Arm® TrustZone® CryptoCell-712

Revision 1.19

FIPS 140-2 Non-Proprietary Security Policy

Non-confidential

arm

Copyrights and Trademarks

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

Contents

List of Figures

List of Tables

1 Cryptographic Module Specification

This document is the non-proprietary security policy for Arm® TrustZone® CryptoCell-712. This security policy describes how CryptoCell-712 meets the security requirements of FIPS 140-2, and how to operate CryptoCell-712 securely, in a FIPS-compliant manner. This policy is submitted as part of the Federal Information Processing Standard (FIPS) 140-2 Level 1 validation process of CryptoCell-712. For more information about the Cryptographic Module Validation Program (CMVP), see: http://csrc.nist.gov/groups/STM/cmvp/.

CryptoCell-712 is a security engine with a root of trust and cryptographic accelerator capabilities. It is intended for use in an SOC (System on Chip), where it provides foundational security services for the entire platform, including cryptography, key management, platform identity, secure boot, secure Life Cycle State (LCS), and secure debug. It offers high-throughput cryptography engines suitable for a diverse set of use cases, such as secure playback of DRM (Digital Rights Management) protected media content, IPsec VPNs, TLS/SSL link protection, drive encryption and more.

Under the FIPS 140-2 definitions, the CryptoCell-712 is a firmware-hybrid module and a sub-chip module. The module includes hardware, defined in the RTL (Register Transfer Language), and firmware for execution on the host CPU, making it a hybrid module. The module's RTL is offered for integration as part of a silicon partner's (customer's) hardware host — an SOC, containing the cryptographic module's hardware alongside the Host CPU, memories and peripherals, making it a sub-chip component.

The following table depicts the security level claimed for each of the eleven sections that comprise the FIPS 140-2:

Table 1: Security levels

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

1.1 Module integration

CryptoCell-712 is primarily intended for integration as an IP (Intellectual Property) block into a partner's silicon product. The module is provided to partners as Silicon IP and accompanying firmware. The partner integrates the silicon with the host CPU hardware, and installs the firmware into the host NVRAM (Non Volatile Random Access Memory).

For the purposes of this Cryptographic Module Validation the module was tested on the expanded Arm® Juno rev.2 in a configuration reflecting partner's hosting SoC implementation. The expanded board contains compute subsystem running the operating firmware connected to the LogicTile FPGA (Field Programmable Gate Array) expansion with the synthesized hardware.

For ease of notation it shall be referred throughout the document as Arm® Juno board. Instead of the partner's applications, Arm® provides test software used to invoke the module's services. The FPGA board does not have some properties of a real silicon product, most importantly OTP (One-Time Programmable memory) which is only simulated (see Section 1.4.1.2.3).

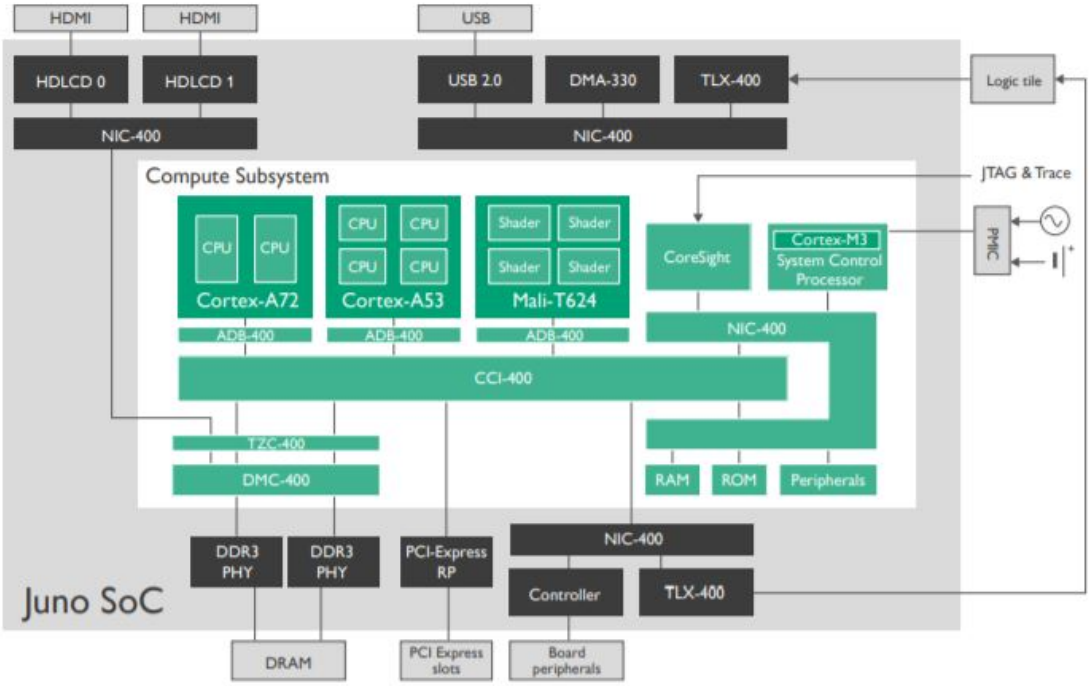

Figure 1: Arm® Juno rev.2 board - hardware architecture

The implementation of the module is located on the LogicTile FPGA expansion as presented in Figure 1.

Figure 2: LogicTile FPGA board front

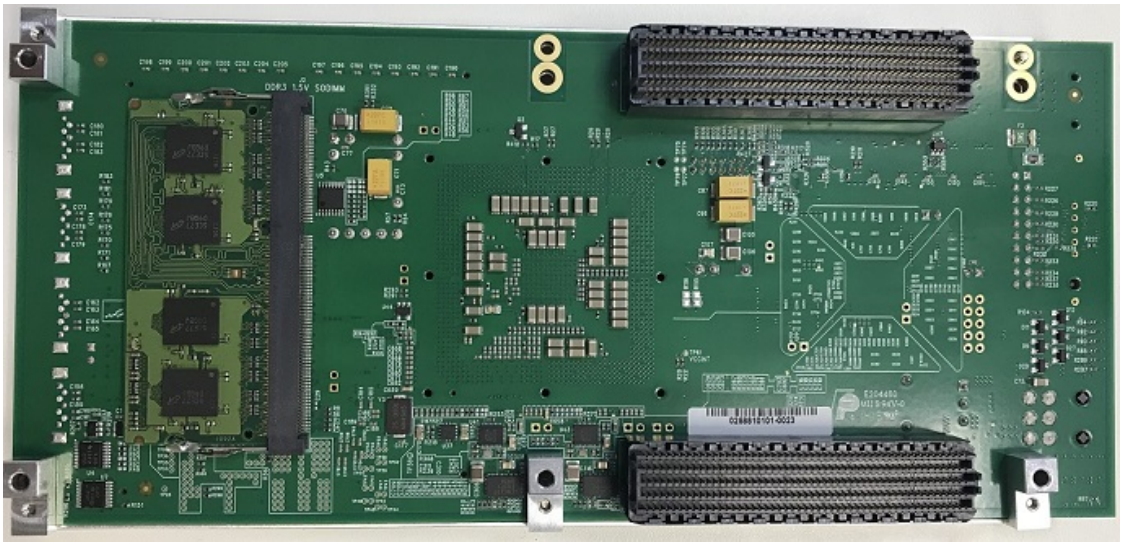

Figure 3: LogicTile FPGA board back

1.2 TrustZone® architecture, TEE and REE

CryptoCell-712 is intended for use in an Arm® TrustZone® platform, where a single Arm® Host processor runs two separate environments: a TEE (Trusted Execution Environment), and a REE (Rich Execution Environment). The TrustZone® architecture uses a single CPU to run both the TEE and the REE, with dedicated hardware enforcing the separation between control states, data and memories belonging to the different components.

The firmware in the TEE is generally compact and strictly controlled by the manufacturer, and is typically dedicated to providing security services to applications in REE. The firmware in the REE consists of an OS (Operating System) kernel, controlled by the manufacturer, and an application layer, typically modifiable by the user. The host OS for this certification is Linux OS kernel version 3.18. The module is compatible with multiple Linux variants including Android.

The version numbers for this certification are:

- Hardware 712
- TEE firmware 1.1.0.48
- TEE secure boot ROM 1.0.0.1145
- REE firmware 1.1.0.49

The CryptoCell-712 hardware has some dedicated TEE and REE components, providing functionality to the corresponding host environment, and some shared components, including the Symmetric Cryptography Engine. The shared engine alternates between TEE and REE states, with hardware enforcement of data and memory separation similar to that of the host CPU. Other shared components communicate between the TEE and REE. A high level diagram of the module is shown in Figure 4.

1.3 Approved security functions and mode of operation

1.3.1 Approved security functions

CryptoCell-712 supports FIPS-approved security functions, as specified in Table 2. The following notes and caveats apply:

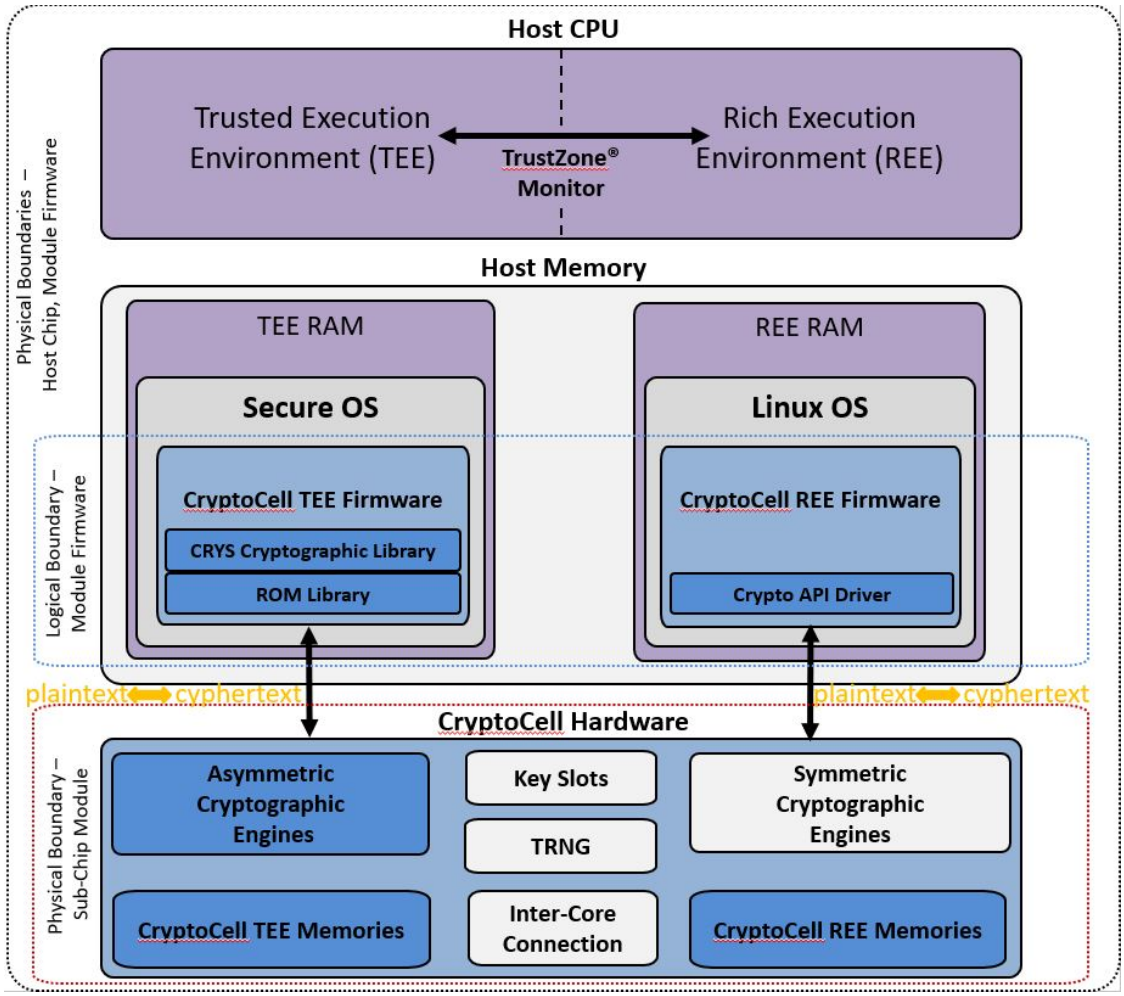

Figure 4: CryptoCell-712 high level diagram

- Triple-DES: To comply with [SP800-67], the user must not exceed 2^{28} encryptions with the same key.
- SHA-1: not approved for signature generation, only for legacy signature verifications. Approved for other uses.
- 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 (KAS FFC): The CAVP certificate covers all of SP800-56A except KDF, which was not tested.
- Key Agreement — Diffie-Hellman and EC-DH (Elliptic Curve Diffie Hellman) functions are only approved when used as part of a [SP800-56A] key agreement protocol, otherwise only allowed.
- Key Agreement — Diffie-Hellman and EC-DH: 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.
- RSA and ECDSA: Only Approved hash functions must be used for signature generation. SHA-1 is only Approved for legacy signature verification.

Table 2: Approved security functions

CAVP Cert	Algorithm	Standard	Mode / Method	Key Lengths, Curves or Moduli	Use	REE	TEE
REE: #4743 TEE: #4749	AES	[FIPS-197, SP800-38A, SP800-38E]	ECB, CBC, OFB, CTR, XTS-AES	128, 192, 256, 512 (XTS)	Encryption / decryption	✓	✓
Vendor-affirmed	AES	[SP800-38Aadd]	CBC-CS1	128, 192, 256	Encryption / decryption	✓	✓
REE: #4743 TEE: #4749	AES	[SP800-38C]	CCM	128, 192, 256	Authenticated encryption / decryption	✓	✓
REE: #4743 TEE: #4749	AES	[SP800-38B]	CMAC	128, 192, 256	Message authentication	✓	✓
REE: #2522 TEE: #2523	Triple-DES	[FIPS-46-3, SP800-67]	ECB, CBC	192	Encryption / decryption	✓	✓
REE: #3887 TEE: #3892	SHS	[FIPS-180-4]	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512		Message authentication	✓	✓
REE: #3158 TEE: #3163	HMAC	[FIPS-198-1, RFC2104]	HMAC SHA-1, SHA-224, SHA-256, SHA-384, SHA-512		Message authentication	✓	✓
#4743	AES	[JEDEC, 6.3.4, 9.4] [FIPS-197, SP800-38A]	CBC for ESSIV	128, 256	Encryption / decryption	✓	
#4743	AES	[JEDEC, 6.3.2, 9.2] [FIPS-197, SP800-38A]	CBC for BitLocker	128, 256	Encryption / decryption	✓	
#2593	RSA	[FIPS-186-4]	SHA functions PSS, PKCS1-v1.5	2048, 3072	Signature Generation / Verification / Key Generation		✓
#2596	RSA (Firmware Certificate Verification)	[FIPS-186-4]	PSS; SHA-256	2048	Signature Verification		✓
#1385	CVL (KAS FFC)	[SP800-56A]	FFC	(2048, 224), (2048, 256)	Shared Secret Computation		✓
#1385	CVL (ECC CDH)	[SP800-56A]	ECC	P-224, P-256, P-384, P-521	ECC CDH Primitive		✓
#1184	ECDSA (Elliptic Curve Digital Signature Scheme)	[FIPS-186-4]	SHA functions	P-224, P-256, P-384, P-521	Signature Generation / Verification / Key Pair Generation / Public Key Validation		✓
#151	KBKDF (Key Based Key Derivation Function)	[SP800-108]	Counter mode	CMAC AES-128, 256	Key derivation		✓
#1386	CVL (ANS 9.63)	[SP800-135, ANSI-X9.63]	Section 5.1, ANS X9.63-2001		Key derivation		✓
#1630	DRBG (Deterministic Random Bit Generator)	[SP800-90A]	CTR-DRBG	256	Random Bit Generation		✓
Vendor-affirmed	CKG	[SP800-133]			Input to asymmetric key generation		✓

- ESSIV: AES-CBC algortihm with proprietary method for IV generation. The encryption key is different from the key used in the IV generation operation. Both keys are same size, either 128 or 256 bit.
- Bitlocker: AES-CBC algortihm with proprietary method for IV generation. The encryption key is different from the key used in the IV generation operation. Both keys are same size, either 128 or 256 bit.

1.3.2 Allowed security functions

CryptoCell-712 TEE also supports several FIPS-allowed security functions, as specified in Table 3.

Table 3: Allowed security functions

Algorithm	Caveat	Use
True Random Number Generator [NDRNG]	There is no NIST-approved standard for NDRNG	DRBG seeding and reseeding
RSAES-OAEP [PKCS1] Moduli: 2048, 3072	NIST allows RSA encryption for key wrapping only, but does not approve it Key establishment strength 112 or 128 bits	Encryption/decryption
RSAES-PKCS1-v1_5 [PKCS1] Moduli: 2048, 3072	NIST allows RSA encryption for key wrapping only, but does not approve it 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] Curves: p224k1, p256k1	Key establishment strength 112 or 128 bits	Key agreement
ECDSA (Elliptic Curve Digital Signature Scheme) Curves: p224k1, p256k1	NIST allows non-NIST recommended curves of at least 112 bit security strength	Signature Generation / Verification / Key Pair Generation / Public Key Validation

The following notes and caveats apply:

- NIST allows RSA encryption/decryption for key wrapping only. Key establishment strength 112 or 128 bits.

1.3.3 Non-Approved security functions

CryptoCell-712 also supports non-FIPS-approved security functions and modes, as specified in Table 4. The following notes and caveats apply:

- IVGEN RNG: see Section 1.4.1.5.1.
- Triple-DES: Supports 2-key bundles, which is no longer approved as of 2016. Enforces distinct keys. Rejects DES weak keys.
- The ECIES relies on CAVP validated components, as it consists of a key agreement step conforming to [SP800-56A] followed by a key derivation step conforming to [PKCS1] or [ANSI-X9.63]. Nevertheless the scheme itself is not FIPS-approved.

Table 4: Non-Approved security functions

Algorithm	Use	REE	TEE
AES non-approved modes — CBC-MAC [ISO/IEC-9797-1:2011]	Message authentication		✓
AES non-approved modes — XCBC-MAC [RFC3566]	Message authentication	✓	✓
AES GCM and GMAC [SP800-38D]	Encryption/decryption Message authentication	✓	
DES ECB, CBC [FIPS-46-3]	Encryption/decryption	✓	✓
Triple-DES ECB, CDC with 2-key bundles [FIPS-46-3, SP800-67]	Encryption/decryption	✓	✓
MD5 [RFC1321]	Message authentication	✓	✓
HMAC-MD5 [FIPS-198-1, RFC2104]	Message authentication	✓	✓
IVGEN RNG (Random Number Generator)	IV generation	✓	
ECIES (Elliptic Curve Integrated Encryption Scheme) [IEEE1363] Curves: p160k1, p160r1, p160r2, p192k1, p224k1, p256k1, P-192, P-224, P-256, P-384, P-521	Key agreement		✓
KDF1 and KDF2 Key Derivation Functions [ISO/IEC-18033-2]	Key derivation		✓
ECC key generation with Non-Approved sizes [FIPS-186-4] Curves: p160k1, p160r1, p160r2, p192k1, P-192	Key generation		✓
ECDSA with Non-Approved sizes [FIPS-186-4] Curves: p160k1, p160r1, p160r2, p192k1, P-192	Signature generation / verification		✓
KAS EC-DH key agreement [SP800-56A] Curves: p160k1, p160r1, p160r2, p192k1, P-192	Key agreement		✓
KAS DH (Diffie-Hellman) with Non-Approved sizes [SP800-56A] 1024 bits	Key generation Key agreement		✓
RSA with Non-Approved sizes [FIPS-186-4] 1024, 4096 bits	Key generation Signature generation / verification		✓

1.3.4 Approved security mode

The cryptographic module supports a single Approved mode of operation. On power-on, the module performs power-on self-tests as described in Section 9. On failure, the module enters the Error state; on success, the user chooses the mode of operation (Approved or Non-Approved) by passing a boolean parameter to the firmware initialization routine. The `FipsGetState` API can be used to check the mode at runtime, and returns Approved, Non-Approved, or Error state; this API is implemented in both the TEE the REE.

In the Approved mode, all approved security functions listed in Section 1.3.1 are available. The non-approved security functions listed in Section 1.3.3 can be accessed but are disallowed by policy. There is no enforcement mechanism in place, therefore it is the user's responsibility not to use the non-approved functions and services. In order to call any non-approved security functions and services (listed as Non-Approved in the service table), the user is instructed to perform a cold power-on and initialize the module in Non-Approved mode. All Approved services remain available in Non-Approved mode, and calling any Approved service temporarily puts the module into Approved mode with the exception of RSA Key Generation and ECC Key Generation (these services omit conditional tests in Non-Approved mode). It is the user's responsibility to avoid sharing user keys between the modes.

The user can only switch between Approved and Non-Approved mode through a power-on reset. All CSPs (Critical Security Parameters) stored in the module's registers are cleared on power-on reset. In the Approved mode, conditional tests are performed as described in Section 9.

1.4 Components and cryptographic boundary

1.4.1 Components

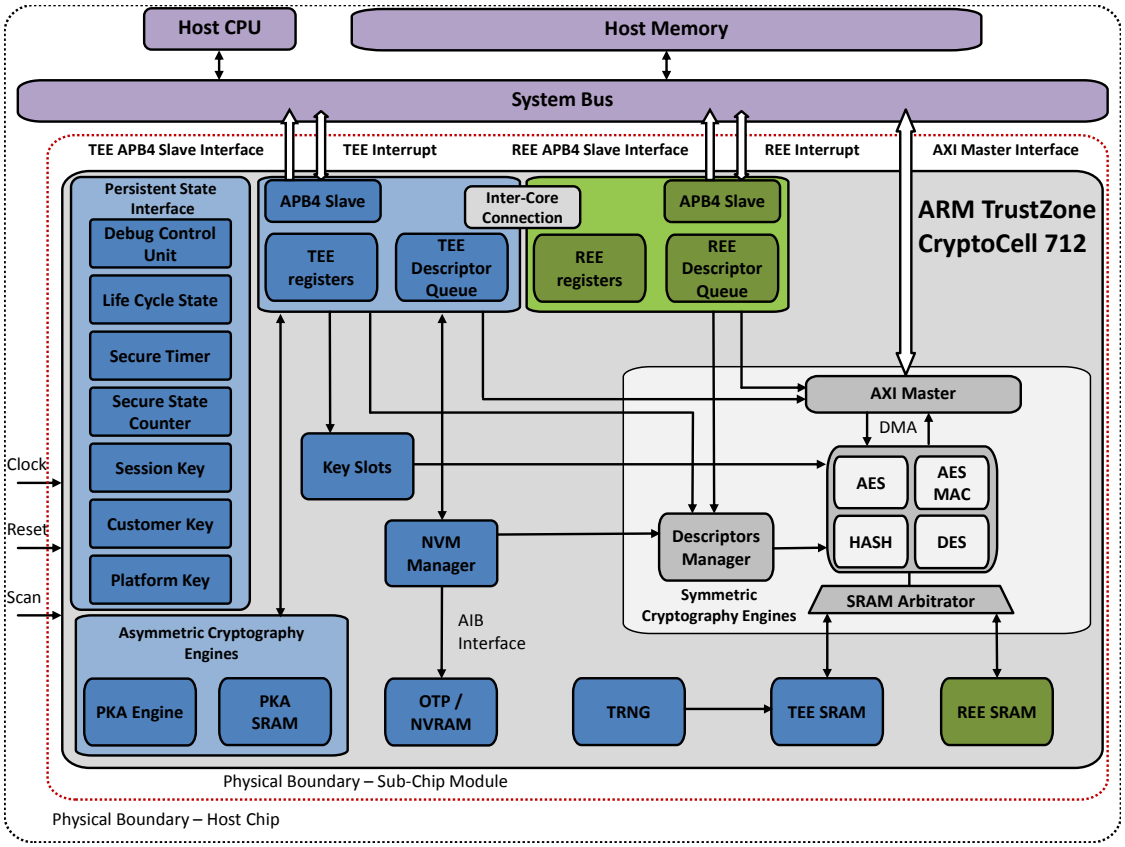

Figure 5: CryptoCell-712 hardware diagram

As described in the introduction, CryptoCell-712 consists of hardware and firmware components, some divided between the TEE and REE, and some shared. Figure 5 illustrates the hardware diagram.

All CryptoCell-712 hardware services are accessed through a firmware layer, providing a high-level interface to its functionality. The TEE is accompanied by a library exposing its functionality through a high-level API to other TEE code, and the REE is accompanied by a driver exposing a high-level interface to the REE kernel and applications.

Each environment's hardware has dedicated components for communication with the corresponding Host environment — bus connectors, register files for control and state passing, descriptor queues for task queuing and high-level control flow, interrupt and completion handling logic.

The symmetric cryptographic engines are shared between the TEE and REE. Descriptor management and memory arbitration ensure the separation of state and data between TEE and REE.

The TEE has additional components not exposed to the REE.

- An asymmetric cryptography accelerator engine, also known as a PKA (Public Key Accelerator).

- A TRNG (True Random Number Generator).
- A non-volatile memory manager in charge of OTP storage and LCS (Life Cycle State) management.
- A DRBG implemented in firmware.
- Additional security functions.

This section documents the major functional system components; for port and interface components responsible for carrying state and data signals to and from the host system, see Section 2.

1.4.1.1 Shared Hardware

1.4.1.1.1 Symmetric Cryptography Engine The symmetric cryptography engine consists of the following components:

- Cryptographic cores (AES encryption, AES message authentication, DES, MD5 and SHA hash functions, HMAC).
- A descriptor queue manager. The descriptor queue manager is responsible for high-level control flow, and allows firmware running on the Host CPU to queue multiple operations for processing by the cryptographic engines. An arbitrator ensures the correct descriptor queue is used in the each context — TEE and REE.
- Input and output DMA (Direct Memory Access) blocks. The DMA blocks allow input and output from host RAM (Random Access Memory) to dedicated TEE and REE SRAM (Static Random Access Memory) blocks.

The module provides hardware isolation for handling of platform keys. The AES cipher can directly load platform keys without exposing them to the SRAM or the Host CPU. The results of some operations on platform keys are limited to loading directly into designated hardware key registers, allowing secure key derivation.

When in the REE context, the symmetric cryptography engine can also use symmetric AES keys loaded into key slots (Section 1.4.1.1.2).

1.4.1.1.2 Hardware Key Slots Hardware key slots are dedicated hardware registers which let TEE securely create symmetric keys for REE to use. The module offers a TEE service to set the key slot values, and the REE can specify a key slot (by index) when invoking AES services. Those are the only ways to access key slots; neither environment can read key slots, and the TEE itself cannot use them for encryption.

The module defines 4 HW (Hardware) key slots for use by REE AES services. The slots be used as individual keys of size 128, 192, 256 bits or as double keys of $2*128$ or $2*256$ bits. In the latter case, two slots are used in each invocation. This functionality serves AES XTS, ESSIV and BitLocker modes.

1.4.1.1.3 Inter-Core Connection Since the module's TEE and REE components of the module have separate state and memories, CryptoCell-712 contains a dedicated mechanism for passing messages between the environments (here, referred to as cores). Consisting of a register and an interrupt signal, the mechanism allows each core to notify the other and pass basic information. This mechanism is used to synchronize the FIPS 140-2 state (Self-Test, Approved, Non-Approved, Error and error code if any) between TEE and REE.

1.4.1.2 TEE Hardware

1.4.1.2.1 Asymmetric Cryptography Accelerator (PKA) The Asymmetric Cryptography Accelerator or PKA (Public Key Accelerator) block operates as a large integer arithmetic logic unit. It supports all mathematical and logical operations required for implementing public-key cryptosystems based on the discrete-logarithm problem, the integer factorization problem, and the prime-field elliptic-curve discrete logarithm problem.

1.4.1.2.2 Non-Volatile Memory Manager The Non-Volatile Memory (NVM) Manager uses an internal, point-to-point Intel AIB (Intel Asynchronous Interface Bus Specification) interface to access a bank of OTP memory. For the contents of OTP memory, see Section 1.4.1.2.3. The NVM Manager processes the compactly-stored data in the OTP memory, and provides a high-level interface for the rest of the module.

Several CSPs such as platform keys and their hash values are stored in the OTP memory (see Section 7.2). The NVM Manager controls all access to these CSPs, protecting against unauthorized reading and modification.

1.4.1.2.3 One-Time Programmable Memory (OTP) The module is meant to use OTP to provide some security features. When synthesized in a partner's system-on-chip, the module uses an on-chip OTP bank based on eFuse or similar technology, depending on the manufacturer.

The FPGA board used for this validation can only simulate the write-once properties of real silicon OTP. Instead, the FPGA's Flash-based file system uses access permissions, allowing only administrative users to overwrite the contents of the file system. This enables behaviors not possible in the real silicon module, such as returning the module to normal operation from a terminal state. This functionality is used for FIPS 140-2 testing.

OTP bits can be written but cannot be erased, which has useful security properties. The OTP stores a counter of zero-bits alongside fields intended to be written exactly once, so that any modification to the field or its zero-bit counter invalidates them. This protection method is used to store non-modifiable values, such as keys and key hashes. The OTP also stores modifiable but monotonically changing fields: firmware version counters and LCS bitmaps. Any changes to those are made non-reversible by the properties of the OTP.

The OTP is used by the TEE to store data and control inputs used for various platform security functions, including platform keys and their hash values (Section 7.2), LCS, and TRNG configuration. The OTP memory is initialized at manufacturing time, and from that point on is hardware-limited to exclusive access through the Non-Volatile Memory Manager (Section 1.4.1.2.2).

1.4.1.2.4 True Random Number Generator (TRNG) The TEE contains a TRNG, which cannot be accessed directly as a service, but can only be used as a source of entropy for seeding the DRBG of Section 1.4.1.4.5. The TRNG collects noise directly from hardware circuits without Host interaction, by sampling the output of a fast free-running ring oscillator.

1.4.1.2.5 Persistent State Interface The Persistent State Interface stores some of the TEE state in the always-on power domain, which survives a warm reset or module sleep (see Section 2.2.3) but not cold power-on. In the module submitted for validation, this is an internal component; however, a host platform can add logic to consume its signals as inputs for other components inside the host's boundary. The Persistent State Interface is shown in Figure 5.

The Persistent State Interface contains the following components:

- LCS (Life Cycle State), as defined in Section 4. Firmware components access the LCS through a dedicated firmware service.
- The DCU (Debug Control Unit) register, driven by the Secure Debug mechanism (Section 1.4.1.4.7). A host platform can use it to control its hardware debug capabilities.
- State Counter: a 32-bit counter, used by the RAM Backup and Restore service (Section 1.4.1.4.4).
- The Session Key K_{SESS}, used by the RAM Backup and Restore service (Section 1.4.1.4.4).
- Secure Timer: see Section 1.4.1.2.6. Firmware components access the Secure Timer through a dedicated firmware service.

1.4.1.2.6 Secure Timer The TEE offers a secure timer, accessible through a firmware service.

1.4.1.2.7 Dedicated SRAM TEE has a block of dedicated SRAM (Static Random Access Memory). This is primarily used by the PKA, the symmetric cryptography engines, and the TRNG entropy collector.

The section of SRAM used by the PKA is cleared on completion of every operation. It is additionally cleared on boot, in the Deployed and RMA (Return Merchandising Authorization) Life Cycle States (see Section 4). This ensures that any Security Critical Parameters and intermediate values used in computations do not pass between different Life Cycle States and different roles.

1.4.1.3 REE Hardware

1.4.1.3.1 Dedicated SRAM REE has a block dedicated SRAM (Static Random Access Memory) for its descriptor queue and other needs. The descriptor queue size is configurable by partners for improved performance or reduced size.

1.4.1.4 TEE Firmware Figure 6 depicts the firmware components associated with the TEE. These are divided between the Secure Boot ROM (Read-Only Memory) library, the CRYS (Cryptographic Software Library), various runtime utility components, and the Hardware and Platform abstraction layers. The different functional components are described in detail below.

1.4.1.4.1 ROM Library The ROM library provides critical security functions used during the initialization stages of the module and the host platform, including Secure Boot and Secure Debug certificate verification functionality. For the purposes of this validation, the library is implemented in a read-only file system rather than an actual ROM, but partners can choose an unmodifiable ROM implementation to improve trust in the initialization process.

1.4.1.4.2 Cryptography Software (CRYS) The major TEE firmware component is the CRYS (Cryptographic Software Library), which in this module is implemented in firmware but may have software implementations. It offers cryptography functions which drive the module's hardware cryptography engines, and perform additional operations in firmware where needed. The CRYS APIs provide access to the symmetric and asymmetric cryptography engines described above in the hardware section, and the DRBG (Section 1.4.1.4.5).

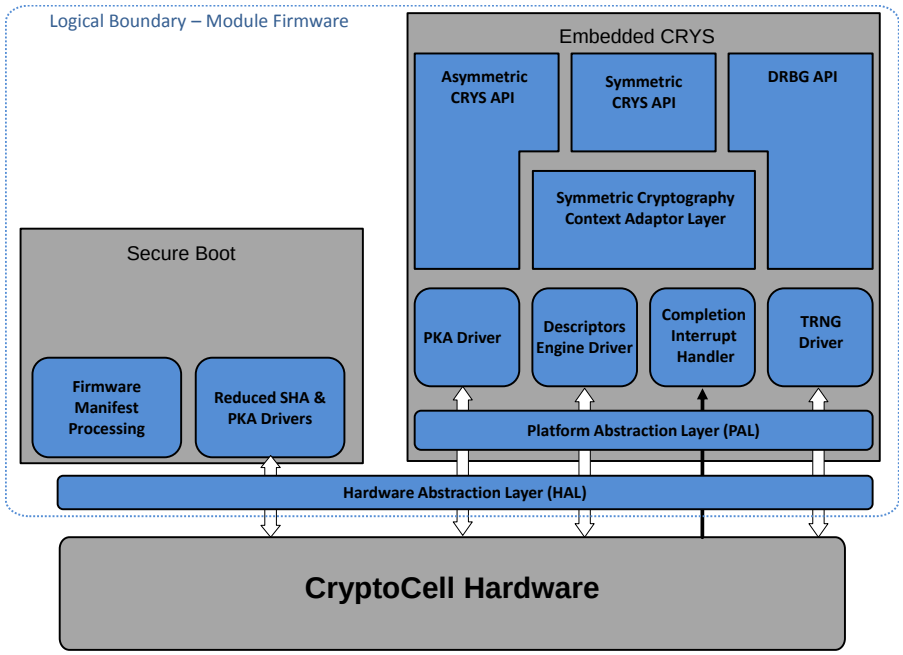

Figure 6: TEE firmware components

1.4.1.4.3 Runtime Utility Functions The utility library provides high-level functions for key management, timestamp management, RPMB (Replay Protected Memory Block) and LCS Management. It also includes the RAM Backup and Restore service below.

1.4.1.4.4 RAM Backup and Restore The RAM Backup and Restore service allows the operator to save and load external (host) RAM state securely, in order to save the state over periods of sleep. The service derives a key using KBKDF (Key Based Key Derivation Function) from the Session Key K_{SESS}, and uses AES-128 CCM authenticated encryption to encrypt and authenticate the RAM contents passed to it on entry to sleep, and validate and decrypt the contents on resumption.

In addition, the service uses a persistent 32-bit State Counter, kept in a dedicated hardware register in the Persistent State Interface (Section 1.4.1.2.5), to prevent rollback. The counter is cleared on power-on reset and incremented on every invocation of the backup service. The counter is included in the AES CCM authenticated encryption operation, as part of the nonce input. The restore operation fails if the counter does not match.

1.4.1.4.5 Deterministic Random Bit Generator (DRBG) The DRBG is a [SP800-90A] CTR_DRBG. The module creates a DRBG instance on boot with AES-256 as the underlying cipher, and uses it for all internal random number generation. The user can create additional instances of the DRBG. A DRBG instance holds its values `V` and `Key`.

The DRBG instance is seeded with 384 bits from the TEE TRNG (see Section 1.4.1.2.4), including 256 for the seed buffer and 128 for the personalization string.

1.4.1.4.6 Secure Boot The CryptoCell-712 Secure Boot component provides integrity services for CryptoCell-712 firmware and optionally other Host firmware. Secure Boot is performed during the power-on boot process, and validates the various components of the firmware image based

on a firmware manifest and a set of certificates stored in NVRAM. Its chain of trust is rooted in a public key hash $\mathtt{H}_{BK}$, stored in OTP during manufacturing. Certificate and image verification uses PKCS#1v2.1 RSA-PSS-2048 with SHA-256 hashes.

Secure Boot supports the following features:

- Chaining of secondary certificates, to permit parts of the system image to be signed by other trusted developers
- Software confidentiality — optional firmware decryption using AES-128 CTR with the Code Encryption Key ($\mathtt{K}_{CE}$)
- Support for firmware updates: an external mechanism can update the firmware image, manifest and certificates
- Image version revocation

Arm® provides tools for system and software providers to support certificate management, device provisioning, and the firmware image signing.

A security failure during Secure Boot, such as an invalid certificate, version mismatch, or signature mismatch, will cause boot failure and the module will enter the Error state.

1.4.1.4.7 Secure Debug The CryptoCell-712 Secure Debug mechanism is a security service provided by the module to the Host. The mechanism performs a boot-time security check on a debug certificate located in the NVRAM, and writes to the module's DCU (Debug Control Unit) register. On the FPGA submitted for this validation, the DCU has no further effect. Actual CryptoCell partners typically use the DCU to control the host's debugging capabilities, which provide authorized parties with extended access to the host CPU and RAM contents.

Secure Debug does not constitute a FIPS Maintenance mode, as it does not provide any extended access to the module itself. Secure Debug is also not managed as a Life Cycle State, since it is reversible. Since Secure Debug is associated with the manufacturer, it is assigned to the Crypto Officer.

In addition to validating a multi-tiered certificate chain on entry, the module takes additional steps to prevent access to Platform Keys during the Debug mode, preventing the leakage or misuse of any CSPs stored within, even by the original manufacturer.

Secure Debug certificates are device-specific. The certificate must contain a 256-bit `SOC ID` field, authenticated with the issuer's signature, which the module validates. The `SOC ID` must match the one produced by the module's Identify SOC service.

The Secure Debug certificate checking mechanism is also used to enter the RMA LCS. The certificates for RMA and for Secure Debug are identical, except for a flag indicating which mode the certificate allows to enter.

1.4.1.4.8 Abstraction layers The CryptoCell-712 defines several internal APIs in the HAL (Host Adaptation Layer) and PAL (Platform Adaptation Layer) firmware components in the TEE runtime library.

The HAL provides a firmware abstraction for the module's basic hardware capabilities, including interrupt and cache control, initialization and termination.

The PAL (Platform Adaptation Layer) provides a firmware abstraction to the module's interface with the Host OS, and includes functions for initialization and termination, DMA control, memory allocation, memory mapping and basic memory manipulation functions such as `memcopy`, and synchronization primitives such as mutexes and barriers. The implementation provided by Arm® is functional and adapted to the reference implementation submitted for

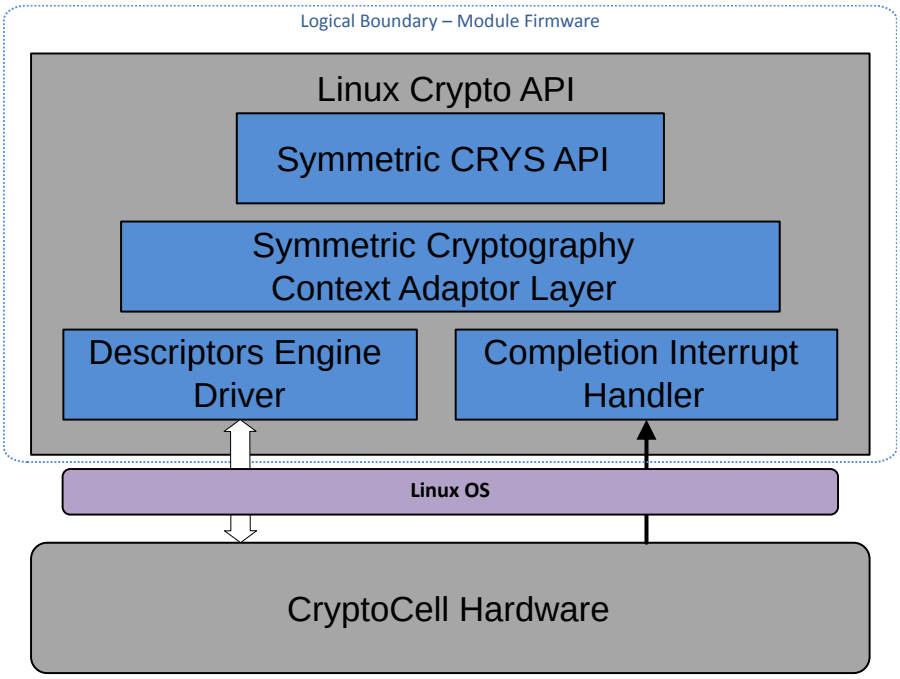

Figure 7: REE firmware components

validation. The partner is expected to replace some HAL (Host Adaptation Layer) and most PAL function implementations, as needed to adapt them to the Host OS of their choice. The partner should refer to the module's Integration Guides for details on adapting those layers.

1.4.1.5 REE Firmware

Figure 7 depicts the firmware components associated with the REE. The firmware exposes the CRYS functionality in a high-level form, accessible to the host OS.

The firmware is a standard Linux Kernel driver, which registers with the OS as a platform device driver. The driver registers the services it provides as Linux kernel crypto algorithms, as detailed in Section 2.4.1.

The driver interfaces allow the OS kernel and application code to invoke the REE crypto functionality, while using caller-supplied keys or key slots (Section 1.4.1.1.2).

1.4.1.5.1 IV Generator

Linux cryptographic interfaces support an optional invocation mode in which the IV (Initialization Vector) is generated internally, rather than passed in as a parameter by the caller. For this purpose, the REE firmware contains a non-approved RNG 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 random number generation service by the module; it is only available for IV generation in REE algorithms. This RNG is non-approved because it does not implement the CRNGT (Continuous Random Number Generator Test). The operator is instructed not to use IVGEN in the FIPS-Approved Mode.

1.4.2 Cryptographic boundary

The module's hardware is integrated into the Host silicon, the module's firmware is stored in the host's NVRAM, and the module firmware is executed on the Host CPU (see Section 1.1). All the module components are contained in the Host SOC package. Since the module firmware runs on the general-purpose Host CPU, the module is defined as a sub-chip firmware-hybrid module, and has four cryptographic boundaries, as defined below and shown in Figure 4.

The sub-chip subsystem boundary is defined as the set of circuitry specified by the module's RTL and terminating at the module's hardware interfaces, listed in Section 2, along with the module's firmware components specified by the module's binaries. The sub-chip hardware physical boundary is defined as the single-chip physical boundary, corresponding to the Host SOC in which the module is integrated.

The physical boundary of the firmware is the platform on which the firmware and OS reside, in this case the Host SOC which includes the memories where the firmware is stored and the CPU on which it is executed. The logical boundary of the firmware is the set of firmware components that implement the module's cryptographic and related functionality, and is wholly contained within the physical boundary.

2 Ports and Interfaces

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

Table 5: Ports and interfaces

Type	Interfaces	TEE	REE
Power Input	Power	✓	✓
Control Input	Clock	✓	✓
	Reset	✓	✓
	Scan	✓	✓
	APB (AMBA Peripheral Bus)	✓	✓
	Firmware API calls	✓	✓
Status Output	Interrupt	✓	✓
	Firmware API return values	✓	✓
Data Input	AXI (Advanced eXtensible Interface)	✓	✓
	Firmware API input arguments	✓	✓
Data Output	AXI	✓	✓
	Firmware API output arguments	✓	✓

2.1 TEE and REE Hardware Interfaces

The following interfaces are duplicated, and present in both TEE and REE.

2.1.1 APB Slave

The two APB (AMBA Peripheral Bus) slaves are used to control the TEE and REE operation. The Host accesses each APB slave as a memory-mapped input/output device. The interface provides access to the module's control registers, and serves to pass descriptors and trigger hardware operations.

2.1.2 Interrupt

Both REE and TEE have an interrupt signal to attract a Host's attention on operation completion or error. Information on the cause of the interrupt is available through dedicated memory-mapped registers.

2.2 Shared Hardware Interfaces

2.2.1 AXI Master

The AXI (Advanced eXtensible Interface) Master is used to pass data between the host memories and the symmetric cryptographic engine's memories. It implements the AXI protocol with

ACE-Lite extensions [AMBA].

2.2.2 Clocks

The module has multiple internal clock domains, with a separate clock domain for each cryptographic engine. All internal clocks are derived from a single clock signal supplied from the host system.

2.2.3 Power

The REE and TEE receive power from the host. When not in use (e.g. during host sleep), the host may power down the REE and TEE hardware, and later turn the hardware back on when needed. The module keeps internal state that needs to be preserved during warm power down in the always-on Persistent State Interface (Section 1.4.1.2.5). The module's firmware, running on the Host CPU, contains logic for orderly suspension and resumption of module operations on power down.

2.2.4 Reset

The module supports a reset signal, which clears all hardware state and registers. This does not affect the module's firmware state.

2.2.5 Scan Interface

The module supports a scan input signal. This signal drives the module's reset line, clearing keys stored in hardware registers. In the TEE, asserting the scan line has the additional effect of making platform keys inaccessible (see Section 7.2.1).

Partners commonly implement scan logic capabilities, allowing to test and extract the contents of memories and registers during manufacturing. The module submitted for the purposes of this validation does not contain such scan logic, however the scan interface security features are present.

2.3 TEE Firmware

The TEE firmware exposes its services through library APIs functions. See Table 6 for the full list of services. Documentation for the API functions implementing those services is available in the full proprietary documentation package.

2.4 REE Firmware

2.4.1 Linux Kernel Driver services

The REE Firmware is a standard Linux Kernel platform device driver. The driver registers itself as a Linux kernel cryptographic algorithms provider, and offers cryptographic services to applications and kernel components. The module's services are registered under a set of string identifiers, which are listed in the full proprietary documentation package.

The REE AES algorithms can use key slots keys, specifying them by index, as well as user keys.

2.4.2 Status service

Other than the Linux driver services, the REE firmware offers one service API:

- `FipsGetState`: FIPS Status API, returns the mode (Approved, Non-Approved, Error) and error code if any.

3 Roles, Services and Authentication

3.1 Roles

CryptoCell-712 offers high-level cryptographic services which operate on CSPs and user inputs, as well as platform security services which can be used by the Host platform to establish a root of trust. In addition, the module defines separate states for manufacturing and manufacturer-authorized recovery and debug.

The following two roles are defined:

User Role
The user is defined as the set of firmware applications running in the TEE and REE, when the module is in Deployed state. This role accesses cryptographic services, including Approved and non-Approved security functions, 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 NVRAM 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. States designated for the Crypto Officer are Chip Manufacturing, Device Manufacturing, RMA, and Security Disabled. The Secure Debug service (see Section 4) is designated to the Crypto Officer.

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

3.2 Services

Table 6 below lists the services offered by CryptoCell-712, along with a concise description of purpose, the list of security functions offered or used by the service, a list of keys in use and the access types for each, the role that can access that service, the Approved or Non-Approved status, whether it's supported in TEE or REE, and the operator's access rights to any keys and CSPs involved.

Note that when a service is implemented by both REE and TEE, the same hardware is being used, with different firmware components passing the arguments and triggering the operations.

Access to keys is denoted with a single letter:

- I — input keys from the user,
- O — output keys to the user,
- R — read keys from internal storage,
- W — write keys to internal storage,
- Z — zeroize.

Access type is defined by the perspective of the module or the service. Input and read access also includes any internal usage for the read key.

Table 6: Services

Service Name	Purpose	Security Functions	Keys and CSPs	Key Access	User Role	CO Role	Approved	REE	TEE
AES — TEE Approved	Encryption Decryption	AES-128, 192, 256 ECB, CBC, CTR, OFB, CMAC, XTS, CTS (CBC-CS1)	**Input:** User keys K_{CST} K_{DR} K_{SESS} K_{PLT}	 - I - R - R - R - R[1]	✓		✓		✓
AES — TEE Non-Approved modes	Encryption Decryption	AES-128, 192, 256 XCBC-MAC	**Input:** User keys K_{CST} K_{DR} K_{SESS} K_{PLT}	 - I - R - R - R - R	✓				✓
AES — REE Approved	Encryption Decryption	AES-128, 192, 256 ECB, CBC, CTR, OFB, CMAC, XTS, CTS (CBC-CS1), ESSIV, BitLocker	**Input:** User keys Key slots	 - I - R	✓		✓	✓	
AES — REE Non-Approved modes	Message authentication	AES-128, 192, 256 modes: XCBC-MAC, GCM, GMAC	**Input:** User keys Key slots	 - I - R	✓			✓	
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	✓				✓
DES Non-Approved	Encryption Decryption	DES ECB, CBC	**Input:** User keys	 - I	✓				✓
SHA Approved	Message authentication	SHA-1[2], SHA-224, SHA-256, SHA-384, SHA-512			✓		✓	✓	✓
MD5 Non-Approved	Legacy message authentication	MD5			✓			✓	✓
HMAC Approved functions	Message authentication	HMAC SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	**Input:** User keys	 - I	✓		✓	✓	✓
HMAC-MD5 Non-Approved	Legacy message authentication	HMAC-MD5	**Input:** User keys	 - I	✓			✓	✓

[1] K_{DR} is limited by firmware to use in key derivation only. K_{SESS} and K_{PLT} are internally used by the module for encryption/decryption, and the operator is instructed to use those keys only for encryption or decryption, to conform to [SP800-57] Section 5.2 “Key Usage”.

[2] SHA-1 is not approved for signature generation, Approved for legacy signature verification and other uses.

Service Name	Purpose	Security Functions	Keys and CSPs	Key Access	User Role	CO Role	Approved	REE	TEE
NIST SP800-108 Key Derivation Function Approved	KBKDF (Key Based Key Derivation Function) [SP800-108]	KDF in Counter Mode. AES-128 or AES-256 in CMAC mode	**Input:** User keys K_{DR} K_{PLT} K_{OEM} **Output:** User keys K_{SESS} K_{PLT} K_{OEM}	 - I - R - R - I - O - W - W - O	✓		✓		✓
NIST SP800-135 Key Derivation Functions Approved	Key derivation [SP800-135][3].	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	**Input:** User keys **Output:** User keys	 - I - O	✓		✓		✓
KDF1 and KDF2 Key Derivation Functions Non-Approved	Key derivation functions KDF1 and KDF2 [ISO/IEC-18033-2]	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	**Input:** User keys **Output:** User keys	 - I - O	✓				✓
Endorsement Key Derivation	Derive the Endorsement Key ECC key-pair	ECC key pair generation[4]	**Input:** K_{DR} **Output:** $PubK_S$ $PrivK_S$	 - R - O - O	✓		✓		✓
RPMB (Replay Protected Memory Block) Key Derivation	Derive K_{RPMB} from K_{DR} using KBKDF [SP800-108]	KBKDF	**Input:** K_{DR} **Output:** K_{RPMB}	 - R - O	✓		✓		✓
RPMB Page MAC Generation	RPMB Frame authentication [JESD84]	HMAC SHA-256	**Input:** K_{RPMB}	 - I	✓		✓		✓
ECC (Elliptic Curve Cryptography) Key Generation Approved	ECC key generation [FIPS-186-4]	**Curves:** P-224, P-256, P-384, P-521, p224k1, p256k1.[5] Uses DRBG.	**Output:** User keys **Sizes:** 224, 256, 384, 521 bits	 - O	✓		✓		✓
ECC (Elliptic Curve Cryptography) Key Generation Non-Approved strengths	ECC key generation [FIPS-186-4]	**Curves:** P-192, p160k1, p160r1, p160r2, p192k1, [6] user-defined domains. Uses DRBG.	**Output:** User keys **Sizes:** 160, 192 bits	 - O	✓				✓
ECC Public Key Validation	Public key validation		**Input:** User keys	 - I	✓		✓		✓

[3]This includes key derivation methods ASN1DER and Concatenation specified in [ANSI-X9.42, 7.7.1] and [ANSI-X9.63, 5.6.3], approved when used as part of a [SP800-56A] agreement scheme

[4]The private key is derived according to [FIPS-186-4, Section B.4.1], but instead of a random input from a RBG (Random Bit Generator), the procedure uses the result of a [SP800-108] KBKDF applied to the K_{DR}. The resulting key pair has a different value per module, permanent until zeroization.

[5]Security strengths for non-NIST-recommended curves: p224k1 — 112 bits, p256k1 — 128 bits.

[6]Security strengths for non-NIST-recommended curves: p160k1, p160r1, p160r2 — 80 bits, p192k1 — 96 bits.

Service Name	Purpose	Security Functions	Keys and CSPs / Key Access	User Role	CO Role	Approved	REE	TEE
ECDSA (Elliptic Curve Digital Signature Scheme) Approved	Signing Verification [FIPS-186-4]	SHA functions	**Input:** User keys - I **Sizes:** 224, 256, 384, 521 bits	✓		✓		✓
ECDSA Non-Approved strengths	Signing Verification [FIPS-186-4]	SHA functions Non-approved when using non-approved key sizes and domains.	**Input:** User keys - I **Sizes:** 160, 192 bits; user-defined domains.	✓				✓
EC-DH (Elliptic Curve Diffie Hellman) Approved	Key Agreement [SP800-56A]	EC-DH	**Input:** User keys - I **Output:** User keys - O **Sizes:** 224, 256, 384, 521 bits	✓		✓		✓
EC-DH (Elliptic Curve Diffie Hellman) Non-Approved strengths	Key Agreement [SP800-56A]	EC-DH, when using non-approved key sizes and domains.	**Input:** User keys - I **Output:** User keys - O **Sizes:** 160, 192 bits; user-defined domains.	✓				✓
ECIES (Elliptic Curve Integrated Encryption Scheme) Non-Approved	Key Transport [ISO/IEC-18033-2]	ECIES-KEM [ANSI-X9.63] KDF with SHA functions 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) [FIPS-186-4]	Uses DRBG	**Output:** User keys - O **Sizes:** 2048, 3072 bit	✓		✓		✓
RSA Key Generation Non-Approved strengths	Key pair generation (regular and CRT) [FIPS-186-4]	Uses DRBG	**Output:** User keys - O **Sizes:** 1024, 4096 bit	✓				✓
RSA Signing and Verification Approved	PSS signing and verification [FIPS-186-4]	RSA SHA-1 (legacy verification only) SHA-224, SHA-256, SHA-384, SHA-512 DRBG	**Input:** User keys - I **Sizes:** 2048, 3072; 1024 (legacy verification only)	✓		✓		✓
RSA Signing and Verification Non-Approved sizes or functions	PSS and RSAES-OAEP signing and verification [FIPS-186-4, PKCS1]	RSA SHA functions PKCS1.5 with MD5 DRBG	**Input:** User keys - I **Sizes:** 1024, 4096	✓				✓
RSA Key Wrapping Allowed	RSAES-PKCS1-v1.5 and RSAES-OAEP encryption and decryption [PKCS1]	RSA SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 DRBG	**Input:** User keys - I **Sizes:** 2048, 3072	✓				✓

Service Name	Purpose	Security Functions	Keys and CSPs	Key Access	User Role	CO Role	Approved	REE	TEE
RSA Encryption and Decryption Non-Approved	RSAES-PKCS1-v1.5 and RSAES-OAEP encryption and decryption [FIPS-186-4, PKCS1] with non-approved sizes	RSA SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 PKCS1.5 with MD5 DRBG	**Input:** User keys **Sizes:** 1024, 4096	- I	✓				✓
Diffie-Hellman Key Generation Approved	Key Generation Domain Parameter Generation [SP800-56A]	Uses DRBG	**Output:** User keys **Sizes:** 2048 bits	- O	✓		✓		✓
Diffie-Hellman Key Generation Non-Approved strengths	Key Generation Domain Parameter Generation [SP800-56A]	Uses DRBG	**Output:** User keys **Sizes:** 1024 bits	- O	✓				✓
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 **Output:** User keys **Sizes:** 2048	- I - O	✓		✓		✓
Diffie-Hellman Key Exchange Non-Approved strengths	Key Agreement [SP800-56A]	SHA functions	**Input:** User keys **Output:** User keys **Sizes:** 1024	- I - O	✓				✓
DRBG (Deterministic Random Bit Generator) Instantiation	DRBG Context Instantiation	Creates a DRBG context structure, using TRNG internally for seeding.	**Output:** DRBG 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 KAT (Known Answer Test)[7]			✓		✓		✓
Set Key Slot	Set key from TEE to be used in REE AES algorithms		**Input:** User keys **Output:** Key slots	- I - W	✓		✓		✓

[7]The operator is instructed not to call those functions at run time, they are intended for power-on testing only.

Service Name	Purpose	Security Functions	Keys and CSPs — Key Access	User Role	CO Role	Approved	REE	TEE
LCS (Life Cycle State) Read	Firmware interface to the LCS			✓		✓		✓
RMA (Return Merchandising Authorization)	Change LCS to RMA, zeroize module	Uses RSA-PSS and SHA-256 to verify a certificate	**Input:** H_{BK} - R RSA public key in RMA certificate - I **Sizes:** 2048 bit **Zeroize:** K_{CE} - Z K_{DR} - Z		✓	✓		✓
Secure Debug	Enter Secure Debug state	Uses RSA-PSS and SHA-256 to verify a certificate	**Input:** H_{BK} - R RSA public key in debug certificate - I **Sizes:** 2048 bit		✓	✓		✓
Non-Volatile Memory Manager	Monotonic Counter Get / Set			✓		✓		✓
TEE Library Initialization	Performs self-tests. Normally only called once on cold power-on.			✓		✓		✓
Secure Timer	Get Timestamp Compare Timestamp			✓		✓		✓
Secure Boot — Read Configuration	Read Certificate Public Key Hash		**Input / Output:** H_{BK} - R, 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:** H_{BK} - R RSA public key in boot certificate - I **Sizes:** 2048	✓		✓		✓
Identify SOC	Return `SOC ID`, unique device ID derived from an H_{BK} and K_{DR}, used in Secure Boot and Secure Debug.	Uses KBKDF [SP800-108] and SHA-256	**Input:** K_{DR} - R H_{BK} - R **Output:** `SOC ID` - O	✓		✓		✓
Session Key Generation	Derive K_{SESS} from K_{DR} and DRBG output.	Uses KBKDF [SP800-108] Uses DRBG	**Input:** K_{DR} - R **Output:** K_{SESS} - W	✓		✓		✓
External RAM Backup and Restore	Backup and restore external RAM buffers securely	Uses AES-128 CCM for encryption and authentication	**Input:** K_{SESS} - R	✓		✓		✓
OEM (Original Equipment Manufacturer) Asset Provisioning Get Provisioning Key [8]	Derive K_{OEM} from K_{PLT} and H_{BK}	Uses KBKDF [SP800-108]	**Input:** K_{PLT} - R H_{BK} - R **Output:** K_{OEM} - O	✓				✓

[8] Disabled in FIPS-compliant modules.

Service Name	Purpose	Security Functions	Keys and CSPs	Key Access	User Role	CO Role	Approved	REE	TEE
OEM Asset Provisioning Asset Unpacking	Derive an asset key from K_{OEM} and Asset ID, authenticate and decrypt an asset package	Uses KBKDF [SP800-108] AES-128 CCM to decrypt asset	**Input:** K_{OEM}	- I	✓				✓
Show Status (FipsGetState)	Indicates the FIPS status: Approved, Non-Approved, or Error state (includes error code)				✓		✓	✓	✓

3.3 Authentication

Role authentication in CryptoCell-712 is implicit. The Crypto Officer Role is defined by access to the initialization and recovery services. Specific Life Cycle States are designated for the Crypto Officer role. At manufacturing time, the module assigns the operator to the Crypto Officer role until its NVRAM is initialized, putting the module into the Deployed LCS. At that point, the module assigns 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 states intended for recovery (Secure Debug and RMA), which are again assigned to the Crypto Officer role.

4 Finite State Model

CryptoCell-712 life cycle and operational states follow a FSM (Finite State Model), fully described in the separate FSM document. The module LCS (Life Cycle State) is tied to user roles and affect the module functionality, a summary is given below.

4.1 Deployed states

Power Off
The module do not receive power and it is completely off. No functionality available until power-on.

Cold Power-on
Following cold power-on, the module's Non-Volatile Memory Manager (Section 1.4.1.2.2) determines the LCS from a dedicated OTP field, and goes to Deployed, or one of the Manufacturing and recovery states below. If the OTP indicates Deployed but the module NVRAM contains a valid debug certificate in a designated area, the module enters Secure Debug.

Deployed
A module in the field normally starts in the Deployed LCS. The module initializes and goes to the Self-Test state. Module services can be invoked to enter RMA or Secure Debug states, but this will only take effect on the next cold power-on.

Self-Test
Performs FIPS power-on tests. If those fail, goes to the Error state. Otherwise, goes to either Approved or Non-Approved state according to input from the Host platform.

Error
The FIPS error state is entered from the Self-Test state or Approved Mode, if a power-on or conditional test fails. All data output is blocked. On power-on reset, the module returns to Deployed state.

Approved Mode
FIPS-Approved mode of operation. FIPS-approved functions are available for use. The module may enter Error state if a conditional test fails. On power-on reset the module returns to Deployed state.

Non-Approved Mode
Non-FIPS-approved mode of operation. All functions are available for use. On power-on reset the module returns to Deployed state.

Secure Debug
This state is non-terminal and non-permanent, and is not recorded in the OTP. It is associated with the Crypto Officer. The device will enter this state on cold power-on if a valid certificate is present in a dedicated location in NVRAM. On power-on reset the module returns to Deployed state, if the certificate is removed. See Section 1.4.1.4.7 for details.

4.2 Manufacturing and recovery states

Chip Manufacturing
The initial LCS. At manufacturing, the ICV (Independent Chip Vendor) uses the CMPU (Chip Manufacturing Production Utility) provided by Arm® to securely initialize the K_{DR}. On completion, the module transitions to Device Manufacturing on the next cold

power-on.

Device Manufacturing

The OEM initializes all remaining security assets in module NVRAM. On completion, the module transitions to Deployed on the next cold power-on. The manufacturer may instead put the module into the Security Disabled state.

Security Disabled

The manufacturer may opt to block the security functionality of some modules. This state is terminal (the module will return to it on each cold power-on). Secure Boot and all cryptography functions are disabled in this state.

RMA

The RMA (Return Merchandising Authorization) state is for devices that return to the manufacturer for failure analysis. This state is entered by placing a manufacturer certificate in NVRAM, and then using the RMA service and a cold power-on. Modules in the RMA state zeroize and block access to platform keys (see Section 7.2.1), but retain full cryptographic functionality[9]. This state is terminal (the module will return to it on each cold power-on).

[9]Modules with encrypted firmware will halt with error during boot, since the firmware encryption key is zeroized.

5 Physical Security

CryptoCell-712 is a sub-chip module provided as set of RTL and firmware sources. The module shall be synthesized in a single host chip with standard passivation and a production grade enclosure that prevents access to the interior of the module and conforms to level 1 requirements for physical security.

6 Operational Environment

The module's operational environment is non-modifiable. The module does not contain a mechanism to update its firmware. In addition, the module uses integrity techniques (Section 9.1.2) to enforce the non-modification of its firmware.
The operational environment for CryptoCell-712, as implemented for this validation, is the Linux OS kernel version 3.18.
The module's TEE firmware is a static library providing cryptographic services for the OS through API function calls. The TEE firmware's interface to the hardware is direct, done using memory and register operations and interrupts; the module does not use OS services to access its hardware, and therefore has no functional dependency on the OS. Any variation in OS mechanisms, such as memory allocators and synchronization primitives, is accounted for by the PAL (Platform Adaptation Layer) (Section 1.4.1.4.8).

7 Cryptographic Key Management

CryptoCell-712 works with three types of keys: user keys, platform keys, and key slots (Section 1.4.1.1.2). Table 7 summarizes the keys defined in the module.

Table 7: Module keys and CSPs

Key	Full name	Purpose	Length	Storage	Entry and Derivation
User keys	N/A	Used with cryptographic services	Various[10]	User RAM Plaintext	Input by user or generated by the module's cryptographic services
Key slots	N/A	Set in TEE for use by REE AES algorithms	128, 192, 256, 2∗128, 2∗256 bit	HW registers Plaintext	Set by TEE
K_{DR}	Device Root Key	Derivation of device-specific keys	256 bit	OTP, HW register Plaintext	Generated by DRBG during manufacturing, with seed from TRNG
K_{RTL}	Provisioning Master Key	Derivation of K_{PLT}	128 bit	OTP Plaintext	Written in OTP during manufacturing
K_{PLT}	Platform Key	Either decryption of K_{CST} or derivation of K_{OEM}[11]	128 bit	HW register Plaintext	Derived using KBKDF from K_{RTL} and H_{BK}.
EK_{CST}	Encrypted Customer Key	Stores K_{CST} encrypted with K_{PLT}	128 bit	OTP Encrypted	Written by OEM during manufacturing
K_{CST}	Customer Key	Used as a key for customer assets	128 bit	HW register Plaintext	Decrypted on boot with AES-128 ECB from EK_{CST} with K_{PLT} as key
K_{OEM}	OEM Provisioning Key	OEM asset decryption and verification. Non-FIPS-compliant modules only.	128 bit	Host SRAM Plaintext	Derived using KBKDF from K_{PLT} and H_{BK}
$PubK_{BK0}$ $PubK_{BK1}$	Boot key 0 Boot key 1	Certificate verification for Secure Boot, Secure Debug and RMA	1∗2048 bit or 2∗2048 bit	NVRAM (Flash) Plaintext	One or two boot keys, written by OEM during manufacturing
H_{BK}	Hash of Boot Key	Boot key integrity protection (SHA-256 hash of one of the boot keys)[12]	1∗256 bit or 2∗128 bit	OTP Plaintext	Written by OEM during manufacturing
K_{CE}	Firmware Code Encryption key	Optional: firmware image decryption as part of Secure Boot	128 bit	OTP and SRAM Plaintext	Written by OEM during manufacturing
K_{SESS}	Session key	Used in the RAM Backup service	128 bit	HW register Plaintext	Derived using KBKDF from K_{DR} and a 96-bit random value
$PubK_S$ $PrivK_S$	Endorsement key pair	ECDSA signing of device messages	256 bit	Host SRAM Plaintext	Derived according to [FIPS-186-4, B.4.1] [13]
K_{RPMB}	RPMB shared key	Computing HMACs for RPMB data frames	256 bit	Host SRAM Plaintext	Derived from K_{DR} using KBKDF

[10] See Table 8 for key lengths used.

[11] To comply with [SP800-57, 5.2], K_{CST} must be limited to one usage type. FIPS-compliant modules only use K_{PLT} to decrypt K_{CST}. Non-compliant modules derive K_{OEM} and offer the OEM Asset Provisioning service instead. The configuration is set in permanent NVRAM per module.

[12] H_{BK} may instead hold two SHA-256 hashes truncated to 128 bits, to verify both $PubK_{BK0}$ and $PubK_{BK1}$. Otherwise only one boot key can be used. The use of SHA-256 hashes truncated to 128 bits is approved according to [SP800-107, 5.1]

[13] Instead of a random input from a RBG, the procedure uses the result of KBKDF applied to the K_{DR}. The resulting key pair value is fixed per module.

7.1 User Keys

User keys are created and owned by the module's user — application firmware running on the Host CPU in the TEE and REE. 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.

Table 8: User key sizes

Algorithm	Key sizes
AES	128, 192, 256 bits
DES	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	1024, 2048, 3072, 4096 bits
Diffie-Hellman	1024, 2048 bits

7.2 Platform Keys

Platform keys are named keys with a specific purpose. Several platform keys are stored in the module's OTP. Those keys are used to derive additional platform keys during the boot sequence, or on demand. Most platform keys are stored in dedicated HW registers and are limited by the hardware to specific uses, but some of the derived keys are output to the user.
The following keys are not FIPS-compliant, and are disabled by configuration in modules passing FIPS certification:

- The provisioning key $\texttt{K}_{OEM}$ is derived and output to the user before the module completes self-tests, and therefore FIPS-certified modules are configured to disable $\texttt{K}_{OEM}$ and use the alternative provisioning key $\texttt{K}_{CST}$ instead.

7.2.1 Blocking access to Platform Keys

The module blocks access to platform keys in the RMA and Secure Debug states, to prevent Crypto Officer access to plaintext platform keys. In the RMA state, root keys are permanently zeroized. In the Secure Debug state, the registers for root keys are either masked with temporary values or cleared to zero, but the OTP value remains, so this does not constitute FIPS 140-2 zeroization. Other keys derived from root keys are also affected. The exact effects are detailed in Table 9.
Key slots (Section 1.4.1.1.2) are also masked. User keys are cleared by the power-on reset on entry to those modes.
In effect, all platform keys are made inaccessible, and so is any content encrypted with them prior to zeroization. Two OTP keys, $\texttt{K}_{DR}$ and $\texttt{K}_{CE}$, are erased upon entry to RMA. $\texttt{EK}_{CST}$ is stored in ciphertext and cannot be decrypted following zeroization.

Table 9: Blocking access to platform keys

Key	Secure Debug	RMA
$\mathtt{K}_{DR}$	Masked	Erased in OTP
$\mathtt{K}_{CE}$	No effect	Erased in OTP
$\mathtt{K}_{RTL}$	Masked	Erased in OTP
$\mathtt{K}_{PLT}$	Cleared	Cleared
$\mathtt{EK}_{CST}$	No effect	No effect
$\mathtt{K}_{CST}$	Cleared	Cleared
$\mathtt{K}_{OEM}$	Cleared	Cleared
$\mathtt{K}_{SESS}$	Alternative value	Alternative value
$\mathtt{K}_{RPMB}$	Alternative value	Alternative value
$\mathtt{PubK}_S$, $\mathtt{PrivK}_S$	Alternative value	Alternative value

7.3 Key generation

CryptoCell-712 uses a DRBG compliant with [SP800-90A], seeded with at least 384 bits of entropy from a TRNG. See Sections 1.4.1.2.4, 1.4.1.4.5 for details on the TRNG and DRBG implementations.

Asymmetric key generation services offered by the module are defined by the [FIPS-186-4] standard, use the DRBG service for random inputs, and are detailed in Section 3.2. No dedicated service is offered for symmetric key generation; the operator is instructed to directly use the output of the DRBG service.

Internally, the module uses the DRBG to generate the session key $\mathtt{K}_{SESS}$ (Table 7) and to provide random inputs to cryptographic services that use randomness (RSA and ECIES).

For initializing the Device Root Key $\mathtt{K}_{DR}$ during the manufacturing process, Arm® offers its partners a CMPU which executes on the module during manufacturing. The CMPU performs FIPS self-tests before using the module's TRNG to seed the DRBG and generate the $\mathtt{K}_{DR}$ internally. Alternatively, an OEM may generate the $\mathtt{K}_{DR}$ externally and burn it into the module's OTP.

7.4 Key establishment

Key establishment and key derivation services offered by the module are detailed in Section 3.2. The module offers key establishment services based on the approved Elliptic Curve Diffie-Hellman and Finite Field Diffie-Hellman algorithms [SP800-56A], and the non-approved ECIES (Elliptic Curve Integrated Encryption Scheme) [IEEE1363]. The module does not offer key establishment services based on the RSA algorithm.

The module offers approved key derivation services using Key Derivation Functions as defined in [SP800-108, SP800-135]: KBKDF, CVL ANS X9.63-2001, and ASN1DER.

In addition, Non-Approved key derivation services are available for use in Non-Approved Mode: 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 [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.

When using the module's DRBG for key generation, a 256-bit key size and state are used,

resulting in a security strength of up to 256 bits.

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 input into the module during manufacturing.
The module does not receive seeds and seed keys from the outside. The DRBG is only seeded and re-seeded from the TRNG.

7.6 Key storage

User keys within the module are stored only temporarily and in plaintext form.
Key slots are stored in dedicated hardware registers, in plaintext.
Several platform keys are persistently stored either in OTP or RTL; platform keys may be kept in HW registers or SRAM at runtime. For a list of platform keys and their storage locations, see Table 7. All Platform keys are stored in plaintext, with the exception of $\texttt{EK}_{CST}$, the encrypted version of $\texttt{K}_{CST}$.

7.7 Key zeroization

The user can zeroize User Keys in the module at any time by putting the module through a power-on reset or a warm reset. The module explicitly 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 cryptographic operation, and following power-on reset.
The user can zeroize keys in Key Slots (Section 1.4.1.1.2) at any time by calling the TEE service to set the key slots to zero. Key slots are cleared on cold power-on but survive a warm reset.
For platform keys, the entry to RMA mode is used as the zeroization technique. The procedure to zeroize the module through RMA is as follows:

1. The operator places a valid Secure Debug certificate (Section 1.4.1.4.7) with an RMA flag in the designated area of NVRAM.
2. The operator puts the module through a power-on reset. During boot, the certificate is verified and the RMA state becomes available.
3. The operator invokes the RMA service to enter the RMA state.
4. The operator puts the module through a power-on reset again.
5. The operator can use the LCS Read service to verify that the device has indeed entered RMA mode.

Following the second power-on reset, platform keys are erased or made unavailable as described in Section 7.2.1.

8 Electromagnetic Interference / Compatibility (EMI/EMC)

The module meets the applicable FCC EMI/EMC requirements for FIPS 140-2 level 1.

9 Self Tests

According to FIPS 140-2 requirements, CryptoCell-712 performs a number of self-tests on power-up and conditionally on selected events.

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 TEE and the REE, 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's TEE Library Initialization performs power-up tests, but is normally only called once on cold power-on. The operator can initiate a power-on reset to have the module perform the tests.

9.1.1 Cryptography test

The module performs KAT (Known Answer Test) 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 repeated in both TEE and REE

The following tests are performed twice, once in the TEE and once in the REE. The module does not perform comparison tests between the different implementations.

- 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 bit key size.
- 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, SHA-256, and SHA-512.
- HMAC generation for HMAC SHA-256.

9.1.1.2 Tests in REE

The following tests are performed in the REE only:

- AES authentication in GCM and GMAC mode, using 128, 192, and 256-bit key sizes.

9.1.1.3 Tests in the TEE

The following tests are performed in the TEE only.

- AES MAC generation with CBC-MAC using 128-bit and 256-bit keys.
- XTS-AES encryption and decryption operations, using 512 bit key size.
- Triple-DES encryption and decryption operations, ECB and CBC modes, with 2-key bundles.
- DRBG Instantiate, Reseed with Additional Input, and 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 digest.
- KAS DH — Primitive "'z"' operation (as part of a NIST SP800-56A key agreement), with prime size of 2048 bits.
- KAS EC-DH — Primitive "'z"' operation (as part of a NIST SP800-56A key agreement protocol), using the P256 curve.

Note: the boot-time Firmware Certificate Verification service has its own implementation of the RSA-PSS scheme. This is used in module's boot process to verify the firmware image integrity, and constitutes a KAT (Known Answer Test) of the algorithm.

9.1.2 Firmware integrity test

The module performs firmware integrity tests as part of the boot sequence.
The TEE Boot 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 TEE and REE firmware integrity is checked as part of Secure Boot (Section 1.4.1.4.6), using the Verify Boot Image service (RSA-2048 PSS with SHA-256).

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 and decryption operation, and ECC is tested with a signature generation and verification operation.
The CRNGT (Continuous Random Number Generator Test) is implemented in the module's two approved 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 either CRNGT (Continuous Random Number Generator Test) will put the entire module into the Error State.

10 Design Assurance

10.1 Guidance

10.1.1 Operator guidance

- 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 1.3.1).
- 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 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 is order to fulfill the requirements of [SP800-56A] section 5.6.2.

10.2 Proprietary documentation

The following proprietary documents are available to Arm® partners for this product:

- Arm® TrustZone® CryptoCell-712 Configuration and Integration Manual
- Arm® TrustZone® CryptoCell-712 Technical Reference Manual
- Arm® TrustZone® CryptoCell-712 Software Integration Manual
- Arm® TrustZone® CryptoCell-712 Software Developers Manual
- Arm® TrustZone® CryptoCell-712 TRNG Characterization Application Note

11 Mitigation of Other Attacks

The cryptographic module is not designed to mitigate specific attacks.

Glossary

AIB	Intel Asynchronous Interface Bus Specification
APB	AMBA Peripheral Bus
AXI	Advanced eXtensible Interface
CMPU	Chip Manufacturing Production Utility
CRNGT	Continuous Random Number Generator Test
CRYS	Cryptographic Software Library
CSP	Critical Security Parameter
DCU	Debug Control Unit
DH	Diffie-Hellman
DMA	Direct Memory Access
DRBG	Deterministic Random Bit Generator
DRM	Digital Rights Management
EC-DH	Elliptic Curve Diffie Hellman
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Scheme
ECIES	Elliptic Curve Integrated Encryption Scheme
FFC	Finite Field Cryptography
FPGA	Field Programmable Gate Array
FSM	Finite State Model
HAL	Host Adaptation Layer
HW	Hardware
ICV	Independent Chip Vendor
IP	Intellectual Property
ISV	Independent Software Vendor
IV	Initialization Vector
KAS	Key Agreement Scheme
KAT	Known Answer Test
KBKDF	Key Based Key Derivation Function
LCS	Life Cycle State
MAC	Message Authentication Code
NVRAM	Non Volatile Random Access Memory
OEM	Original Equipment Manufacturer
OS	Operating System
OTP	One-Time Programmable memory
PAL	Platform Adaptation Layer
PKA	Public Key Accelerator
RAM	Random Access Memory
RBG	Random Bit Generator
REE	Rich Execution Environment
RMA	Return Merchandising Authorization
RNG	Random Number Generator
ROM	Read-Only Memory
RPMB	Replay Protected Memory Block
RTL	Register Transfer Language
SOC	System on Chip
SRAM	Static Random Access Memory
TEE	Trusted Execution Environment
TRNG	True Random Number Generator

References

[AMBA] ARM. AMBA specifications. http://www.arm.com/products/system-ip/amba-specifications.php.

[ANSI-X9.42] ANSI. ANSI X9.42:2003 agreement of symmetric keys using discrete logarithm cryptography. http://webstore.ansi.org/RecordDetail.aspx?sku=ANSI+X9.42-2003+(R2013), 2003.

[ANSI-X9.63] ANSI. ANSI X9.63 public key cryptography for the financial services industry: Key agreement and key transport using elliptic curve cryptography. http://webstore.ansi.org/RecordDetail.aspx?sku=X9.63-2011, 2011.

[FIPS-180-4] FIPS. FIPS PUB 180-4: Secure hash standard (SHS). http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf, 2015.

[FIPS-186-4] NIST. FIPS PUB 186-4 - digital signature standard (DSS). http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf, 2013.

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

[FIPS-198-1] FIPS. FIPS PUB 198-1: The keyed-hash message authentication code (HMAC). http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf, 2008.

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

[IEEE1363] IEEE. IEEE 1363-2000: Standard specifications for public key cryptography. http://grouper.ieee.org/groups/1363/P1363/index.html, 2000.

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

[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. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50375, 2011.

[JEDEC] JEDEC. JEDEC standard, universal flash storage host controller interface (ufshci), version 2.1. http://www.jedec.org/standards-documents/results/jesd223c, Mar 2016.

[JESD84] JEDEC. JEDEC standard, embedded multimedia card (eMMC), electrical standard - version 5.0. http://www.jedec.org/sites/default/files/docs/JESD84-B50.pdf, 2014.

[PKCS1] RSA Laboratories. PKCS#1 RSA cryptography standard. http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/pkcs-rsa-cryptography-standard.htm, 2012.

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

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

[RFC3566] Sheila Frankel and Howard Herbert. RFC 3566-the AES-XCBC-MAC-96 algorithm and its use with IPsec. https://www.ietf.org/rfc/rfc3566.txt, 2003.

[SP800-107] Quynh Dang. NIST special publication 800-107 revision 1: Recommendation for applications using approved hash algorithms. http://csrc.nist.gov/publications/nistpubs/800-107-rev1/sp800-107-rev1.pdf, 2012.

[SP800-108] Lily Chen. NIST special publication 800-108: Recommendation for key derivation using pseudorandom functions. http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf, 2009.

[SP800-133] NIST. SP 800-133: Recommendation for cryptographic key generation. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133.pdf, 2012.

[SP800-135] Quynh Dang. NIST special publication 800-135 revision 1: Recommendation for existing application-specific key derivation functions. http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-135r1.pdf, 2011.

[SP800-38A] Morris Dworkin. NIST special publication 800-38A - recommendation for block cipher modes of operation. http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf, 2001.

[SP800-38Aadd] Morris Dworkin. Addendum to NIST special publication 800-38a: Recommendation for block cipher modes of operation: Three variants of ciphertext stealing for CBC mode. http://csrc.nist.gov/publications/nistpubs/800-38a/addendum-to-nist_sp800-38A.pdf, 2010.

[SP800-38B] Morris Dworkin. NIST special publication 800-38b: Recommendation for block cipher modes of operation: The CMAC mode for authentication. http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf, 2005.

[SP800-38C] Morris Dworkin. NIST special publication 800-38c: Recommendation for block cipher modes of operation: The CCM mode for authentication and confidentiality. http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C_updated-July20_2007.pdf, 2004.

[SP800-38D] Morris Dworkin. NIST special publication 800-38d: Recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC. http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf, 2007.

[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. http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf, 2010.

[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. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.pdf, 2013.

[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). http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf, 2012.

[SP800-67] William C. Barker and Elaine Barker. NIST special publication 800-67: Recommendation for the triple data encryption algorithm (TDEA) block cipher. http://csrc.nist.gov/publications/nistpubs/800-67-Rev1/SP-800-67-Rev1.pdf, 2012.

[SP800-90A] NIST. NIST special publication 800-90 A: Recommendation for random number generation using deterministic random bit generators. http://dx.doi.org/10.6028/NIST.SP.800-90Ar1, 2015.

