MediaTek CryptoCore HW v1.0, FW v1.0

FIPS 140-2 Non-Proprietary Security Policy

Version 2.2

MediaTek Inc.

June 2018

MediaTek Inc.
No. 1, Dusing 1st Rd.
Hsinchu Science Park
Hsinchu City 30078
Taiwan

Table of Contents

List of Figures

List of Tables

1. Introduction

This document is the non-proprietary security policy for MediaTek (MTK) CryptoCore cryptographic module. It contains a specification of the rules under which the module must operate and describes how this module meets the requirements as specified in Federal Information Processing Standard Publication 140-2 (FIPS PUB 140-2) for a Security Level 1 module.

1.1 Purpose of the Security Policy

There are three major reasons that a security policy is needed

- It is required for FIPS 140-2 validation,
- It allows individuals and organizations to determine whether the implemented cryptographic module satisfies the stated security policy, and
- It allows individuals and organizations to determine whether the described capabilities, the level of protection and the access rights provided by the cryptographic module meet their security requirements.

1.2 Target Audience

This document is part of the package of documents that are submitted for FIPS 140-2 conformance validation of the module. It is intended for the following people:

- Developers working on the release
- FIPS 140-2 testing lab
- Cryptographic Module Validation Program (CMVP)
- Consumers

1.3 Document Organization / Copyright

This non-proprietary security policy document may be reproduced and distributed only in its original entirety without any revision, © 2016 MediaTek Inc.

2. Cryptographic Module Specification

The following section describes the cryptographic module and how it conforms to the FIPS 140-2 specification in each of the required area.

2.1 Module Overview

The MTK CryptoCore cryptographic module (hereafter referred to as "the module") is designed to provide foundational security services for the platform, including secure boot, secure life cycle state, platform identity and key management. It offers high-throughput cryptography operations, suitable for a diverse set of use cases, such as secure playback of DRM-protected media content, IPSec VPNs, TLS/SSL link protection, drive encryption and more.

2.1.1 Module Embodiment

MTK CryptoCore is integrated in MediaTek Helio X30 mobile SoC, where the host processor runs two separate operational environments: a Secure OS, also called a Trusted Execution Environment (TEE), and a Public OS as a Rich Execution Environment (REE). The hardware isolation technology enforces data and control separation between the two environments containing the hardware and the firmware components. The firmware in the TEE is generally compact and strictly controlled. The firmware in the REE consists of a fixed kernel and an application layer, and the latter usually modifiable by the user. Therefore hardware and firmware belonging to the TEE are collectively called “Secure World” while those belonging to the REE are called “Public World”. The REE OS for this certification is Linux kernel and Android operating system. The module’s hardware and firmware reflects the system architecture, with the components largely divided into two “cores” — the Secure Core and Public Core. The cores communicate via a Persistent State Interface registers to synchronize their state and pass parameters. The module’s high level diagram, divided between the TEE and REE “worlds”, is shown in Figure **1**

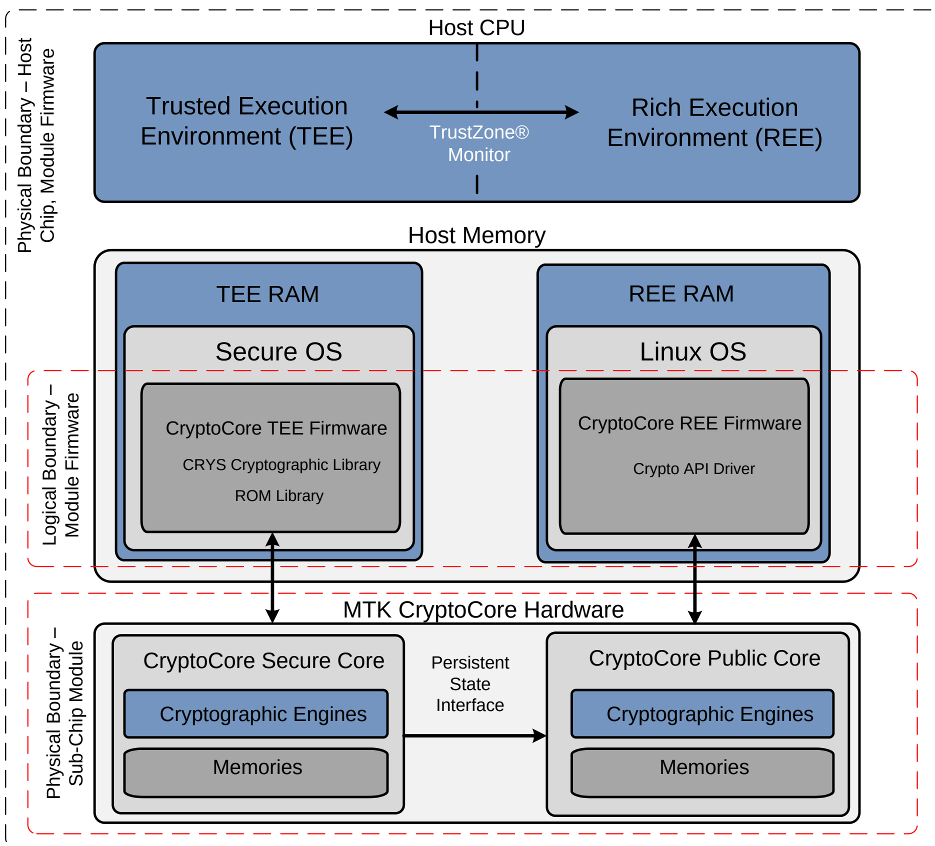


Figure 1 MTK CryptoCore High Level Diagram

Under the FIPS 140-2 definitions, MTK CryptoCore cryptographic module is classified as a firmware-hybrid and sub-chip module. The hardware that implements the module is integrated in Helio X30 SoC, making it a sub-chip cryptographic subsystem. The firmware part of the module, consisting of TEE and REE components, is stored in flash memory and runs on the host CPU, making it a firmware-hybrid module.

2.1.2 Module Validation Level

The module is validated at FIPS 140-2 security level 1. The following table depicts the security level claimed for each of the eleven sections that comprise the FIPS 140-2:

FIPS 140-2 Sections		Security Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1
3	Roles, Services and Authentication	1
4	Finite State Model	1
5	Physical Security	1
6	Operational Environment	N/A
7	Cryptographic Key Management	1
8	EMI/EMC	1
9	Self-Tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A

2.1.3 Module Hardware

The hardware portion of the cryptographic module is the MTK CryptoCore, which resides in MediaTek Helio X30 SoC. Detail descriptions are addressed in section 2.3.4.1 (Secure Core Hardware), section 2.3.4.3 (Public Core Hardware) and section 2.3.4.6 (Persistent State Interface).

Hardware Version
v1.0

2.1.4 Module Software

MTK CryptoCore has an API layer that provides consistent interfaces to the supported algorithms. Detail descriptions are described in section 2.3.4.2 (Secure Core Firmware) and section 2.3.4.4 (Public Core Firmware).

Firmware Version
v1.0

2.1.5 Tested Platform

The module has been tested on the following platform

Module name	Hardware Version / Test platform	Linux Version	Rich OS version	Secure OS Version
MTK CryptoCore	v1.0 / MediaTek Helio X30 SoC	Kernel 4.4	Android Nougat	Trustonic t-base 311b

2.2 Approved Security Functions and Mode of Operation

2.2.1 Approved Security Functions

MTK CryptoCore supports FIPS-approved security functions, as specified in Table 1. The following notes and caveats apply:

- KTS: The 128 bit key size is used for key transport while AES CCM was tested with 128, 192, and 256 bit keys.
- SHA-1: not approved for signature generation; approved only for legacy signature verifications and other uses.
- SHA and HMAC: SHA-384, SHA-512, HMAC-SHA-384 and HMAC-SHA-512 are only supported by Secure Core and are not supported by the Public Core.
- RSA: the key generation interface supports public exponents of size up to and including 2^{16}+1. Of these, only 2^{16}+1 is compliant with FIPS 186-4. The operator is instructed not to use smaller public exponents in Approved mode.
- Key Agreement – Diffie-Hellman and ECDH functions are only approved when used as part of a NIST [SP800-56A] key agreement protocol, otherwise only allowed.
- Key Agreement – Diffie-Hellman and ECDH: According to [SP800-56A], externally received public keys and domain parameters must be validated. The operator is instructed to invoke the module's domain parameter and public key validation functions provided for this purpose.
- A given Triple-DES key is allowed to be used for no more than 2^{28} times by policy.
- RSA and ECDSA: Approved hash functions must be used for signature generation, and SHA-1 is only Approved for legacy signature verification.
- CKG: generate asymmetric key using unmodified output from an approved DRBG as the random seed for FIPS 186-4 key generation

CAVP Cert	Algorithm	Standard	Mode / Method	Key Lengths, Curves or Moduli	Use	Public Core	Secure Core
4788	AES	[FIPS197,SP800-38A]	ECB, CBC, OFB, CTR	128, 192, 256	Encryption / Decryption	✓	
4788	AES	[SP800-38E]	XTS	256, 512	Encryption / Decryption	✓	
4788	AES	[SP800-38C]	CCM	128, 192, 256	Encryption / Decryption	✓	
Vendor-affirmed	AES	[SP800-38A-add]	CBC-CS1	128, 192, 256	Encryption / Decryption	✓	
4788	AES	[SP800-38B]	CMAC	128, 192, 256	Authenticated encryption / Decryption	✓	
4788	KTS	[SP800-38C,SP800-38F]	AES CCM	128	Key Transport	✓	

2541	Triple-DES	[FIPS46-3,SP800-67]	ECB, CBC	192	Encryption / Decryption	✓	
3926	SHS	[FIPS180-4]	SHA-1, SHA-224, SHA-256		Message authentication	✓	
3190	HMAC	[FIPS198-1,RFC2104]	HMAC SHA-1, SHA-224, SHA-256		Message authentication	✓	
4789	AES	[FIPS197,SP800-38A]	ECB, CBC, OFB, CTR	128, 192, 256	Encryption / Decryption		✓
4789	AES	[SP800-38E]	XTS	256, 512	Encryption / Decryption		✓
Vendor-affirmed	AES	[SP800-38A-add]	CBC-CS1	128, 192, 256	Encryption / Decryption		✓
4789	AES	[SP800-38C]	CCM	128, 192, 256	Encryption / Decryption		✓
4789	AES	[SP800-38B]	CMAC	128, 192, 256	Encryption / Decryption		✓
4789	KTS	[SP800-38C,SP800-38F]	AES CCM	128	Key Transport		✓
2542	Triple-DES	[FIPS46-3,SP800-67]	ECB, CBC	192	Encryption / Decryption		✓
3927	SHS	[FIPS180-4]	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512		Message authentication		✓
3191	HMAC	[FIPS198-1,RFC2104]	HMAC SHA-1, SHA-224, SHA-256, SHA-384, SHA-512		Message authentication		✓
2621	RSA	[FIPS186-4]	SHA functions PSS, PKCS-v1.5	2048, 3072	Signature generation / Verification		✓
CVL 1429	ECC CDH Primitive	[SP800-56A]	ECC	P-224, P-256, P-384, P-521	Key agreement		✓
1203	ECDSA	[FIPS186-4]	SHA functions	P-224, P-256, P-384, P-521	Signature Generation / Verification		✓
155	KBKDF (Key Derivation Functions)	[SP800-108]	Counter mode	CMAC AES-128, 256	Key derivation		✓
CVL 1428	KDF (Key Derivation Function)	[SP800-135,ANSI-X9.63,ANSI-X9.42]	SHA functions	SHA-224, SHA-256, SHA-384, and SHA-512	Key derivation		✓
1661	DRBG (Deterministic	[SP800-90A]	CTR-DRBG	256	Random Number Generation		✓

	Random Bit Generator)						
Vendor-affirmed	CKG (Crypto Key Generation)	[SP800-133]		128	Symmetric Key Generation		✓

Table 1 Approved Security Functions

2.2.2 Allowed Security Functions

MTK CryptoCore also supports FIPS-allowed security functions and modes, as specified in Table 2.

Algorithm	Caveat	Use
True Random Number Generator (NDRNG)	There is no NIST-approved standard for TRNG	DRBG seeding and reseeding
ECDSA [FIPS186-4] Curves: p224k1 and p256k1	p224k1 and p256k1 are non-NIST-Recommended Elliptic Curves. The associated security strength for the curves are 112 bits and 128 bits respectively	Signature generation/verification
RSAES-OAEP [PKCS1] Moduli: 2048, 3072	NIST allows for key wrapping only. Key establishment strength 112 or 128 bits	Encryption/decryption
RSAES-PKCS1-v1.5 [PKCS1] Moduli: 2048, 3072	NIST allows for key wrapping only. Key establishment strength 112 or 128 bits	Encryption/decryption
Diffie-Hellman [SP800-56A]	Key establishment strength 112 bits	Key agreement
EC Diffie-Hellman [SP800-56A] (CVL Cert# 1429)	Key establishment strength between 112 and 256 bits	Key agreement

Table 2 Allowed Security Function

2.2.3 Non-Approved Security Functions

Algorithm	Use	Public Core	Secure Core
AES non-approved modes: CBC-MAC [ISO/IEC-9797-1:2011]	Message authentication		✓
AES non-approved modes: XCBC-MAC [RFC3566]	Message authentication	✓	✓
AES GCM, GMAC [SP800-38D]	Encryption / Decryption	✓	
DES ECB, CBC [FIPS46-3]	Encryption / Decryption	✓	✓
Triple-DES, ECB, CDC with 2-key bundles [FIPS46-3,SP800-67]	Encryption / Decryption	✓	✓
MULTI-2 ECB, OFB [ARIB-STD-B25,ISO/IEC-9979]	Encryption / Decryption	✓	
IVGEN RNG	IV generation	✓	
MD5 [RFC1321]	Message authentication	✓	✓
HMAC-MD5 [RFC2104,FIPS198-1]	Message authentication		✓
ECIES [IEEE1363] Curves: p160k1, p160r1, p160r2, p192k1, p224k1, p256k1, P-192, P-256, P-384, P-512	Key agreement		✓
KDF1 and KDF2 Key Derivation Functions [ISO/IEC-18033-2]	Key derivation		✓

NIST SP800-135 KBKDF when using SHA1 [SP800-135,ANSI-X9.63,ANSI-X9.42]	Key derivation		✓
ECC key generation with Non-Approved size [FIPS186-4] Curves: p160k1, p160r1, p160r2, p192k1, P-192	Key generation		✓
ECDSA with Non-Approved sizes [FIPS186-4] Curves: p160k1, p160r1, p160r2, p192k1, P-192	Signature generation/verification		✓
RSA cryptographic primitives [PKCS1] Moduli: 1024, 2048, 3072, 4096	Encryption / Decryption		✓
RSA with Non-Approved sizes [FIPS186-4] 1024 bits	Key generation Signature generation/verification		✓
EC Diffie-Hellman with Non-Approved sizes [FIPS186-4] Curves: p160k1, p160r1, p160r2, p192k1, P-192	Key agreement		✓

Table 3 Non-Approved Security Functions

MTK CryptoCore also supports non-FIPS-approved security function and modes, as specified in Table 3. The following notes and caveats apply:

- MULTI-2: the MULTI-2 cryptosystem, used by the Japanese ARIB standard for conditional access in ISDB digital television, is specified in the ARIB-STD-B25 standard [ARIB-STD-B25], and in slightly more detail in the (withdrawn) ISO IEC-9979 standard [ISO/IEC-9979].
- IVGEN: See Section 2.3.4.5.
- Triple-DES: Supports 2-key bundles, which is no longer approved as of 2016. Enforces distinct keys. Rejects DES weak keys.
- The Elliptic Curve Integrated Encryption Scheme (ECIES) relies on CAVP validated components, as it consists of a key establishment step conforming to NIST [SP800-56A] followed by a key derivation step conforming to [PKCS1] or [ANSI-X9.63]. Nevertheless the scheme itself is not FIPS-approved.
- Key derivation functions: when using non-approved internal hash functions, the key derivation algorithms defined in NIST [SP800-135] are not approved.
- Key derivation functions: The key derivation function used to derive the Endorsement Key key pair is no approved.

2.2.4 Approved Security Mode

The module firmware can be compiled to Approved mode or Non-Approved mode image at compile-time. In the Approved mode, all approved security functions listed in Section 2.2.1 are available, and the non-approved security functions listed in Section 2.2.3 are disallowed by policy. If an operator calls the Non-Approved service in Approved mode, the module is temporarily put into Non-Approved mode.

2.3 Cryptographic Module Boundary and Components

2.3.1 Cryptographic Module Boundary

The physical boundary of the module is the physical boundary of MTK Helio X30 SoC that contains the sub-chip which implements the module hardware and the host processor which executes the module firmware. The sub-chip hardware also has a sub-chip boundary which is defined as the set of hard circuitry cores that comprises the sub-chip cryptographic subsystem.

The physical boundary of the firmware is the platform on which the firmware and operating systems reside. The logical boundary of the firmware is the set of binary files that make up the firmware.

2.3.2 Software Block Diagram

In Figure 2, the bidirectional arrows depict the flow of the status, control and data. The operations within the logical cryptographic boundary (the red dashed region) use the services from the hardware that is included in the logical boundary.

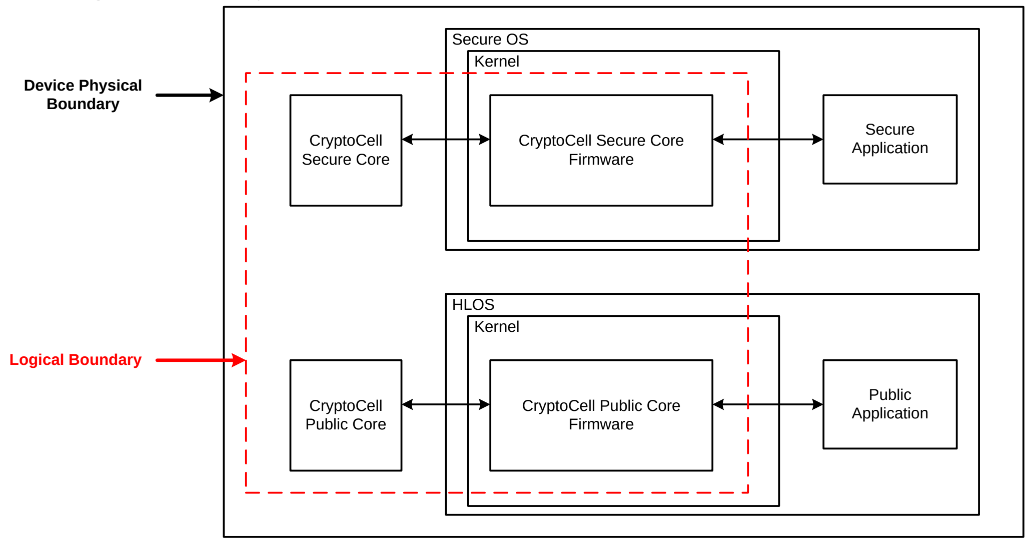


Figure 2 Cryptographic boundary

2.3.3 Hardware Block Diagram

The cryptographic boundary of the module and the relationship among the various internal components of the module are depicted in Figure 3.

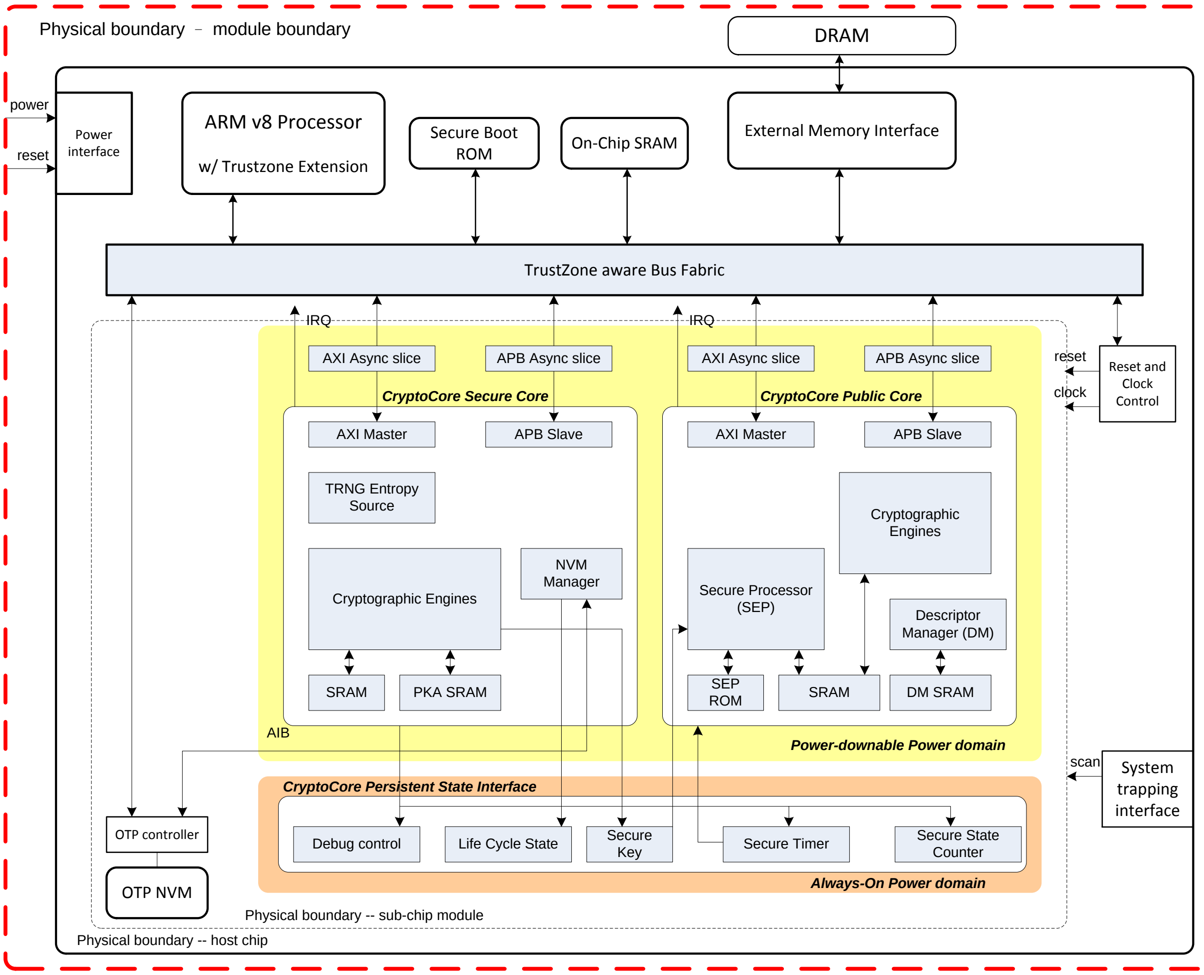
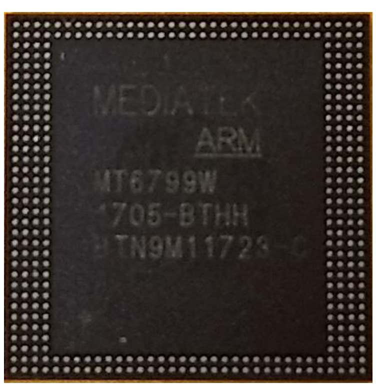
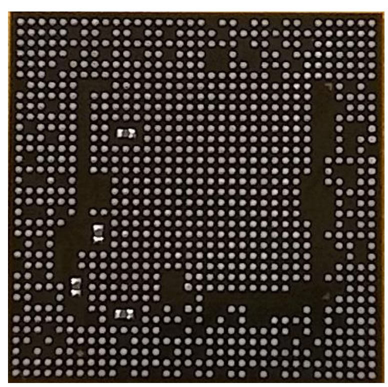


Figure 3 MTK CryptoCore System Diagram



Front View



Back View

Figure 4 MTK Helio X30 SoC

2.3.4 Module Component

As described in the introduction, MTK CryptoCore consists of hardware and firmware components, divided between the secure and the public "worlds". The sections below describe the hardware and firmware components of the Public and Secure cores.

2.3.4.1 Secure Core Hardware

Secure Core Hardware components are listed as follows:

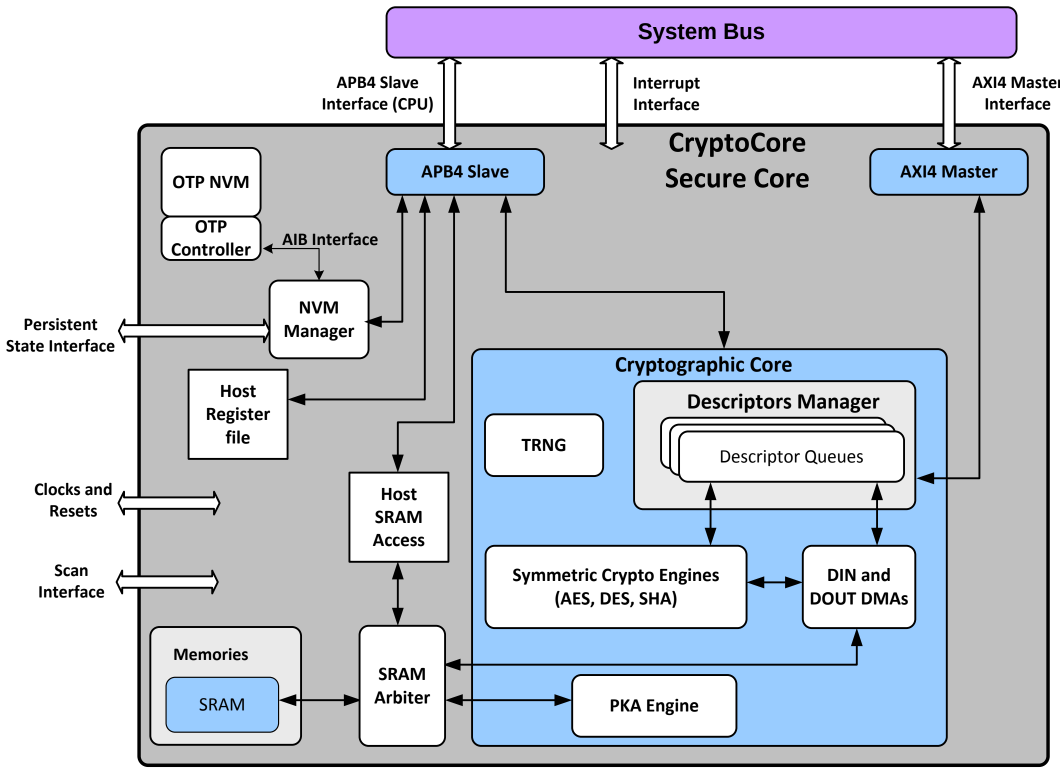


Figure 5 CryptoCore Secure Core Hardware

- Symmetric Cryptographic Engine
  - AES engine, supporting ECB, CBC, CBC-CTS, OFB, CTR, CBC-MAC, CMAC, XCBC-MAC, XTS and CCM modes.
  - DES engine, supporting DES (ECB and CBC modes) and TDES (EDE and DED) algorithms.
  - Hash engine, supporting SHA-1, SHA-256, SHA-384, SHA-512 and MD5 modes, as well as HMAC.
- Asymmetric cryptography Accelerators (PKA)
  - Support public-key cryptosystems based on the Discrete Logarithm problem, the Elliptic Curve Discrete Logarithm problem, and the Integer Factorization problem.
  - Support the following operations:
    - Modular arithmetic: addition, subtraction, and multiplication
    - Regular arithmetic: addition, subtraction, multiplication, and division
    - Modular inversion
    - Modular exponentiation
    - Bitwise operations: AND, OR, XOR and SHIFT

        - All operations work on integers between 128 bits and 4160 bits in size
- True random number generator (TRNG)
    - Collects random bits from the ring oscillators with guaranteed entropy
    - Supports post-processing like Von Neumann correction, followed by an auto-correlation test
- Non-volatile memory manager
    - Controls an internal, point-to-point Intel AIB (Intel Asynchronous Interface Bus Specification) interface to access a bank of one-time-programmable (OTP) memory
    - Controls all access to CSPs stored in OTP and protecting against unauthorized reading and modification
    - Sets the module's security lifecycle state according to the value of the various flag words in the OTP memory
- One-time programmable memory (OTP)
    - Stores data including platform keys, Life Cycle States, and TRNG configuration
    - Accessible solely through the non-volatile memory manager
- Dedicated SRAM
    - Storages intermediate data for PKA, the symmetric cryptography engines, and the TRNG entropy collector
- Secure timer
    - Provides secure time stamp for security sensitive application
- DMA controller
    - Requests data reads and writes to/from MTK CryptoCore

2.3.4.2 Secure Core Firmware

Figure 6 depicts the firmware components associated with the Secure Core. These are grouped in the Secure Boot ROM Library, the embedded Cryptography Software (CRYS) Library, various runtime utility components, and the Hardware and Platform abstraction layers.

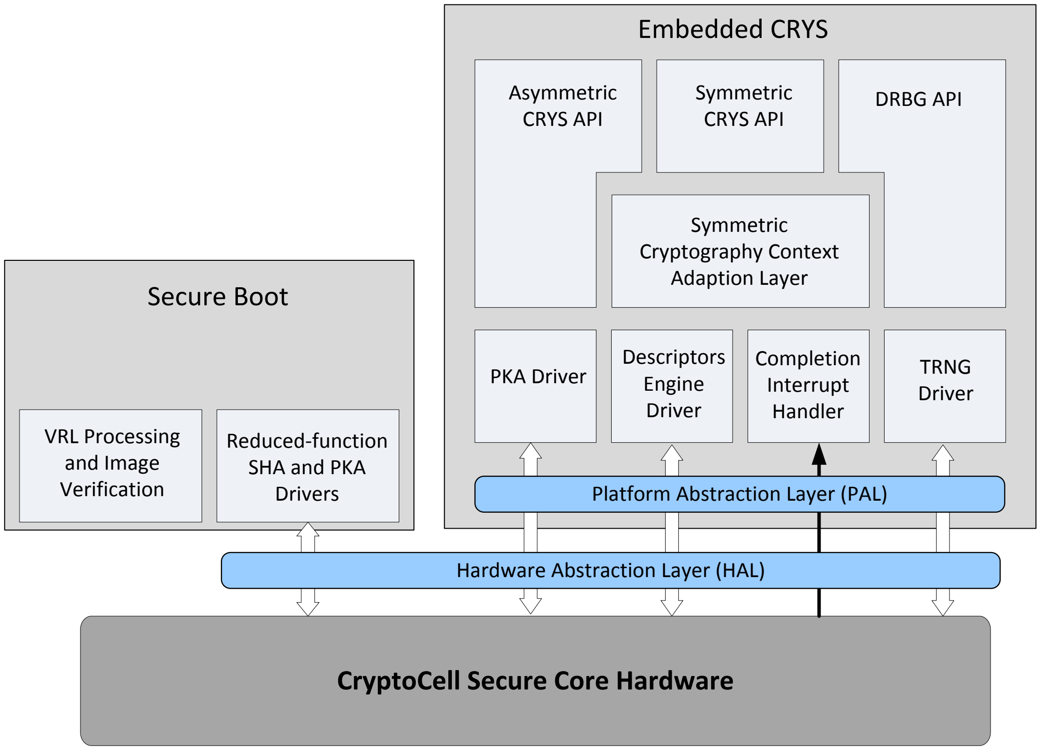


Figure 6 CryptoCore Secure Core Firmware Components

The functional components are described as follows:

- Secure Boot ROM Library
  - Integrated in the Boot ROM that acts as the root-of-trust
  - Provides security functions that are used for
    - Chaining of secondary public keys, signed by the root-of-trust public key, to permit parts of the system image to be signed by other trusted developers
    - Firmware updates — support for changes introduced by an external mechanism
    - Image version revocation
    - Secure Debug authentication
- Embedded Cryptography Software Library
  - Provides the symmetric, asymmetric cryptography and hash services that utilizing the engines in Secure Core
  - Provides Deterministic Random Bit Generation (DRBG) which follows NIST SP 800-90A CTR_DRBG [SP800-90A], and with seed from TRNG
  - Provides services for key management, secure timers, RPMB key and Life Cycle Management
  - Provides RAM Backup and restore service for external RAM that holds sensitive system state over periods of sleep mode or warm resets
  - The Platform Abstraction Layer (PAL) provides a firmware abstraction to the module's interface to the secure OS, and includes functions for initialization and termination, DMA

control, memory allocation, memory mapping and basic memory manipulation functions and implementations of synchronization primitives such as mutexes and barriers.

- Hardware Abstraction Layer (HAL)
    - Provides a firmware abstraction for the module's basic hardware capabilities, including interrupt and cache control, initialization and termination.

2.3.4.3 Public Core Hardware

Public Core Hardware components are listed as follows

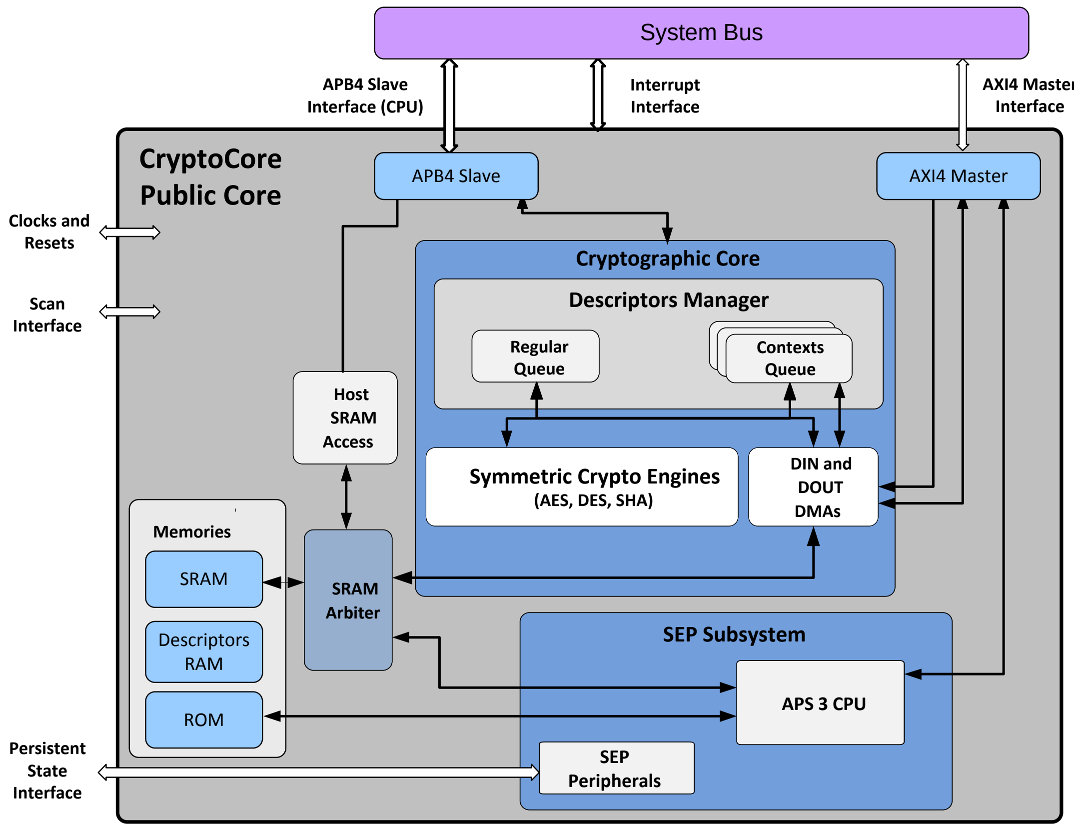


Figure 7 CryptoCore Public Core Hardware

- Symmetric Cryptographic Engine
    - AES engine, supporting ECB, CBC, CBC-CTS, OFB, CTR, CMAC, XCBC-MAC, XTS and CCM modes.
    - DES engine, supporting DES (ECB and CBC modes) and TDES (EDE and DED) algorithms.
    - Hash engine, supporting SHA-1, SHA-256, and MD5 modes, as well as HMAC.
    - AES-MAC engine, supporting AES CMAC and AES XCBC MAC modes.
    - MULTI-2 engine, supporting MULTI-2 ECB and OFB modes.
- Security Processor Subsystem (SEP)
    - An internal 32-bit APS3 CPU core with its own memory-mapped peripherals, timer and clocks, a watchdog timer, an interrupt controller, UART, and clock.
    - The SEP executes code from a dedicated ROM which passes the integrity check at power-on.

  - The SEP manages the Secure Key mechanism for decrypting the Secure Key packages that are exported from the Secure Core to the Public Core, and keeps the logic and the CSPs inaccessible to the Host CPU and the Public Core firmware.
- Dedicated SRAM
  - Used for the descriptor queue and other input/output operations.

2.3.4.4 Public Core Firmware

Figure 8 depicts the Public Core firmware components. The firmware is implemented as a standard Linux Kernel driver which registers with the operating system as a platform device driver. By registering the services the firmware serves as Linux kernel crypto algorithms.

The functional components are described as follows:

- Cryptography Firmware Library
  - Provides the symmetric cryptography and hash services that utilizing the engines in Public Core
  - Provides Secure Key mechanism which enable cryptographic operations in the Public Core using keys generated by the Secure Core, while keeping the logic and the CSPs inaccessible to the Host CPU and the Public Core firmware.

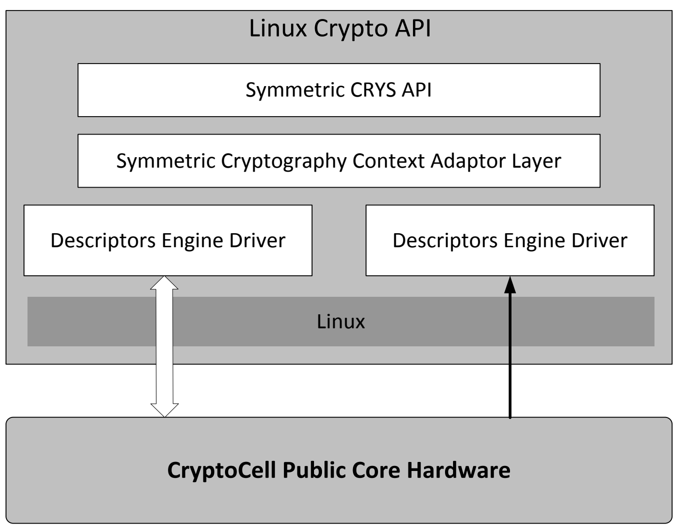


Figure 8 CryptoCore Public Core Firmware Components

2.3.4.5 IV Generator

Linux cryptographic interfaces support an invocation mode in which the Initialization Vectors (IVs) are generated internally, rather than passed in as a parameter by the caller. For this purpose, the Public Core firmware contains a non-approved DRBG called IVGEN. IVGEN is seeded from the Linux kernel API *get_random_bytes()*, invokes the module's AES CTR service to expand the seed into a 1KB buffer, and uses that buffer to return IVs on request. Once the buffer runs out, it is regenerated. This design allows for efficient IV generation in performance-intensive scenarios.

IVGEN is not offered as a service by the module; it is only used for IV generation in Public Core algorithms. This DRBG is non-approved because it does not implement the Continuous RNG Test. IVGEN is not to be used in the FIPS-Approved Mode.

2.3.4.6 Persistent State Interface

Persistent State Interface passes critical states, data, and shared secrets between the Secure Core and Public Core, and is kept in the always-on power domain. This states survive in "sleep mode" that is the power down of both the Secure Core and the Public Core. The Persistent State Interface is shown in Figure 3, and contains the following components:

- Session Key is a key written by Secure Core at boot time. This key is passed on direct, point-to-point physical lines, precluding external access.
- Life cycle state, a state value written by Secure Core based on dedicated OTP field. Firmware components should access the Life Cycle State through a dedicated firmware service.
- Debug Control Unit word, driven by the Secure Debug mechanism to control host debugging capabilities.
- 32-bit State Counter, this counter holds the number of times the Secure Core was power cycled since the last full-system boot. This is used by the RAM Backup and Restore services
- Secure timer, holds the current tick count. It is output from the Secure Core and input by the Public Core. Firmware components should access the Secure Timer through a dedicated firmware service.

2.3.4.7 Secure Key Mechanism

The Secure Key mechanism enables cryptographic operations with the Public Core's high throughput while the Secure Core takes full control of the keys. The mechanism uses AES-128 CCM authenticated encryption for key wrapping with the Session Key, and a dedicated hardware channel for data passing between the Secure Core and the Public Core. The Session Key is passed from Secure Core to Public Core. The Secure Core uses the Session Key to encrypt a data structure containing a user key and parameters controlling its usage. On the Public Core side, the SEP subsystem unwraps the data using the Session Key, checks the parameters, and loads the key directly into the Public Core encryption engines. The keys are left inaccessible to the Public Core Firmware. The encrypted Secure Key packages are returned to user, and are passed from Secure Core RAM to Public Core RAM by an out-of-band mechanism, such as a TEE application. From there, Secure Key packages can be passed as the key parameter to Public Core functions supporting Secure Key.

Secure Key packages are rendered inaccessible when Session Key is regenerated, either on power-on or on demand.

2.4 Life Cycle State and Operational State

MTK CryptoCore life cycle and operational states follow a Finite State Model. The following is a summary outlining the life cycle states (LCS) and the operational states (OPS) comprising the module.

Name	State	Description
Cold Power-on	OPS	System powers on

Operational	LCS	A module in the field is normally in the *Operational* life cycle state, which means it has all platform keys in place and security functions enabled
Self-Test	OPS	Performs FIPS power-on tests.
Error	OPS	The FIPS error state is entered from the *Self-Test* state or through one of the conditional tests in case of error
Approved Mode	OPS	FIPS-Approved mode of operation. FIPS-approved functions are available for use
Non-approved Mode	OPS	Non-FIPS-approved mode of operation. All functions are available for use
Chip Manufacturing	LCS	This is the initial life cycle state at manufacturing, associated with the Independent Chip Vendor
Device Manufacturing	LCS	This state is associated with the Original Equipment Manufacturer (OEM), who packages the chip into a finished device
Security Disabled	LCS	This is the option for some manufacturer may select some chips or devices to have their security functionality blocked
RMA	LCS	The *Return Merchandise Authorization* state is for devices that return to the manufacturer for failure analysis
Secure Debug	OPS	The debug is enabled

Table 4 Life Cycle and Operational States

3. Cryptographic Module Ports and Interfaces

The module supports a number of physical and logical ports and interfaces, as shown in Table 5 and described in details below. The AXI4 and APB4 buses and interrupt signals are used exclusively by the module's firmware, and carry data, control and status between module's firmware and hardware.

Type	Interfaces	Secure Core	Public Core
Power	Power	✓	✓
Control	Clock	✓	✓
	Reset	✓	✓
	Scan	✓	✓
	APB4	✓	✓
	Firmware API calls	✓	✓
Status Output	Interrupt	✓	✓
	Firmware API return values	✓	✓
Data Input	APB4	✓	✓
	Firmware API inputs	✓	✓
Data Output	AXI4	✓	✓
	Firmware API outputs	✓	✓

Table 5 Ports and Interfaces

3.1 Secure Core and Public Core Hardware Interfaces

The interfaces are duplicated and presented in both Secure Core and Public Core.

- AXI4 Master: The two AXI4 masters are used to pass data between the host system memory and each core's internal memory.
- APB4 Slave: The two APB4 slaves are used to control each core's operation. The host accesses the APB4 slaves as a memory-mapped IO device. The interface provides access to the module's control registers, and serves to pass descriptors and trigger hardware operations.
- Interrupt: Both cores have an interrupt signal to inform the host for operation completion or error condition.
- Clocks: Each core has multiple internal clock domains with a separate clock domain for each cryptographic engine.
- Power: Each core uses a single power domain which can be powered down individually for power saving purposes. The power states are preserved in the always-on Persistent Power State Interface.
- Reset: Each core uses a single reset signal which resets core logics and registers
- Scan: Each core supports a scan input signal. The signal is connected to the core reset line. It clears keys stored in hardware registers when asserted.

3.2 Secure Core Firmware Interface

The Secure Core firmware exposes its services through its APIs which are grouped in several categories corresponding to the organization of the firmware components. See section 4.2 for the full list of services. Documentation for the APIs is available in the proprietary documentation package.

3.3 Public Core Firmware Interface

The Public Core firmware is implemented in a standard Linux Kernel driver. The driver register itself as a Linux kernel cryptographic algorithm provider, and offers cryptographic services to applications and kernel components. The services are documented in the full proprietary documentation package.

4. Roles, Services and Authentication

4.1 Roles

MTK CryptoCore offers high-level cryptographic services which operate on CSPs and user inputs, as well as platform security services which can be used by the incorporating platform to establish a root of trust. In addition, the module requires initialization at manufacturing time, and offers special states for recovery and debug.

The following two roles are defined:

User Role

The user is defined as a set of firmware applications running in the Secure World and the Public World, when the module is in Operational state. This role accesses cryptographic services, including Approved and non-Approved security fucntions, as well as platform security services.

Crypto Officer Role

The Crypto Officer is defined as the platform's manufacturer. The Crypto Officer is responsible for initializing the module's non-volatile memory and OTP, in order to make the module operational. This role has exclusive access to the services that change the module's life cycle state. Life cycle states that are designated for the Crypto Officer Role are Chip Manufacturing, Device Manufacturng, RMA (Return Merchandise Authorization), and Security Disabled and Secure Debug.

The module does not define a Maintenance Role. While the module does enable the operator to enter the Secure Debug and RMA states after the module has already been operational, those states do not provide additional access to CSPs in the module. Secure Debug and RMA states are limited to holders of the certificates signed by the manuacturer.

4.2 Services

Table 6 below lists the services offered by MTK CryptoCore along with a concise description of purpose, the security function(s) offered by the service, key(s) in use, the role that can access that service, the Approved or Non-Approved status, the core(s) implementing it, and the operator's access rights to any keys and CPSs involved.

Note that when a service is marked as implemented by both Public Core and Secure Core , those represent two distinct implementations.

Access to keys is denoted with a single letter according to the following notation:

- I – input key(s) from the user
- O – output key(s) to the user
- R – read key(s) from internal storage
- W – write key(s) to internal storage
- Z – zeroize key(s)

Service Name	Purpose	Security Functions	Keys and CSPs	User Role	CO Role	Approved	Public Core	Secure Core
AES Secure Core Approved	Encryption Decryption	AES-128, 192, 256, modes: ECB, CBC, CTR, OFB, CMAC, XTS, CTS (CBC-CTS CS1)	**Input:** User keys (I) Platform keys (R)	✓		✓		✓
AES Secure Core Non-Approved modes	Encryption Decryption	AES-128, 192, 256, modes: CBC-MAC, XCBC-MAC	**Input:** User keys (I) Platform keys (R)	✓				✓

Service Name	Purpose	Security Functions	Keys and CSPs	User Role	CO Role	Approved	Public Core	Secure Core
AES Public Core Approved	Encryption Decryption	AES-128, 192, 256, modes: ECB, CBC, CTR, OFB, CMAC, XTS, CTS (CBC-CS1), GCM, GMAC	**Input:** User keys (I)	✓		✓	✓	
AES Public Core Non-Approved functions	Message authentication	AES-128, 192, 256, modes: XCBC-MAC	**Input:** User keys (I)	✓			✓	
DES Non-Approved	Encryption Decryption	DES ECB, CBC	**Input:** User keys (I)	✓				✓
Triple-DES Approved	Encryption Decryption	Triple-DES ECB, CBC with three-key bundles (DED and EDE)	**Input:** User keys (I)	✓		✓	✓	✓
Two-key Triple-DES Non-Approved	Encryption Decryption	Triple-DES ECB, CBC, with two-key bundles (DED and EDE)	**Input:** User keys (I)	✓				✓
MULTI2 Non-Approved	Encryption Decryption	ECB, OFB modes.	**Input:** User keys (I)	✓			✓	
SHA Approved functions – Secure Core	Message authentication	SHA-1[1], SHA-224, SHA-256, SHA-384, SHA-512		✓		✓		✓
SHA Approved functions – Public Core	Message authentication	SHA-1[1], SHA-224, SHA-256		✓		✓	✓	
MD5 Non-Approved	Legacy message authentication	MD5		✓			✓	✓
HMAC Approved functions – Secure Core	Message authentication	HMAC SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	**Input:** User keys (I)	✓		✓		✓
HMAC Approved functions – Public Core	Message authentication	HMAC SHA-1, SHA-224, SHA-256	**Input:** User keys (I)	✓		✓	✓	
HMAC Non-Approved functions	Legacy message authentication	HMAC-MD5	**Input:** User keys (I)	✓			✓	✓
NIST SP800-108 Key Derivation Function Approved	Key-Based Key derivation function (KBKDF) [SP800-108]	KDF in Counter Mode. AES-128 or 256 in CMAC mode	**Input:** User keys (I) Device Root Key (R) Device Root Key (R) OEM Key (I) **Output:** User Keys (O) Session Key (W)	✓		✓		✓

[1] Not approved for signature generation. Approved for legacy signature verification and other uses.

Service Name	Purpose	Security Functions	Keys and CSPs	User Role	CO Role	Approved	Public Core	Secure Core
			Platform Key (W) OEM Key (O)					
NIST SP800-135 Key Derivation Functions Approved	Key derivation as specified in NIST [SP800-135][2]	SHA-224, SHA-256, SHA-384, SHA-512	**Input:** User keys (I) **Output:** User Keys (O)	✓		✓		✓
NIST SP800-135 Key Derivation Functions Non-Approved	Key derivation as specified in NIST [SP800-135][2]	SHA-1	**Input:** User keys (I) **Output:** User Keys (O)	✓				✓
KDF1 and KDF2 Key Derivation Functions Non-Approved	Key derivation functions KDF1 and KDF2 defined in [ISO/IEC-18033-2]	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	**Input:** User keys (I) **Output:** User Keys (O)	✓				✓
Endorsement Key Derivation	Derive an Endorsement Key ECC key-pair	ECC key pair generation[3]	**Input:** Device Root Key (R) **Output:** Endorsement key pair (O)	✓				✓
Replay Protected Memory Block (RPMB) Key Derivation	Derive RPMB Key using KBKDF [SP800-108]	KBKDF	**Input:** Device Root Key (R) **Output:** RPMB Shared Key (O)	✓		✓		✓
RPMB Page MAC generation	RPMB Frame authentication [JESD84]	HMAC SHA-256	**Input:** RPMB Shared Key (I)	✓		✓		✓
Elliptic Curve Cryptography (ECC) Key Generation Approved	ECC key generation [FIPS186-4]	Curves: P-224, P-256, P-384, P-P521, p224k1, p256k1 Uses DRBG	**Output:** User keys (O) **Sizes:** 224, 256, 384, 521 bits	✓		✓		✓
Elliptic Curve Cryptography (ECC) Key Generation Non-Approved strengths	ECC key generation [FIPS186-4]	Curves: P-192, p160k1, p160r1, p160r2, p192k1, user defined domains Uses DRBG	**Output:** User keys (O) **Sizes:** 160, 192 bits	✓				✓
ECC Public Key Validation	Public key validation		**Input:** User Keys (I)	✓		✓		✓

[2] This includes key derivation methods ANS1DER and Concatenation specified in [ANSI-X9.42] section 7.7.1 and [ANSI-X9.63] section 5.6.3, approved when used as part of [SP800-56A] agreement scheme

[3] ECC private key is derived according to [FIPS186-4] Section B4.1. Instead of a random input from an RBG, the procedure uses the result of a [SP800-108] KBKDF applied to the Device Root Key. The resulting key pair has a different value per module. It is fixed until zeroization.

Service Name	Purpose	Security Functions	Keys and CSPs	User Role	CO Role	Approved	Public Core	Secure Core
Elliptic Curve Digital Signing Algorithm Approved	Signing Verification [FIPS186-4]	SHA functions	**Input:** User keys (I) **Sizes:** 224, 256, 384, 521 bits.	✓		✓		✓
Elliptic Curve Digital Signing Algorithm Non-Approved strengths	Signing Verification [FIPS186-4]	SHA functions Non-approved when using non-approved key sizes and domains	**Input:** User keys (I) **Sizes:** 160, 192 bits; user-defined domains	✓				✓
Elliptic Curve Diffie-Hellman Approved	Key Agreement [SP800-56A]	EC Diffie-Hellman	**Input:** User keys (I) **Output:** User Keys (O) **Sizes:** 224, 256, 384, 521 bits	✓		✓		✓
Elliptic Curve Diffie-Hellman Non-Approved strengths	Key Agreement [SP800-56A]	EC Diffie-Hellman when using non-approved key sizes and domains	**Input:** User keys (I) **Output:** User keys (O) **Sizes:** 160, 192 bits; user-defined domains	✓				✓
Elliptic Curve Integrated Encryption Scheme Non-Approved	Key Transport [ISO/IEC-18033-2]	ECIES-KEM KDF with SHA hashes. Uses DRBG	**Input:** User keys (I) **Output:** User keys (O) **Sizes:** 160, 192, 224, 256, 384, 521 bits; user-defined domains.	✓				✓
RSA Key Generation Approved	Key pair generation (regular and CRT) [FIPS186-4]	Uses DRBG	**Output:** User keys (O) **Sizes:** 2048, 3072 bits	✓		✓		✓
RSA Key Generation Non-Approved strengths	Key pair generation (regular and CRT)	Uses DRBG	**Output:** User keys (O) **Sizes:** 512, 1024 bits	✓				✓
RSA Signature Generation and Verification Approved	PSS signing and verification [FIPS186-4]	RSA SHA-1 (legacy verification only); SHA-224, SHA-256, SHA-384, SHA-512; DRBG	**Output:** User keys (I) **Sizes:** 2048, 3072; 1024 (legacy verification only)	✓		✓		✓
RSA Signature Generation and Verification Non-Approved sizes or functions	PSS and RSAES-OAEP signing and verification [FIPS186-4,PKCS1]	RSA SHA functions PKSC1.5 with MD5 DRBG	**Input:** User keys (I) **Sizes:** 512, 1024, 4096 bits	✓				✓
RSA Encryption and Decryption for key	RSAES-OAEP encryption and	RSA SHA1, SHA-224, SHA-256, SHA-384, SHA-512	**Input:** User keys (I) **Sizes:** 2048, 3072 bits	✓				✓

Service Name	Purpose	Security Functions	Keys and CSPs	User Role	CO Role	Approved	Public Core	Secure Core
wrapping Allowed	decryption [PKCS1]	DRBG						
RSA Encryption and Decryption for key wrapping Allowed	RSAES-PKCS1-v1.5 encryption and decryption [PKCS1]	RSA SHA1, SHA-224, SHA-256, SHA-384, SHA-512 DRBG	**Input:** User keys (I) **Sizes:** 2048, 3072 bits	✓				✓
RSA Encryption and Decryption Non-Approved sizes or functions	RSAES-OAEP encryption and decryption [FIPS186-4,PKCS1] with non-approved sizes	RSA SHA1, SHA-224, SHA-256, SHA-384, SHA-512 PKSC1.5 with MD5 DRBG	**Input:** User keys (I) **Sizes:** 512, 1024 bits	✓				✓
RSA Encryption and Decryption Non-Approved sizes or functions	RSAES- PKCS1-v1.5 encryption and decryption [FIPS186-4,PKCS1] with non-approved sizes	RSA SHA1, SHA-224, SHA-256, SHA-384, SHA-512 PKSC1.5 with MD5 DRBG	**Input:** User keys (I) **Sizes:** 512, 1024 bits	✓				✓
RSA Encryption and Decryption primitives Non-Approved Schemes	Hashing and exponentiation with no padding scheme. Insecure, must not be used directly.	RSA SHA1, SHA-224, SHA-256, SHA-384, SHA-512, MD5.	**Input:** User keys (I) **Sizes:** 512, 1024, 2048, 3072, 4096 bits	✓				✓
Diffie-Hellman Key Generation Approved	Key Generation Domain Parameter Generation [SP800-56A]	Use DRBG	**Input:** User keys (O) **Sizes:** 2048 bits	✓		✓		✓
Diffie-Hellman Key Generation Non-Approved strengths	Key Generation Domain Parameter Generation [SP800-56A]	Use DRBG	**Input:** User keys (O) **Sizes:** 1024 bits	✓				✓
Diffie-Hellman Key and Domain Verification Approved	Key Verification Domain Parameter Verification [SP800-56A]		**Input:** User keys (I)	✓		✓		✓
Diffie-Hellman Key Exchange Approved	Key Agreement [SP800-56A]	SHA functions	**Input:** User keys (I) **Output:** User keys (O) **Sizes:** 2048	✓		✓		✓
Diffie-Hellman Key Exchange	Key Agreement [SP800-56A]	SHA functions	**Input:** User keys (I) **Output:** User keys (O) **Sizes:** 1024	✓				✓

Service Name	Purpose	Security Functions	Keys and CSPs	User Role	CO Role	Approved	Public Core	Secure Core
Non-Approved strengths								
DRBG Instantiation	DRBG Context Instantiation	Creates a DRBG context, using TRNG internally for seeding.	**Output:** DRNG context (V, Key) (O)	✓		✓		✓
DRBG Reseeding	DRBG Reseeding with optional Additional Input [SP800-90A]		**Input/Output:** DRBG context (V, Key) (I, O)	✓		✓		✓
DRBG Generation	Generate Random Vector Generate Random Vector in Range [SP800-90A]	CTR-DRBG (AES-256 in CTR mode)	**Input/Output:** DRBG context (V, Key) (I,O)	✓		✓		✓
DRBG Testing	Enable KAT mode Disable KAT mode	Auxiliary functions used to perform a Known-Answer Test[4]		✓		✓		✓
Generation Secure Key Package	See section 2.3.4.7 "Secure Key Mechanism"	AES-128 CCM (package encryption)	**Input:** Session Key (R) User Keys (I) **Output**: Secure Key package (O)	✓		✓		✓
Secure Key Approved Algorithms	Encryption Decryption	AES-128 CCM, AES-128, AES-256 in CBC, CTR, OFB, CTS (CBC-CS1) modes	**Input:** Session Key (R), Secure Key package (I)	✓		✓	✓	
Secure Key Non-Approved Algorithms	Encryption Decryption	AES-128 CCM; Multi2 in CBC, CTR, OFB, CTS (CBC-CS1) modes.	**Input:** Session Key (R) Secure Key package (I)	✓			✓	
Secure Copy	Secure Key operation without encryption or decryption	AES-128 CCM (package decryption and verification)	**Input:** Session Key (R) Secure Key package (I)	✓		✓	✓	
Lifecycle State (LCS) Read	High-level interface to the LCS			✓		✓		✓
RMA (Return Merchandise Authorization)	Change Life Cycle State (LCS) to RMA, zeroize module	Uses RSA-PSS and SHA-256 to verify a RMA certificate	**Input:** Hash of Boot Key (R) RSA public in RMA certificate (I) **Sizes:** 2048 bits **Zeroize:**		✓	✓		✓

[4] The functions are intended for power-on testing only. The policy instructs the operator not to call those functions at run time. The module is not operating in Approved mode if an operator doesn't follow the policy.

Service Name	Purpose	Security Functions	Keys and CSPs	User Role	CO Role	Approved	Public Core	Secure Core
			Device root key, Provision root master key, and Firmware code encryption key					
Secure Debug	Enter Secure Debug state	Uses RSA-PSS and SHA-256 to verify a certificate	**Input:** Hash of Boot Key (R) RSA public key in debug certificate (I) **Sizes:** 2048 bits		✓	✓		✓
Non-Volatile Memory Manager	Monotonic Counter Get/Set			✓		✓		✓
Reset	Warm reset Perform self-tests	Clears keys stored in internal memory and registers (but not in OTP)	**Zeroize:** User keys (Z) Platform keys in HW registers (Z)	✓		✓	✓	✓
Secure Timer	Get Timestamp and Compare Timestamp			✓		✓		✓
Identify SOC	Return SOC ID, unique device ID derived from unique values stored in OTP, used in Secure Boot and Secure Debug.	Uses NIST SP 800-108 KBKDF and SHA-256.	**Input:** Device root key (R) Hash of Boot key (R) **Output:** SoC ID (O)	✓		✓		✓
Secure Boot — Firmware Certificate Verification	Certificate Chain verification Firmware image verification	Uses RSA-PSS, SHA-256 to verify boot certificate and firmware contents	**Input:** Hash of Boot Key (R) RSA public key in boot certificate (I) **Sizes:** 2048 bit	✓		✓		✓
Firmware Update	Module firmware update as an integrated part of device firmware update	Uses RSA-PSS, SHA-256 to verify firmware image certificate and firmware contents	**Input:** Hash of Boot Key (R) RSA public key in boot certificate (I) **Sizes:** 2048 bit	✓		✓		✓
Session Key Generation	Derive Session Key from Device Root Key and DRBG output	Uses the NIST SP800-108 KBKDF; Uses DRBG	**Input:** Device Root Key (R) **Output:** Session Key (W)	✓		✓		✓
External RAM Backup and Restore	Backup and restore external RAM buffers securely	UsesAES-128 CCM for encryption and authentication	**Input:** Session Key (R)	✓		✓		✓

Service Name	Purpose	Security Functions	Keys and CSPs	User Role	CO Role	Approved	Public Core	Secure Core
Customer Asset Provisioning Decrypt Customer Key[5]	Decrypts Encrypted Customer Key into Customer Key	AES-128 ECB	**Input:** Platform Key (R) Encrypted Customer Key (R) **Output:** Customer Key (W)	✓		✓		✓
OEM Asset Provisioning Get Provisioning Key[5]	Derives OEM Provisioning Key from Platform Key and Hash of Boot Key Hash	Uses the NIST SP 800-108 KBKDF	**Input:** Platform Key (R) Hash of Boot Key (R) **Output:** OEM Provisioning Key (O)	✓				✓
OEM Asset Provisioning Asset Unpacking	Derives an asset key from OEM Provisioning Key and Asset ID, authenticate and decrypt an asset package	Uses the NIST SP 800-108 KBKDF AES-128 CCM to decrypt asset	**Input**: OEM Provisioning Key (I)	✓				✓
Show Status (FipsGetState)	Indicates the FIPS status: Approved, Non-Approved, or Error state (includes error code).			✓		✓	✓	✓

Table 6 Services

4.3 Operator Authentication

Role authentication in MTK CryptoCore is implicit. The Crypto Officer Role is defined by access to the initialization and recoveray services. Specific Life Cycle states are designated for the Crypto Officer role. At manufacturing time, the module assgins the operator to the Crypto Officer role until the OTP is initialized, putting the device into the Operational Life Cycle State. At that point, the module assgins the operator to the User role.

The module remains assigned to the User Role permanently, unless the operator moves the module into one of the special Life Cycle States intended for fualt discovery (Secure Debug and RMA), which are again assigned to the Crypto Officer role.

4.4 Mechanism and Strength of Authentication

No authentication is required at security level 1; authentication is implicit by assumption of the role.

[5] The OTP configuration determines which one of the two services – Customer Asset Provisioning and OEM Asset Provisioning – is offered per device. The OEM Asset Provisioning service is not FIPS-Approved because the OEM Key may be output to the user before the completion of module self-tests.

5. Physical Security

The MTK CryptoCore Cryptographic Module is a sub-chip cryptographic subsystem and firmware-hybrid module. The SoC containing the sub-chip module and the memory chips is a production-grade component. The final device which contains the SoC and memory chips are entirely contained within a hard plastic production-grade enclosure.

6. Operational Environment

The module hardware which is integrated in the SoC and the module firmware which resides in flash device are all contained in a commercial mobile device manufactured by the device OEM. The module does not have a mechanism to update its firmware. In addition the module uses integrity techniques to enforce the non-modification of its firmware. Therefore, the operational evironment is considered non-modifiable.

7. Cryptographic Key Management

MTK CryptoCore works with two types of keys: platform keys and user keys.

7.1 User Keys

User keys are created and owned by the module's user that is the application firmware running on the Host CPU in the Secure and Public world. From the module's perspective, user keys are transient, and the module never stores them in non-volatile memory. The module may copy user keys into its internal RAM and cryptographic engine registers, and will clear them on completion of every operation and on zeroization.

Key Type	**Key Size**
AES	128, 192, 256 bits
DES	64 bits
MULTI2	64 bits
Triple-DES	128, 192 bits
HMAC	Up to 2^{26} bytes
KBKDF	Up to 4080 bytes
SP800-135	Up to 2048 bytes
KDF1, KDF2	Up to 2048 bytes
ECC	160, 192,224, 256, 384, 521 bits
RSA	512, 1024, 2048, 3072, 4096 bits
Diffie-Hellman	1024, 2048 bits

Table 7 User Keys

7.2 Platform Keys

Table 8 summarizes the keys defined in the modules. Platform keys may be derived during the boot-up or on demand. Most platform keys are limited by hardware to use in a few specific operations, but some derived keys are output to the user.

The OEM Provisioning Key, is derived and outputted to the user before the module completes self-tests, and therefore FIPS-certified modules are configured to use the alternative provisioning key called Customer Key whose encrypted version is stored in the OTP.

Name	**Type**	**Length (bit)**	**Purpose**	**Storage**	**Entry and Derivation**	**Zeroization**
Device Root Key	AES	256	Derivation of device specific keys	OTP and HW register Plaintext	Provisioned into OTP during manufacturing	OTP: command to set key to all 1's Register: via power-on reset
Provisioning Root Master Key	AES	128	Derivation of the Provisioning Master Key	OTP Plaintext	Provisioned into OTP during manufacturing	command to set key to all 1's
Provisioning Master Key	AES	128	Derivation of the Platform Key	HW register Plaintext	Derived using CKG from Provisioning Root Master Key	via power-on reset
Platform Key	AES	128	Either i) decryption of Customer Key or ii) derivation of OEM Provisioning Key[6]	HW register Plaintext	Derived using KBKDF[7] from Provisioning Master Key and CSP in OTP	via power-on reset
Encrypted Customer Key	AES	128	Customer Key encrypted with Platform Key	OTP Encrypted	Provisioned into OTP during manufacturing	N/A
Customer Key	AES	128	Used as key for customer assets	HW register Plaintext	The result of decryption of Encrypted Customer Key with the Platform Key	via power-on reset
OEM Provisioning Key	AES	128	OEM asset decryption and verification	SRAM Plaintext	Derived using KBKDF from Platform Key and CSP in OTP	via power-on reset
Public Key Hash	RSA or ECDSA	256	Used for signature verification	OTP Plaintext	Provisioned into OTP during manufacturing	command to set key to all 1's
Firmware Code Encryption Key	AES	128	Optionally used to decrypt the firmware image	OTP and SRAM Plaintext	Provisioned into OTP during manufacturing	OTP: command to set key to all 1's SRAM: via power-on reset

[6] In FIPS-compliant modules, only used for decryption of Customer Key

[7] All mentions of the KBKDF function in this table refer to the NIST SP800-108 Key Derivation function described above in Services, Section 4.2.

Session Key	AES	128	Encryption of Secure Key packages Encryption of RAM backups	HW register Plaintext	Derived using KBKDF from a 96-bit random value and CSP in OTP	via power-on reset
Endorsement key pair	ECDSA	256	ECDSA signing of device messages	SRAM Plaintext	Derived from CSP in OTP according to FIPS 186-4 B.4.1	via power-on reset
RPMB shared key	HMAC	256	Computing HMACs for RPMB data frames	SRAM Plaintext	Derived from Device Root Key using NIST SP800-108 KDF	via power-on reset

Table 8 Platform Keys and CSPs

7.3 Key Generation

MTK CryptoCore uses a DRBG compliant with the NIST SP 800-90A standard [SP800-90A], seeded with 384 bits from a TRNG, at least 349 bits of entropy. See Sections 2.3.4.1 and 2.3.4.2 for details on the TRNG and DRBG implementations.

Asymmetric key generation services offered by the module are defined by the [FIPS186-4] standard, using the DRBG service for random inputs, and are detailed in Section 4.2. No dedicated service is offered for symmetric key generation; the user is instructed to directly use the output of the DRBG service.

Internally, the module uses the DRBG to generate the Session Key and to provide random inputs to RSA and ECIES cryptographic services that use randomness.

7.4 Key Establishment

Key establishment and key derivation services offered by the module are detailed in Section 4.2. The module offers key establishment services based on the approved Elliptic Curve Diffie-Hellman and Finite Field Diffie-Hellman [SP800-56A], and the non-approved Elliptic Curve Integrated Encryption Scheme [IEEE1363]. The key establishment services are only allowed when using keys allowed by [SP800-131A]. The module does not offer key establishment services based on the RSA algorithm.

The module offers the approved key derivation services using Key Derivation Functions as defined in NIST [SP800-108], NIST [SP800-135]: KBKDF, and ASN1DER.

In addition, Non-Approved key derivation services based are available for use in Non-Approved Mode, implementing the KDF1 and KDF2 functions as defined in [ISO/IEC-18033-2].

For internal key establishment, the module's Approved services follow the guidelines of NIST [SP800-57] with regards to required key strengths. All internal key derivations are performed with approved algorithms, and the derived keys are of strength less or equal than the source key, ensuring that the derived keys have full cryptographic strength, for all Approved key sizes.

7.5 Key Entry and Output

The module supports electronic key entry methods only. The module does not support manual key import or export methods. User keys are input and output electronically, in plaintext, as parameters to the module's firmware APIs. Platform keys are only inputted into the module during manufacturing.

The module outputs RPMB shared key, OEM provisioning Key, and endorsement key pair to secure world for consumption by secure OS.

The module does not received seeds and seed keys from the outside. The DRBG is only seeded internally from the TRNG; the DRBG does support a service to receive additional data from the user, according to the definitions in NIST SP 800-90A.

The Secure Key mechanism (Section2.3.4.7) is a form of encrypted key entry and output. In the Secure Core, a user key is wrapped using AES CCM with the Session Key; the wrapped key is then decrypted and verified in the Public Core by the SEP subsystem.

7.6 Key Storage

User keys within the module are stored only temporarily and in plaintext form, as described in Section 7.1.

Platform keys are stored in OTP, and may be kept in HW registers or SRMA at runtime (see Table 8).

7.7 Key Zeroization

The user can zeroize user keys stored in the module at any time by putting the module through a power-on reset. The reset clears hardware registers and SRAM contents. The module specifically erases the portions of SRAM which may hold keys, CSPs and intermediate values: the PKA SRAM and the work buffer described in Section 7.1 are cleared on every operation, and following reset.

For platform keys, user can trigger the key zeroization via one of the two methods:

a) Run a special boot loader that can be flashed to the device
   - Flash a special boot loader to the device and power up
   - The boot loader enables RMA state
   - Set all 1's for all the platform keys that are stored in OTP

b) Use a PC tool to send command to the device
   - Send a zeroization command from PC tool to the device
   - Device firmware interpreters the command and enables RMA state
   - Set all 1's for all the platform keys that are stored in OTP

8. Electromagnetic Interference / Compatibility (EMI/EMC)

According to 47 Code of Federal Regulations, Part 15, Subpart B, Unintentional Radiators, Chapter 15.101, "No authorization is required for a peripheral device or a subassembly that is sold to an equipment manufacturer for further fabrication; that manufacturer is responsible for obtaining the necessary authorization prior to further marketing to a vendor or to a user." As this product is a subassembly sold for further fabrication, it is exempt from part 15 of the FCC rules.

9. Self Tests

According to FIPS 140-2 requirements, MTK CryptoCore performs a number of self-tests on power-up, conditionally on selected events. Those tests are detailed below.

9.1 Power-up Tests

The following tests are performed on module initialization, before the operator can request any of the module's cryptographic services. The tests are performed automatically, without needing any action from an operator. A platform-global variable is used to synchronize the state of the power-on self-tests between the Secure Core and the Public Core, preventing data output before self-tests complete.

A failure of any of these tests will cause the module to enter the Error state, which is indicated by the status output. On success, the module will enter the Approved or Non-Approved state as requested by the Host, and will change its status output accordingly.

The module does not support a dedicated service or API to invoke those tests on demand; the operator can initiate a power-on reset, to have the module perform the tests.

9.1.1 Cryptography Test

The module performs Known-Answer Tests (KAT) on power-up with the cryptographic algorithms as listed below. All tests perform a comparison versus a stored reference value, unless explicitly noted for pairwise tests.

9.1.1.1 Tests in Public Core

The following tests are performed in the Public Core.

- AES encryption and decryption operations, in ECB, CBC, OFB, CTR, CTS (CBC-CS1), GCM and GMAC modes, using 128, 192, and 256-bit key sizes.
- XTS-AES encryption and decryption operations, using 256 bit key sizes.
- AES MAC generation in CCM mode, using 128, 192, and 256-bit key sizes.
- Triple-DES encryption and decryption operations, ECB and CBC modes, with 3-key bundles.
- Hash generation for SHS, with SHA-1, and SHA-256 tested.
- HMAC generation for HMAC SHA-256.

9.1.1.2 Tests in Secure Core

The following tests are performed in the Secure Core.

- AES encryption and decryption operations, in ECB, CBC, OFB, CTR and CTS (CBC-CS1) modes, using 128, 192, and 256-bit key sizes.
- XTS-AES encryption and decryption operations, using 256 and 512 bit key sizes.
- AES MAC generation in CCM mode, using 128, 192, and 256-bit key sizes.
- AES MAC generation with CBC-MAC using 128-bit and 256-bit keys[7]

- Triple-DES encryption and decryption operations, ECB and CBC modes, with 2-key and 3-key bundles.
- Hash generation for SHS, with SHA-1, SHA-256 and SHA-512 tested.
- HMAC generation for HMAC SHA-256.
- DRBG Instantiate, Reseed, Additional Data, random vector generation.
- RSA encryption and decryption with 2048-bit modulus, using PKCS#1v2.1 OAEP scheme.
- RSA signature generation and verification with 2048-bit modulus, using PKCS#1v2.1 PSS scheme. The PSS scheme has its random input disabled for this test.
- ECDSA signature generation and verification using the P-256 curve, with the SHA-256 message digests.
- Diffie-Hellman – Primitive “z”’ operation (as part of a NIST SP800-56A key agreement protocol), with prime size of 2048 bits.
- Elliptic Curve Diffie-Hellman – Primitive “z”’ operation (as part of NIST SP800-56A key agreement protocol), using the P256 curve.

9.1.2 Firmware Integrity Test

The module performs firmware integrity tests as part of the boot sequence.

The Secure Core ROM integrity is checked using the SHA-256 function, computed over the entirety of the ROM and compared with a value stored in the firmware image.

The SEP ROM integrity is checked using an Error Detection Code (EDC), namely the 16-bit “Internet Checksum” specified in [RFC1071], computed over the entirety of the ROM.

The Firmware integrity is checked by ROM code as part of the Secure Boot, using RSA-2048 PSS scheme with the SHA-256 hash function.

9.2 Conditional Tests

Upon RSA and ECC key generation, a conditional test is run. Since the key’s intended use (for encryption and decryption, or for signature creation and verification) is unknown at the time of generation, each algorithm is tested for pairwise consistency with one usage: RSA is tested with an encryption/decryption operation, and ECC is tested with a signature generation/verification operation.

Continuous RNG tests are implemented in the module’s two random number generators — the TRNG and the DRBG. In both, each block of output generated is compared for equality with the previous block (exempting the very first block generated). The TRNG uses a block size of 16 bits. The DRBG uses a block size of 128 bits. A failure of each Continuous RNG Test will put the entire module into the Error State.

Conditional firmware load test is implemented in the module. The module firmware, as an integrated part of the device firmware, is updated when the device firmware is updated. The module’s secure core performs the integrity check for all the firmware images being updated.

10. Design Assurance

10.1 Configuration Management

10.1.1 Software

GIT is used for software source control and configuration management. It supports full history and version tracking.

10.1.2 Hardware

Perforce is used for hardware code and configuration management. It supports full history and version tracking.

10.2 Delivery and Operation

For the public core firmware in kernel and secure core firmware in secure OS, they are delivered by GIT open source. For the crypto driver in boot-loaders, they are delivered by binary format. All of the images are signed and protected by secure boot. The cryptographic model is built-in and can't be disabled by any feature option.

10.3 Guidance

10.3.1 Operator Guidance

The guidance for the operator are listed below

- The operator is instructed not to use functions and key sizes marked as Non-Approved in FIPS mode.
- The operator is instructed to follow the additional caveats on approved functions (Section 2.2.1)
- The operator is instructed not to invoke the insecure RSA primitives directly, and to use the approved encryption and signature schemes instead.
- When using the module's services for key derivation and key establishment, the operator is instructed to follow NIST guidelines on the relative strengths of source key and derived key, or the asymmetric key used to establish a shared symmetric key.
- When using the module's DRBG for key generation, the operator is warned to use a DRBG with a 256-bit key size and state, in order to generate keys with strengths of 256 bits. Instantiating a DRBG context with 128-bit key size is only suitable for keys with strength of 128 bits.
- When receiving another party's public key in the Diffie-Hellman and Elliptic-Curve Diffie-Hellman protocols, the operator is instructed to invoke the Domain Parameter Verification service in order to fulfill the requirements of NIST SP800-56A Section 5.6.2.

10.4 Proprietary Document

The following proprietary documents are available to the customers for this module:

- MTK CryptoCore TEE Runtime Library API Documentation
- MTK CryptoCore TEE SBROM Library API Documentation
- MTK CryptoCore TEE Software Integration Guidelines

- MTK CryptoCore TEE Hardware Datasheet
- MTK CryptoCore REE Hardware Datasheet
- MTK CryptoCore TEE Hardware Technical Reference Manual
- MTK CryptoCore REE Hardware Technical Reference Manual
- MTK CryptoCore REE Linux Driver Documentation

11. Mitigation of Other Attacks

The cryptographic module is not designed to mitigate specific attacks.

Bibliography

[ANSI-X9.42] ANSI, ANSI X9.42:2003 Agreement of Symmetric Keys Using Discrete Logarithm Cryptography, 2003, [Online] http://webstore.ansi.org/RecordDetail.aspx?sku=ANSI+X9.42-2003+(R2013)

[ANSI-X9.63] ANSI, ANSI X9.63 Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport using Elliptic Curve Cryptography, 2011, [Online] http://webstore.ansi.org/RecordDetail.aspx?sku=X9.63-2011

[ARIB-STD-B25] ARIB, ARIB STD-B25: Conditional Access System Specifications for Digital Broadcasting, 2007, [Online] http://www.arib.or.jp/english/html/overview/doc/6-STD-B25v5_0-E1.pdf

[FIPS180-4] FIPS, FIPS PUB 180-4: Secure Hash Standard (SHS), 2015, [Online] http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

[FIPS186-4] FIPS, FIPS PUB 186-4 - Digital Signature Standard (DSS), 2013, [Online] http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

[FIPS197] FIPS, FIPS 197 - Advanced Encryption Standard (AES), Nov 2001, [Online] http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[FIPS198-1] FIPS, FIPS PUB 198-1: The Keyed-Hash Message Authentication Code (HMAC), 2008, [Online] http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf

[FIPS46-3] FIPS, FIPS PUB 46-3 : Data Encryption Standard (DES), 1999, [Online] http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[IEEE1363] IEEE, IEEE 1363-2000: Standard Specifications For Public Key Cryptography, 2000, [Online] http://grouper.ieee.org/groups/1363/P1363/index.html

[ISO/IEC-18033-2] ISO/IEC, ISO/IEC 18033-2, 2006, [Online] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=37971

[ISO/IEC-9797-1:2011] ISO/IEC, ISO/IEC 9797-1:2011 - Information technology -- Security techniques -- Message Authentication Codes (MACs) -- Part 1: Mechanisms using a block cipher, 2011, [Online] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50375

[ISO/IEC-9979] ISO/IEC, ISO/IEC 9979: Register of Cryptographic Algorithms (withdrawn), 1999, [Online] http://www.iso.org/iso/catalogue_detail?csnumber=28107

[JESD84] JEDEC, JEDEC Standard, Embedded Multimedia Card (eMMC), Electrical Standard - Version 5.0, 2014, [Online] http://www.jedec.org/sites/default/files/docs/JESD84-B50.pdf

[PKCS1] R. S. A. Laboratories, PKCS1 RSA Cryptography Standard, 2012, [Online] http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/pkcs-rsa-cryptography-standard.htm

[RFC1071] R. Braden, D. Borman, and C. Partridge, Computing the Internet Checksum, 1988, [Online] https://tools.ietf.org/html/rfc1071

[RFC1321] Ronald Rivest, The MD5 message-digest algorithm, 1992, [Online] https://www.ietf.org/rfc/rfc1321.txt

[RFC2104] Hugo Krawczyk, Mihir Bellare, and Ran Canetti, RFC 2104: HMAC: Keyed-hashing for message authentication, 1997, [Online] https://www.ietf.org/rfc/rfc2104.txt

[RFC3566] Sheila Frankel and Howard Herbert, RFC 3566-The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec, 2003, [Online] https://www.ietf.org/rfc/rfc3566.txt

[SP800-108] Lily Chen, NIST Special Publication 800-108: Recommendation for Key Derivation Using Pseudorandom Functions, 2009, [Online] http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf

[SP800-131A] Elaine Barker and Allen Roginsky, NIST Special Publication 800-131A: Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths, 2011, [Online] https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-131a.pdf

[SP800-133] Elaine Barker and Allen Roginsky, NIST Special Publication 800-133: Recommendation for Cryptographic Key Generation, 2011, [Online] http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133.pdf

[SP800-135] Quynh Dang, NIST Special Publication 800-135 Revision 1: Recommendation for Existing Application-Specific Key Derivation Functions, 2011, [Online] http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-135r1.pdf

[SP800-38A-add] Morris Dworkin, NIST Special Publication 800-38A Addendum: Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for

CBC Mode, 2010, [Online] http://csrc.nist.gov/publications/nistpubs/800-38a/addendum-to-nist_sp800-38A.pdf

[SP800-38A] Morris Dworkin, NIST Special Publication 800-38A: Recommendation for Block Cipher Modes of Operation, 2001, [Online] http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

[SP800-38B] Morris Dworkin, NIST Special Publication 800-38B: Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, 2005, [Online] http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf

[SP800-38C] Morris Dworkin, NIST Special Publication 800-38C: Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, 2004, [Online] http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C_updated-July20_2007.pdf

[SP800-38D] Morris Dworkin, NIST Special Publication 800-38D: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, 2007, [Online] http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf

[SP800-38E] Morris Dworkin, NIST Special Publication 800-38E: Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices, 2010, [Online] http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf

[SP800-38F] Morris Dworkin, NIST Special Publication 800-38F: Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping, 2012, [Online] http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf

[SP800-56A] Elaine Barker, Lily Chen, Allen Roginsky, and Miles Smid, NIST Special Publication 800-56A Revision 2: Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, 2013, [Online] http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.pdf

[SP800-57] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid, NIST Special Publication 800-57: Recommendation for Key Management - Part 1: General (Revision 3), 2012, [Online] http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf

[SP800-67] William C. Barker and Elaine Barker, NIST Special Publication 800-67: Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, 2012, [Online] http://csrc.nist.gov/publications/nistpubs/800-67-Rev1/SP-800-67-Rev1.pdf

[SP800-90A] NIST, NIST Special Publication 800-90 A: Recommendation for Random Number Generation Using Deterministic Random Bit Generators, 2015, [Online] http://dx.doi.org/10.6028/NIST.SP.800-90Ar1