart qseecom_sample_client
  - For build sampleapp use:
    - cd trustzone_images/build/ms
    - python build_all.py -b TZ.XF.5.0 CHIPSET=sdm670 --cbt=sampleapp
  - RPMB key provisioning example: qseecom_sample_client v smplap64 14 1
- The smplserv and smplcert trusted applications are provided as a reference for the QTEE IPC mechanism
- The smplcert trusted application uses services that smplserv provides

Secure File System Overview

- Secure file system (SFS) is provided to store sensitive data such as keys and biometric data
- There are two independent SFSs, one on the TZ and other on the modem
- After devices are enabled for secure boot, the two secure file systems use different hardware crypto engines for file data encryption and decryption so that they are secure from each other
- SFS anti-rollback protection feature is enabled by default
- Anti-rollback protection is implemented through eMMC/UFS RPMB hardware support of the storage and an RPMB driver in TZ
- During development, OEMs must use sampleapp TrustZone applications to provision their devices with a test RPMB key
- On commercial devices, a production RPMB key is provisioned automatically by TZ after it detects secure boot and JTAG disables eFuses
- The RPMB key of a device can only be provisioned once to ensure SFS security; thus the device already provisioned with a test key cannot be provisioned again with a production key
- OEMs must plan their development devices carefully due to the mentioned restrictions
- OEMs can enable or disable the RPMB-based SFS anti-rollback protection in the XML configuration file trustzone_images/core/securemsm/trustzone/qsee/mink/oem/config/common/cmnlib_oem_config.xml



Qualcomm Access Control

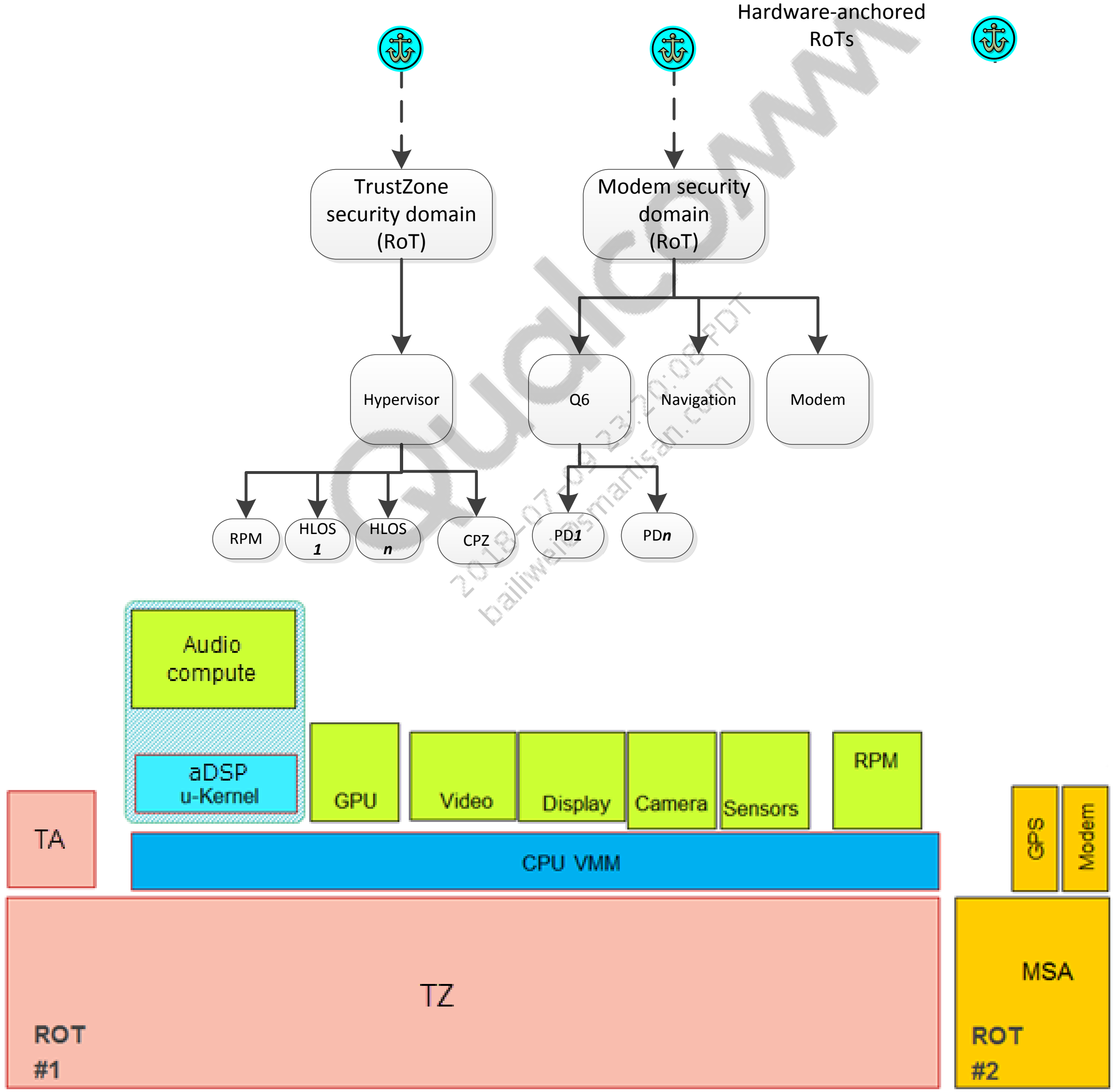
Qualcomm Access Control Overview

- Independent trust model for TZ and MSA (modem)
- TZ and modem manage xPUs to enforce access control policies governing the modem and the TZ assets
- Persistent security and access control configurations
- xPU provides slave side access control for register/fixed address/dynamic memory regions
- Master-side access control for content protection use cases
- Memory management on security domains is restricted to the owner of the domain or an ancestor in the trust hierarchy
- Dynamic memory sharing across security domains and execution environments

Access Control Domains

- There are three levels of ownership:
  - TZ manages TZ domains
  - MSA manages MSA modem domains
  - Hypervisor (EL2) manages legacy non-secure domains including content protection zone (CPZ)
- TZ and MSA provide slave-side access control via xPUs
- Hypervisor configures master-side access control through second-stage page tables via MMUs
- Requires universal 4 KB separation of system resources

Access Control – Trust Hierarchy and Security Domain

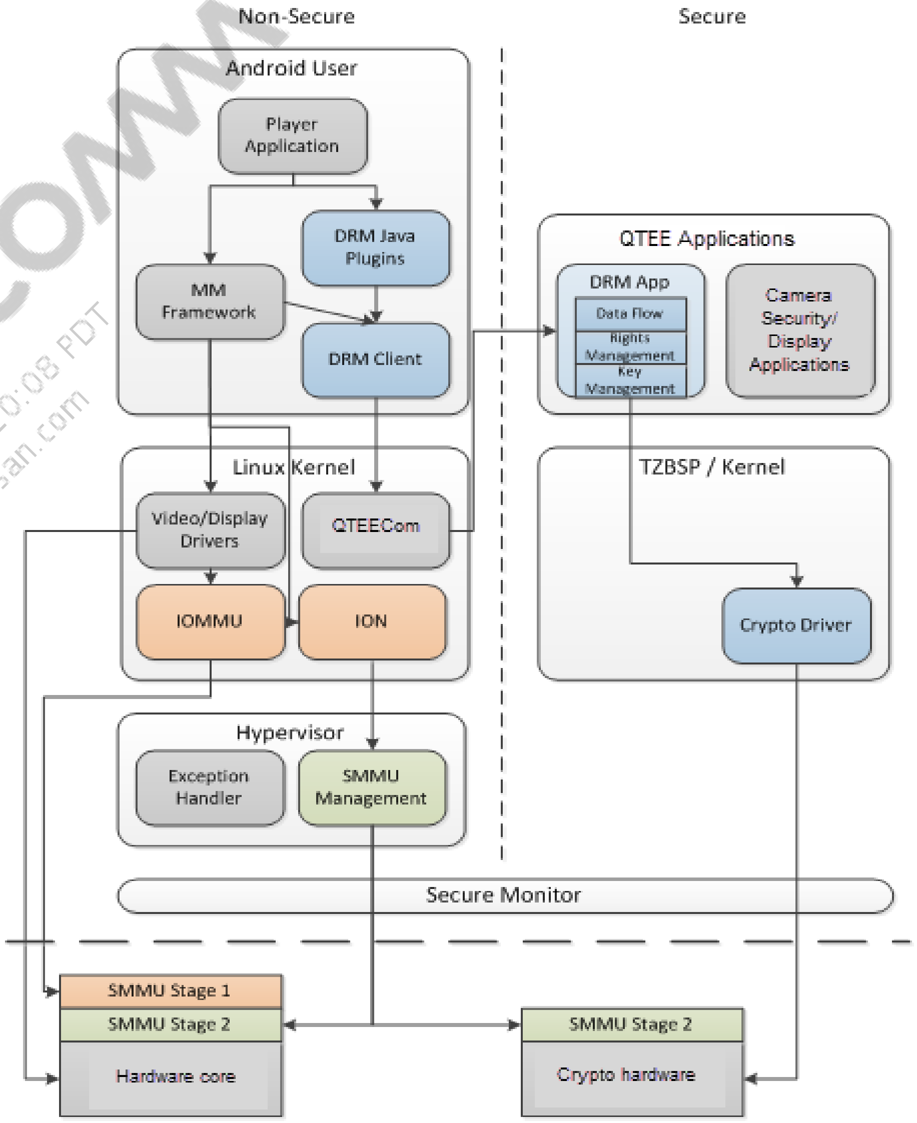


System MMU

- Hypervisor manages stage 2 MMUs and system memory management units (SMMU) to enforce policies that protect the assets in virtual machines (VMs)
- HLOS kernel manages stage 1 MMUs and SMMUs to enforce policies that protect the assets in applications that run in the user privilege level

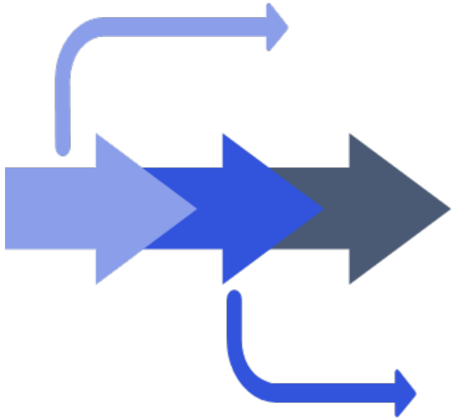
Access Control – CPZ

- Master side content protection features
  - Builds on master side access control
  - Uses MMU stage 2 to control access to protected buffers
  - Drivers remain in HLOS
  - MMU management in Hypervisor
  - DRM remains in TZ
- Key benefits
  - Dynamically allocated buffers, and no carve outs
  - Scales to large memory sizes



Secure Peripheral Image Loading

- During boot, hypervisor protects the subsystem start addresses, remaps registers, and resets control
- PIL driver implemented in HLOS kernel loads image into DDR and calls into Hypervisor
- Secure PIL calls are trapped in Hypervisor, and are forwarded to TZ for image authentication
- Hypervisor updates MMU stage 2 to protect the image, HLOS can no longer access the image
- Modem PIL is handled by modem boot authenticator (MBA) image



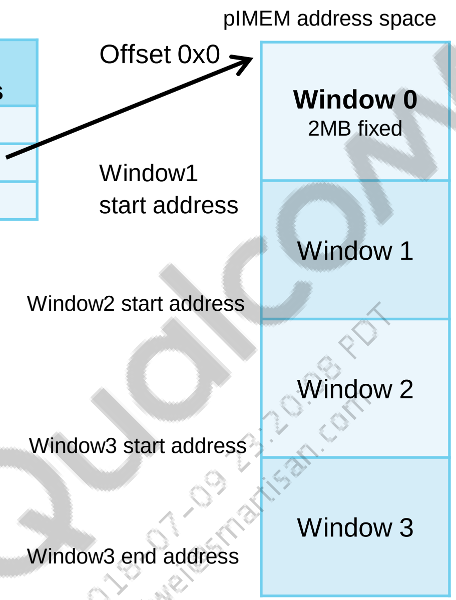
pIMEM

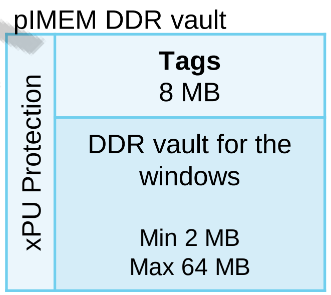
pIMEM DDR Vaults

- pIMEM is pseudo-IMEM that is backed by a secure region in DDR
- Read/write operations go through crypto-hardware
- Motivation
  - Increases the size of the TZ kernel image due to 64-bit migration
  - Provides additional memory to implement OEM proprietary features
  - Runs TZ apps (for example, DRM) in physical attack-resistant memory
  - DDR vulnerable to physical attacks

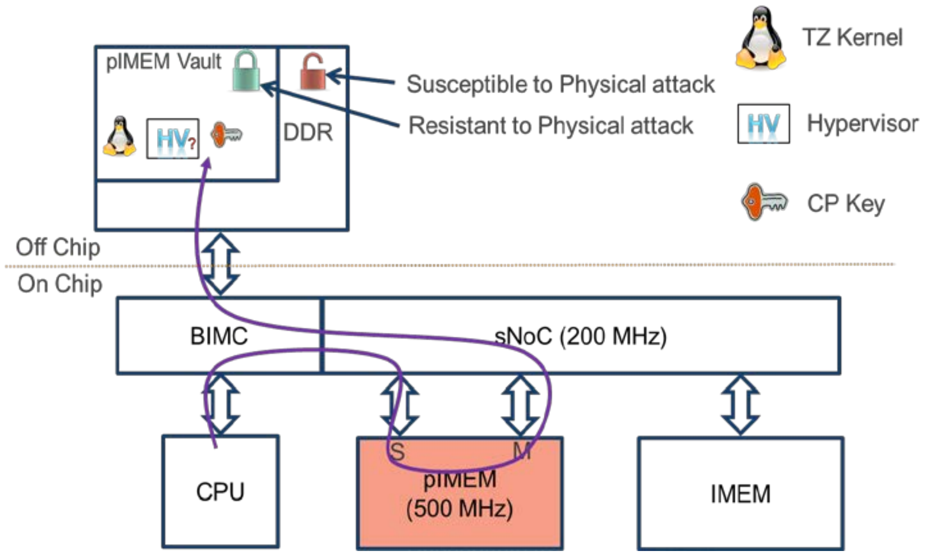
pIMEM – Architecture

Start address	End address	Size	Remarks
180000000	27FFFFFFF	4 GB	DDR
80000000	17FFFFFFF	4 GB	DDR
1C000000	1FFFFFFF	64 MB	pIMEM





- pIMEM related cryptographic operations implemented in a hardware block
- pIMEM connected to system NoC via QSB master and slave ports



Crypto

Crypto Implementations

- QTI crypto core provides cryptographic algorithms implemented in device hardware
  - Provides register based access
  - Includes: AES, Triple-DES, and SHA
  - Provides access to hardware crypto through Linux kernel crypto driver (qcrypto)
  - QTEE provides crypto APIs to trusted applications (qsee_SW_Cipher_*)
- ARMv8-A crypto extension instructions enabled with HLOS kernel configuration
- PRNG and crypto hardware blocks are Federal Information Processing Standards (FIPS) 140-2 certified

Inline Crypto Engine

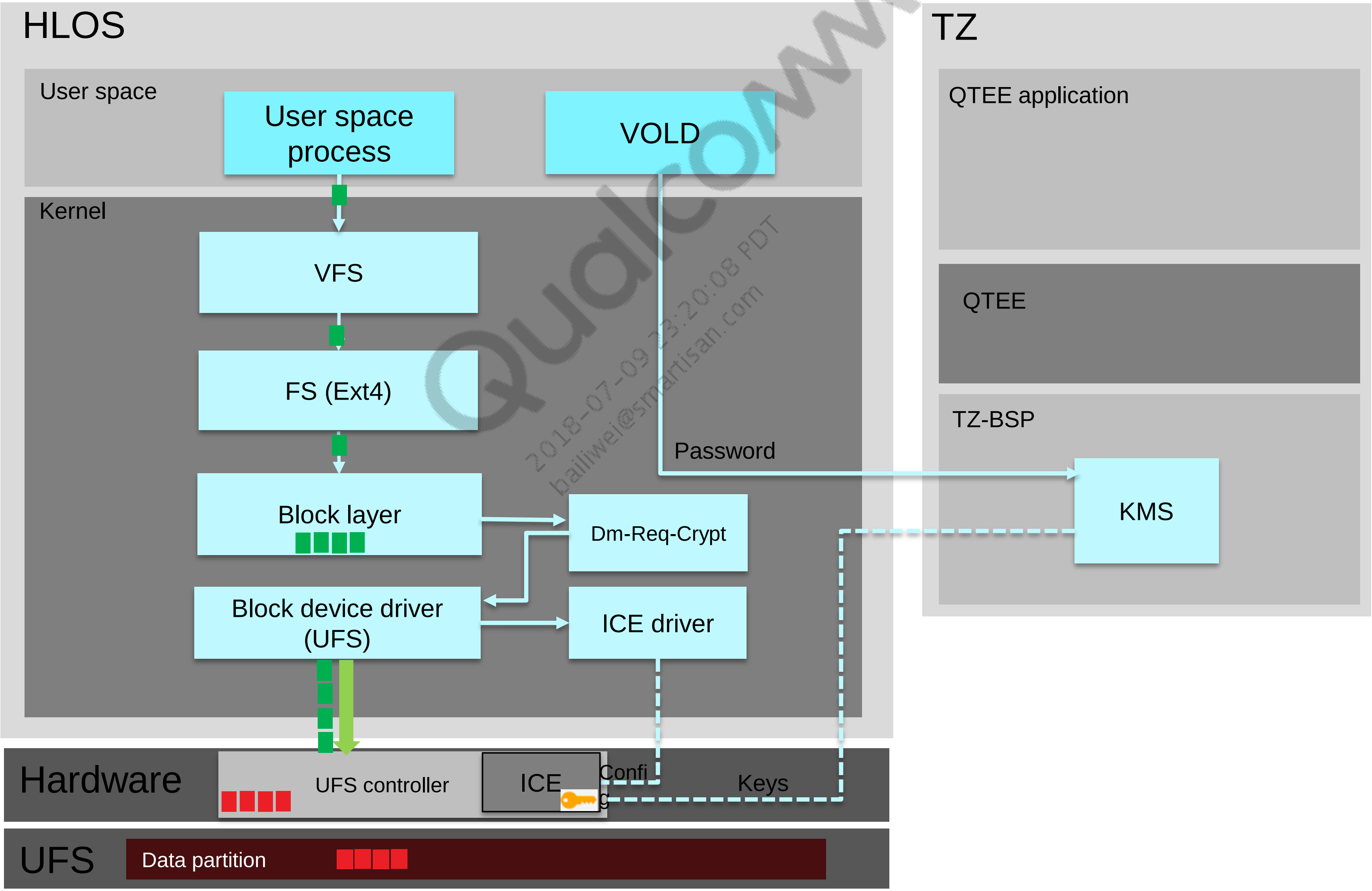
- Inline crypto engine 3.0 (ICE 3.0) is intended for high throughput cryptographic encryption of storage data
- Supports AES 128/256 ECB/XTS
- Supports multiple crypto streams to meet high throughput
- Multiple AES cores per crypto stream
- 32 software-configurable keys
- Asymmetric and symmetric operations

HLOS Security

ICE − Full Disk Encryption

- Android provides a software-based mechanism to protect user data through disk encryption
- Currently, it supports encryption of a user data partition and any non removable SD card, and does not provide encryption functionality for any other partition
- Android disk encryption is based on the dm-crypt Linux kernel module
- It uses the mapped device feature of the kernel and works at the block layer
- QTI developed an ICE to provide a significant boost to encryption throughput
- With a hardware-based solution, the ICE provides the following benefits:
  - Improves performance
  - Reduces power consumption through hardware crypto
  - Enhances security as the crypto key is not stored in RAM, which can potentially be dumped by hackers
- Android supports new pin less and force encryption features

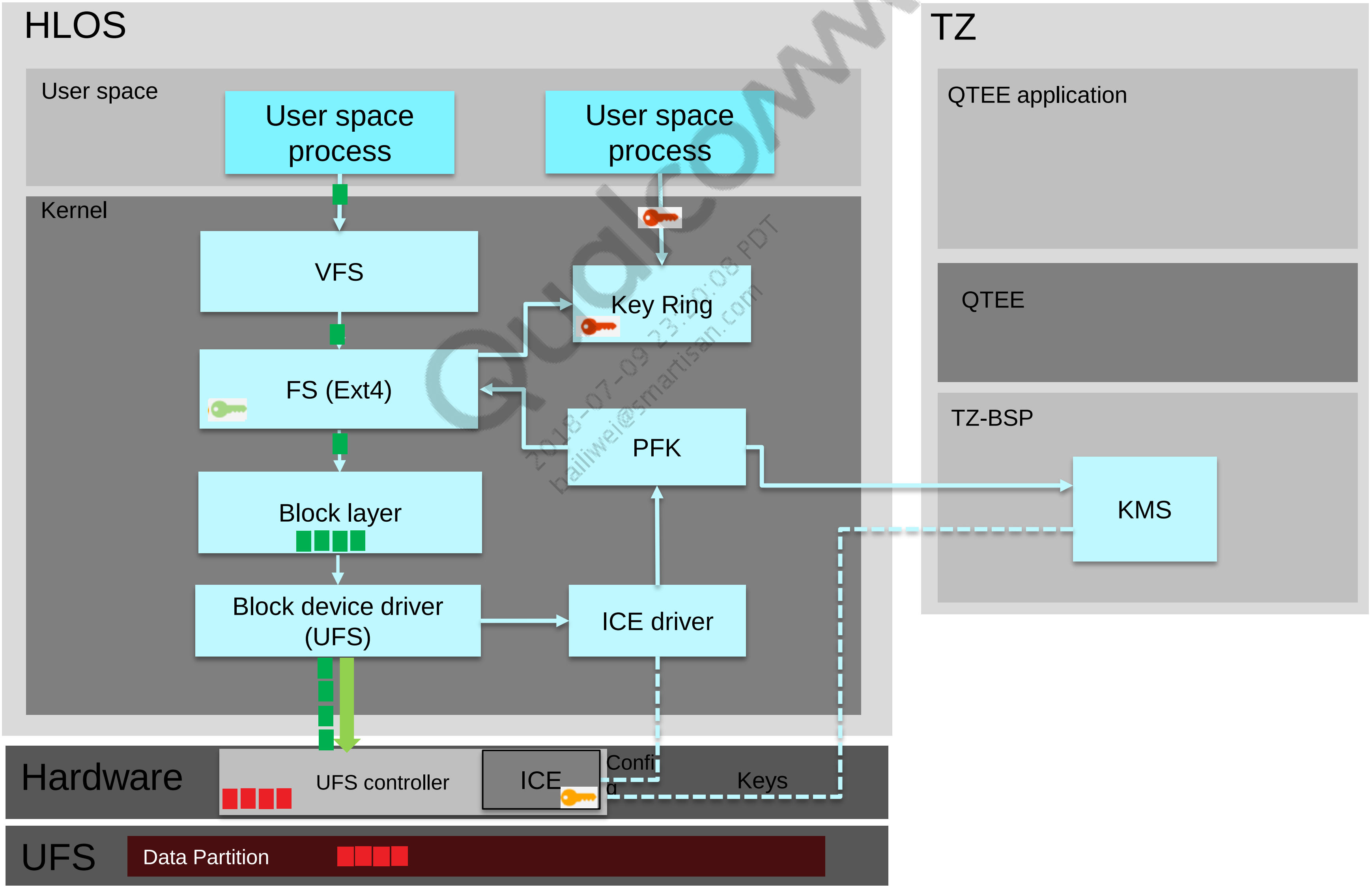
Hardware-Based FDE Solution



ICE − File Base Encryption

- File base encryption (FBE/ext4Crypt)
  - Each file encrypted with different key
  - Enables separation in higher granularity − users/groups/files
- Unique encryption key for each file
- FBE in ext4 level – ext4Crypt
- Files name encryption
- FEK are stored in xattr
- Master keys
  - Coupled with user credentials
  - Encrypt FEK
  - Stored in the key-ring

Hardware-Based FBE solution

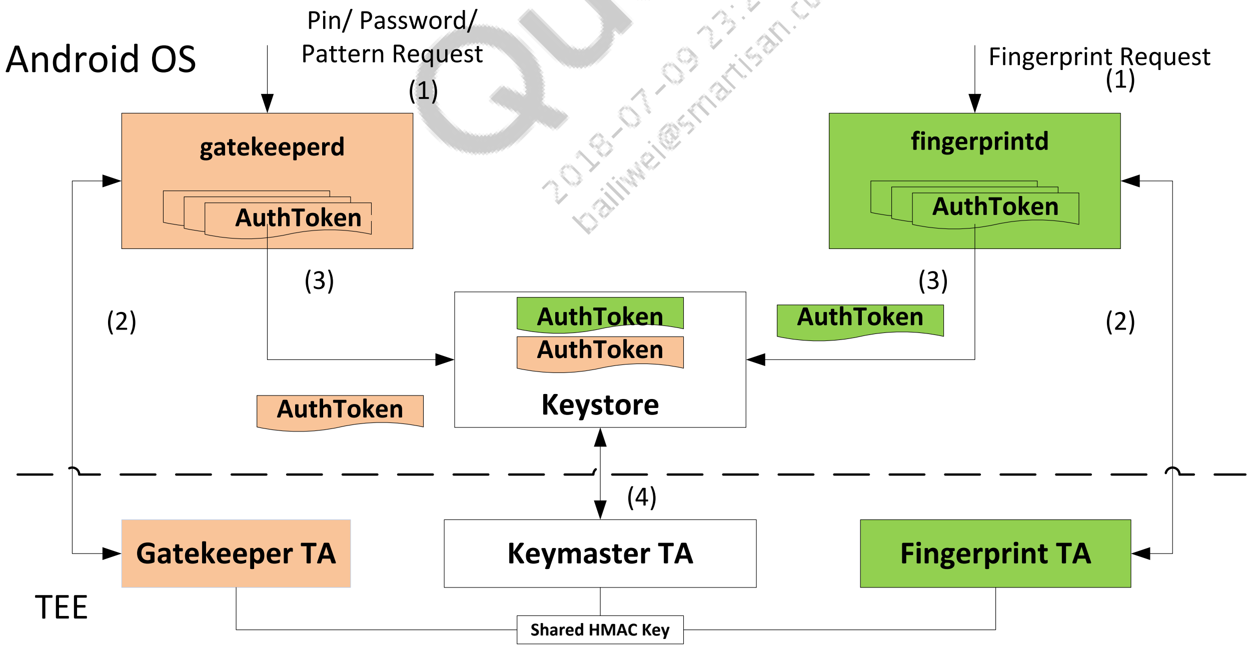


Hardware-Based Keymaster and Keystore

- Uses TZ application APIs to ensure that the key data stored is not accessible by HLOS
- Currently only an RSA key type is supported
- The keyblob generated are encrypted by a key accessible by TZ only and is stored in the file system (FS) on the HLOS end
- HLOS components include implementation of the keymaster APIs defined in keymaster.h:
  - generate_keypair
  - import_keypair
  - get_keypair_public
  - delete_keypair
  - delete_all
  - sign_data
  - verify_data
- Each of these APIs calls into TZ to process the request
- TZ components include a keymaster T application, which implements the APIs and is generated as split binary images in the TZ image build
- OEMs can determine whether to use RPMB anti-rollback support for keystore after the tzbsp_keystore_enable_rpmb() API is updated

Keymaster, Gatekeeper, and Fingerprint Process

- An authenticated token is created by gatekeeper when the user is authenticated
- A shared (keymaster, gatekeeper, or fingerprint trusted application) hashed message authentication code (HMAC) key is used to sign AuthToken

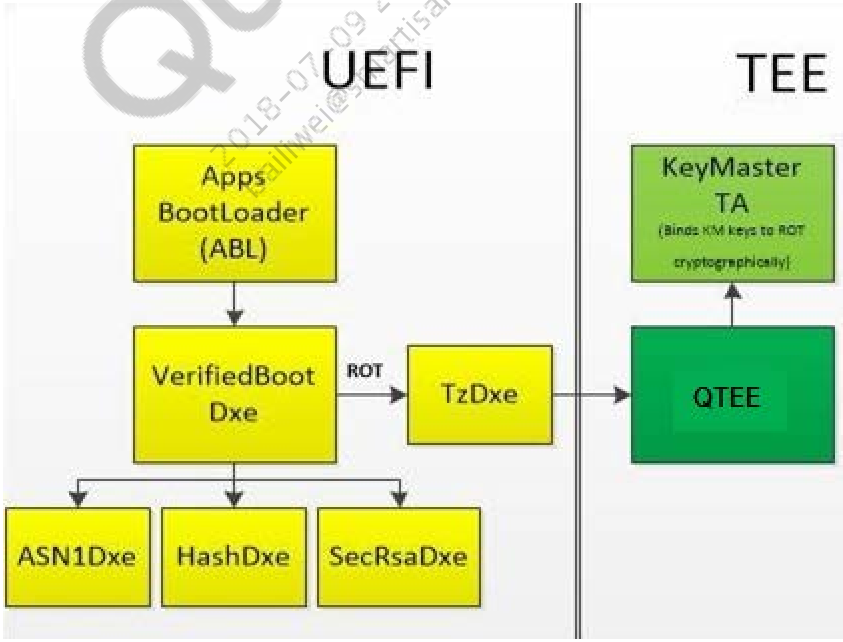


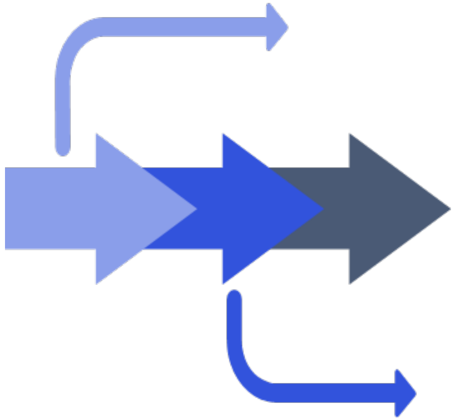
Keymaster 3.0

- Conforms to the new HIDL HAL
- Adds new attestation and device ID attestation features
- Device ID attestation: TEE can attest to the serial number, IMEI, MEID, brand, device, product, manufacturer, or the model with the following privacy restrictions:
  - Caller must provide correct values to get attestation
  - Feature can be permanently disabled by user
  - Caller cannot distinguish between user disable or incorrect values
  - Feature is accessible only to privileged applications
- CDD requirements
  - KM 3.0 with attestation are mandatory in Android O for new devices
  - Device ID extension is optional
- Upgrades
  - Can continue to use KM 1.5 or 2.0
  - Attestation keys provisioned on device using KM 1.5 or 2.0 needs to be reprovisioned if KM 3.0 is integrated
  - OEM should plan to erase user data

Verified Boot

- UEFI-based implementation
- Consists of both ABL and XBL codes
- In ABL the BootLib invokes the APIs implemented in XBL (VerifiedBootDxe driver) to perform image verification
- The EFIVerifiedBoot.h header file provides the API prototypes that are available for ABL.
- Verified boot status stored in RPMB

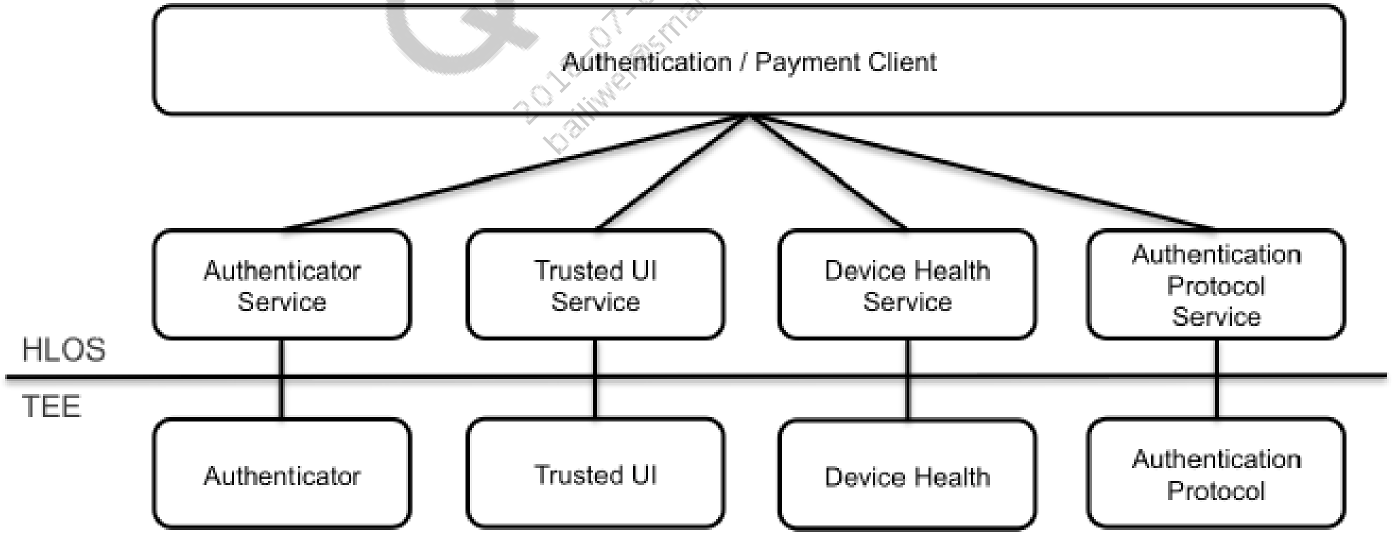




Authentication Frameworks

Authentication Frameworks Overview

- Trusted UI
- Secure Payment
- Camera Security
- Qualcomm Malware Detection


Authentication / Payment Client

Authenticator Service

Trusted UI Service

Device Health Service

Authentication Protocol Service

HLOS

TEE

Authenticator

Trusted UI

Device Health

Authentication Protocol

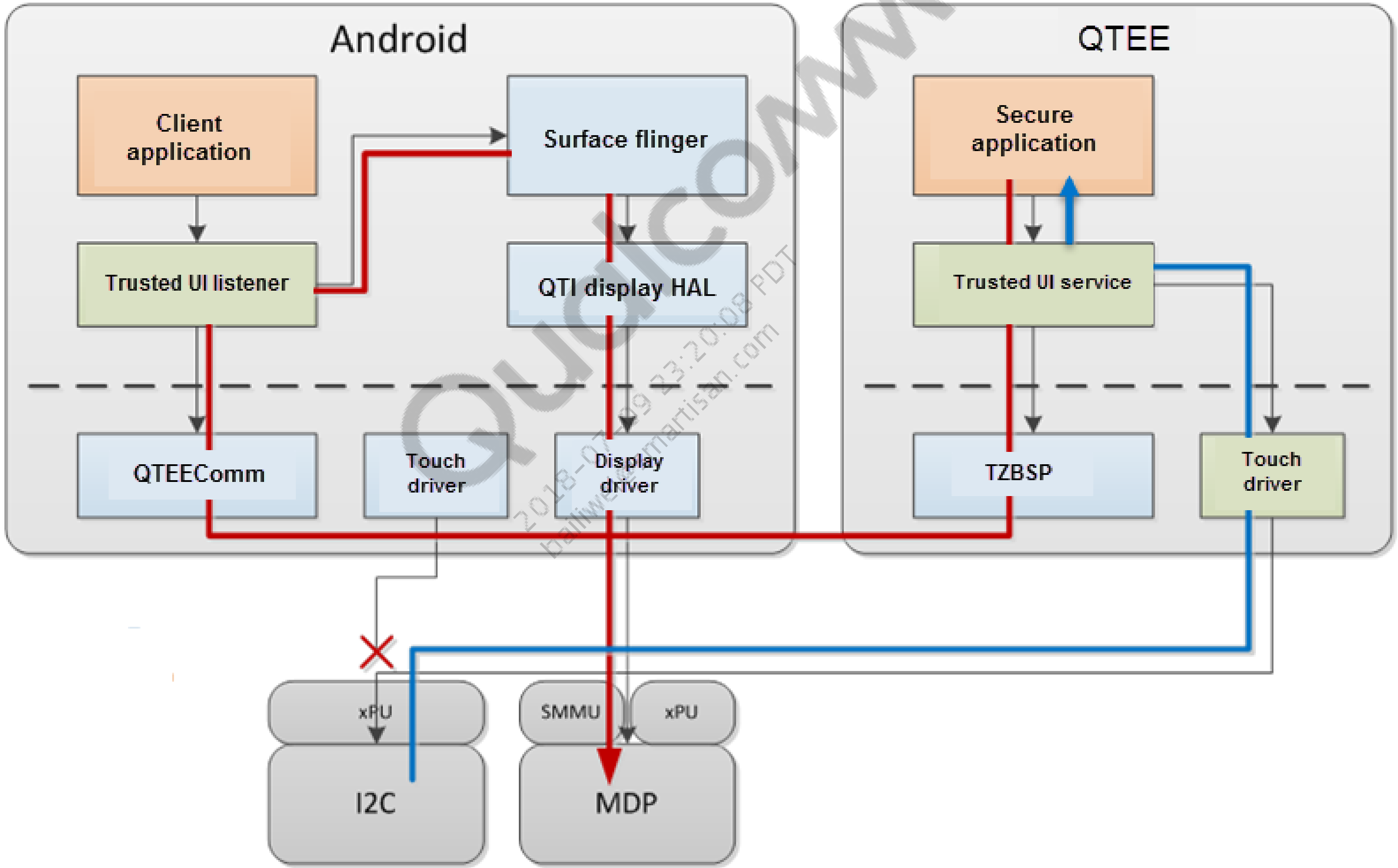
Authentication Framework Software Architecture

- The authentication framework comprises several services exposed to the Android operating system at the Java layer
- Authenticated trusted UI, and supplemental data services
  - IAuthenticator service – A generic authenticator interface that supports user verification with HLOS user verification token (UVT) routing
  - IAuthenticator2 service – A generic authenticator interface that supports user verification with TEE UVT routing
  - Trusted UI service – Trusted user interface service for transaction confirmation
  - Device health service – Supplemental data service to report device health metrics
- Authentication protocol services
  - FIDOCrypto service – Fast identity online (FIDO) universal authentication framework (UAF) 1.0 cryptographic protocol support
  - FIDOCryptov2 service – FIDO UAF 1.0 cryptographic protocol support, supplemental data, and trusted location support, transaction confirmation support that uses trusted UI

Trusted UI

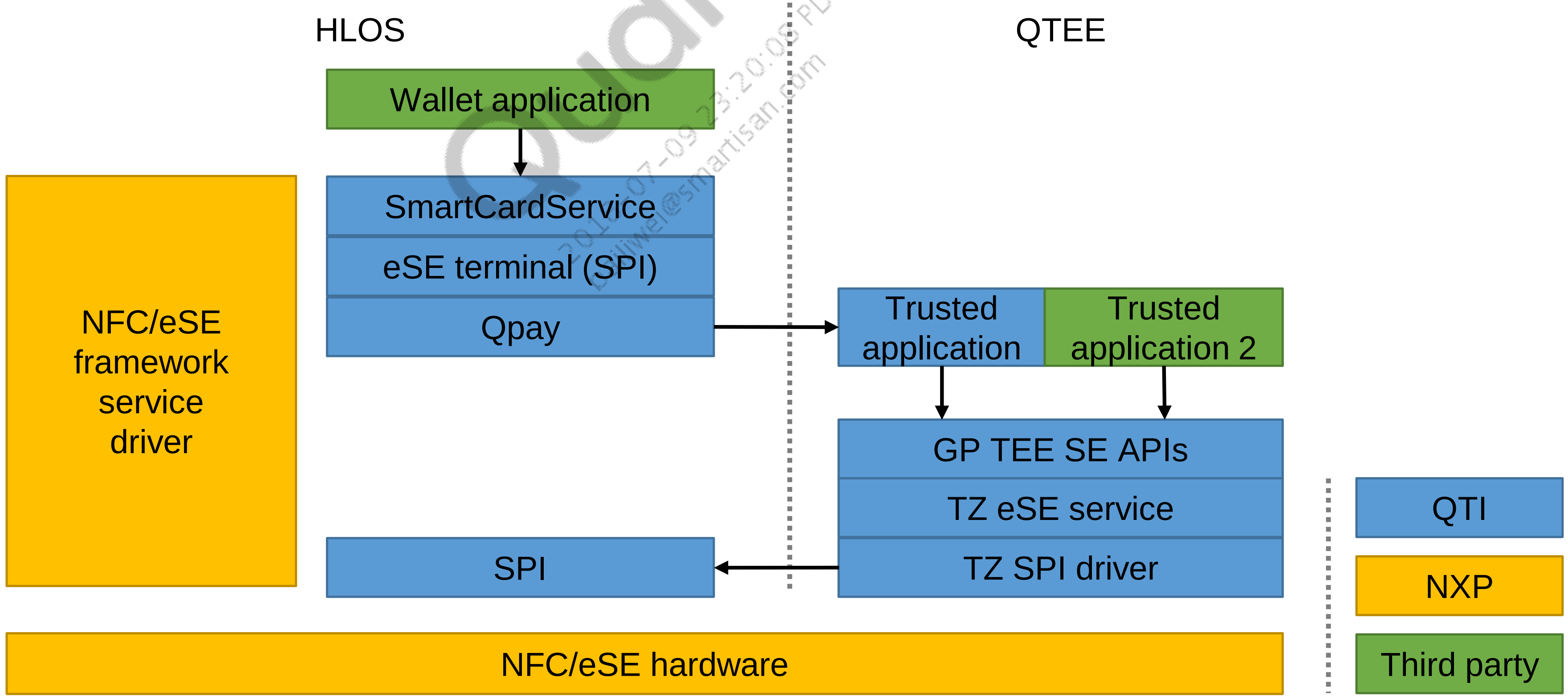
- The Trusted UI system feature allows QTEE secure applications to present a secure user interface to the end user
- Display contents and touch input are protected from operating system interference
- Typical use cases include banking and secure password entry, storage, and verification
- Cannot be used concurrently with secure camera

Trusted UI Block Diagram



Secure Payment

- Secure payment (QPay) is a transaction service that forges powerful payment processing solutions for virtually any biller or service provider using NFC enablement
- Integrates with NQxxx solutions for NFC/eSE
- Components of NFC enabled phone:



HLOS

QTEE

Wallet application

SmartCardService

eSE terminal (SPI)

Qpay

NFC/eSE framework service driver

SPI

Trusted application

Trusted application 2

GP TEE SE APIs

TZ eSE service

TZ SPI driver

NFC/eSE hardware

QTI

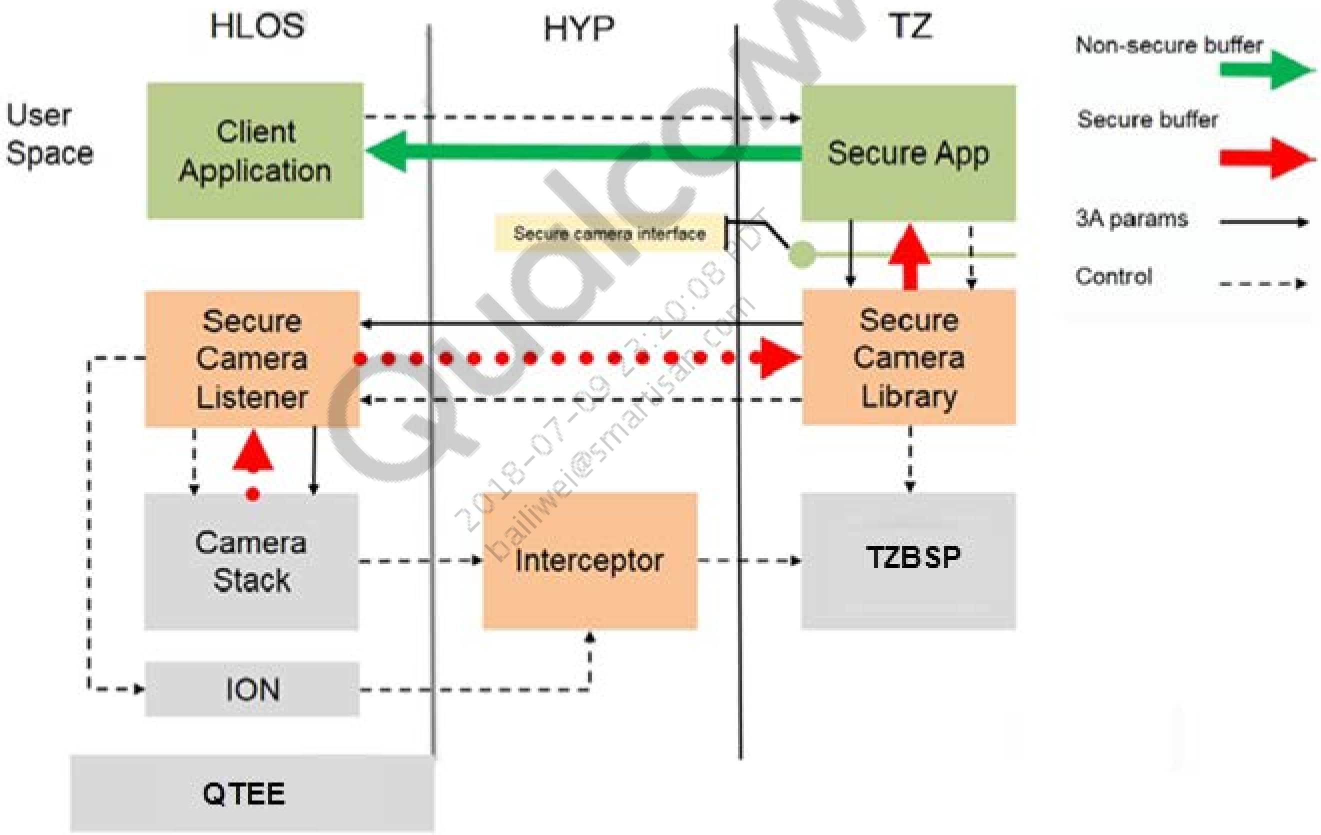
NXP

Third party

Camera Security

- Hardware-based authentication feature for camera-based biometrics that protects image data against malware attacks
- Eye-based authentication can use secure camera to provide strong security, designed to protect the link between the camera and the processor
- Biometric authentication is only as secure as the physical inputs and sensors used to gather it
- Isolates data received from the camera and securely stores it in the hardware, to safeguard against untrusted applications and processes
- Designed to allow only the correct person to access the device secure information, helps to deliver a protected authentication method to make payments and to conduct safe transactions

Camera Security Block Diagram



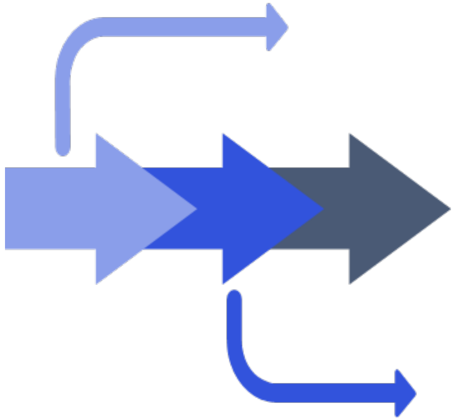
Qualcomm Malware Detection

- Intelligent malware detection that identifies zero-day, transitional malware, and rootkits using advanced machine learning and physical memory scanning
- Malware Detection performs real-time on-device monitoring using minimal CPU load to deliver superior field-proven results against known and unknown malware variants
- The device provides malware protection to monitor and issue alerts on the health of the device based on application behavior
- Analysis is performed by a secure application that runs outside the software OS
- Device health dhsecapp trusted application

Content Protection

Content Protection Overview

- Content protection is a solution to protect premium video content
- Leverages the strong security foundation provided in the hardware by Qualcomm® Processor Security (formerly known as SecureMSM) technology
- Approved by major content studios
- Multiple DRM supports most widely used content protection solutions in the industry, including PlayReady, Widevine, HDCP 2.x, and ISDB-T full segment
- End to end protection
  - All video streams (compressed and uncompressed) are protected
  - DRM stack runs in a secure execution environment
  - DRM keys are protected by hardware secure storage
- Pre-integrated
  - Turnkey solution is pre-integrated on Qualcomm® Embedded Platform chipsets
- Autonomous security
  - Does not rely on HLOS security
  - Not affected by HLOS root or attacks
  - Not affected by user-installed malicious software



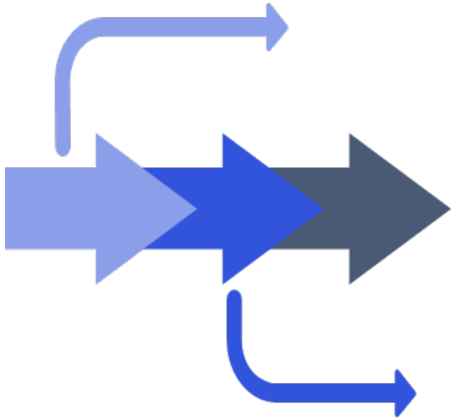
References

References

Title	Number
Qualcomm Technologies, Inc.	
Security Software Master Document	80-PA692-1
TZ.XF.5.0 TrustZone Architecture Overview	80-P9301-28
Qualcomm Android Security Features	80-NU861-1
Device Health	80-P3992-1
Application Note: Qualcomm Secure Execution Environment Version 2.4 SFS	80-NM249-1
Authentication Framework Overview	80-NU395-1
QPAY Integration Overview	80-P7202-15
Secure FIDO 2 Authenticator Overview	80-P4935-1
Secure Camera Library API	80-P2888-1

References (cont.)

Acronym or term	Definition
EL	ARM exception level
FIPS	Federal Information Processing Standards
HAL	Hardware abstraction layer
HLOS	High-level operating system
ICE	Inline crypto engine
KM	Keymaster
PBL	Primary boot loader
QTEE	Qualcomm Trusted Execution Environment (formerly known as Qualcomm Secure Execution Environment (QSEE))
RoT	Root of trust
RPMB	Replay protected memory block
SFS	Secure file system
TA	Trusted applications
TZ	ARM TrustZone
TZBSP	TrustZone board support package
XBL	Boot loader



Questions?

https://createpoint.qti.qualcomm.com