Rambus

CryptoManager Root of Trust (CMRT)

FIPS 140-2 Non-Proprietary Security Policy

Document Revision: 1.3

Document Date: September 2022

Prepared by:

atsec information security Corp.

9130 Jollyville Road, Suite 260

Austin, TX 78759

www.atsec.com

Rambus Inc.

4453 North First Street, Suite 100
San Jose, CA 95134
United States of America

https://www.rambus.com/

Copyrights and Trademarks

Table of contents

1 Introduction

This document is the non-proprietary FIPS 140-2 Security Policy for the Rambus CryptoManager Root of Trust (CMRT) 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 FIPS PUB 140-2 (Federal Information Processing Standards Publication 140-2) for a Security Level 2 module. Throughout the document CryptoManager Root of Trust (CMRT) is referred as “CMRT” or “the 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 a cryptographic module, as implemented, satisfies the stated security policy,
- it describes the capabilities, protection and access rights provided by the cryptographic module, allowing individuals and organizations to determine whether it will 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 audience:

- Hardware designers integrating the CMRT Secure IP into an SoC,
- Software designers who want to understand the capabilities and requirements of the CMRT hardware,
- Hardware verification engineers who want to verify the CMRT hardware behavior and CPU interaction,
- The Cryptographic Module Validation Program (CMVP) an FIPS 140-2 testing lab

2 Cryptographic Module Specification

2.1 Module Overview

The CryptoManager Root of Trust (CMRT) Secure IP is a secure CPU subsystem that supports execution of arbitrary code. The CMRT provides a hierarchical, secure execution environment with a security monitor running at the highest privilege and middleware, akin to the kernel running at the supervisor level. There is a transient single-user process, called a ‘container’, that is akin to a user process. Hardware cores, along with a security monitor, implement and enforce security properties of the CMRT. CMRT can be used stand-alone or as a 'Root of Trust' to support a Trusted Execution Environment (TEE)-based platform.

CMRT completely shields all key and security sensitive data from all CPUs, interfaces and memory. Security sensitive materials are stored as assets that never leave the module in unencrypted and/or non-authenticated form.

Additionally, CMRT offers hardware security features that are needed when operating in a TEE. These features include One-Time-Programmable memory (OTP) access and management, Random Number Generation / entropy source, timers, (short) monotonic/ non-volatile counters and import and export of keys and other assets.

2.2 Intended Usage

CMRT is primarily integrated in the design of Application-Specific Integrated Circuits (ASIC). However, it can also be synthesized in a Field-Programmable Gate Array (FPGA).

The primary application is to provide Root of Trust capabilities to SoCs, where authentication, encrypted content processing using standard protocols, and protection of keys and other sensitive assets are required. CMRT is suited to a wide range of products from low power battery powered devices such as mobile phones, tablets, and wireless handsets, to automotive and AI/ML accelerators. These products require an IP solution with these features available in CMRT:

- Low-power and small footprint IP.
- Internal storage for protection and management of sensitive keys and assets.
- Root of Trust as true hardware interface to on chip One-Time Programmable (OTP) memory.
- Secure Timers (hardware counters).
- Encryption engines to offload computationally intensive AES symmetric algorithms
- Hash engine to offload computational intensive hash algorithms: SHA-2 and supported HMAC-SHA-2.
- Public Key Engine (PKE), supporting RSA, ECDSA (sub-) functions.
- ENT (P).
- Embedded Direct Memory Access Controller (DMAC) for optimizing data transfer within the module.

2.3 FIPS 140-2 Module Information

For the purpose of this Cryptographic Module Validation, CMRT is synthesized and tested on the Xilinx Zynq XC7Z045 FPGA chip soldered into a Xilinx ZC706 base board, which belongs to the Zynq-7000 All Programmable SoC (System on a Chip) series. CMRT is defined as a sub-chip hardware module validated at security level 2; its embodiment is of the type of single chip.

The table below shows 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	2
2	Cryptographic Module Ports and Interfaces	2
3	Roles, Services and Authentication	2
4	Finite State Model	2
5	Physical Security	2
6	Operational Environment	N/A
7	Cryptographic Key Management	2
8	EMI/EMC	2
9	Self Tests	2
10	Design Assurance	2
11	Mitigation of Other Attacks	N/A

Table 1 Security Levels

2.4 Approved Mode of Operation

The module mode of operation is the Operational Mode in which the module is able to use FIPS approved service. The module transitions to Operational Mode upon successful initialization.

2.5 System Block Diagram

The Figure 1 below provides a system block diagram in which the CMRT module (called here RT-630) is integrated in a SoC with one or more CPUs and connects to a common bus system.

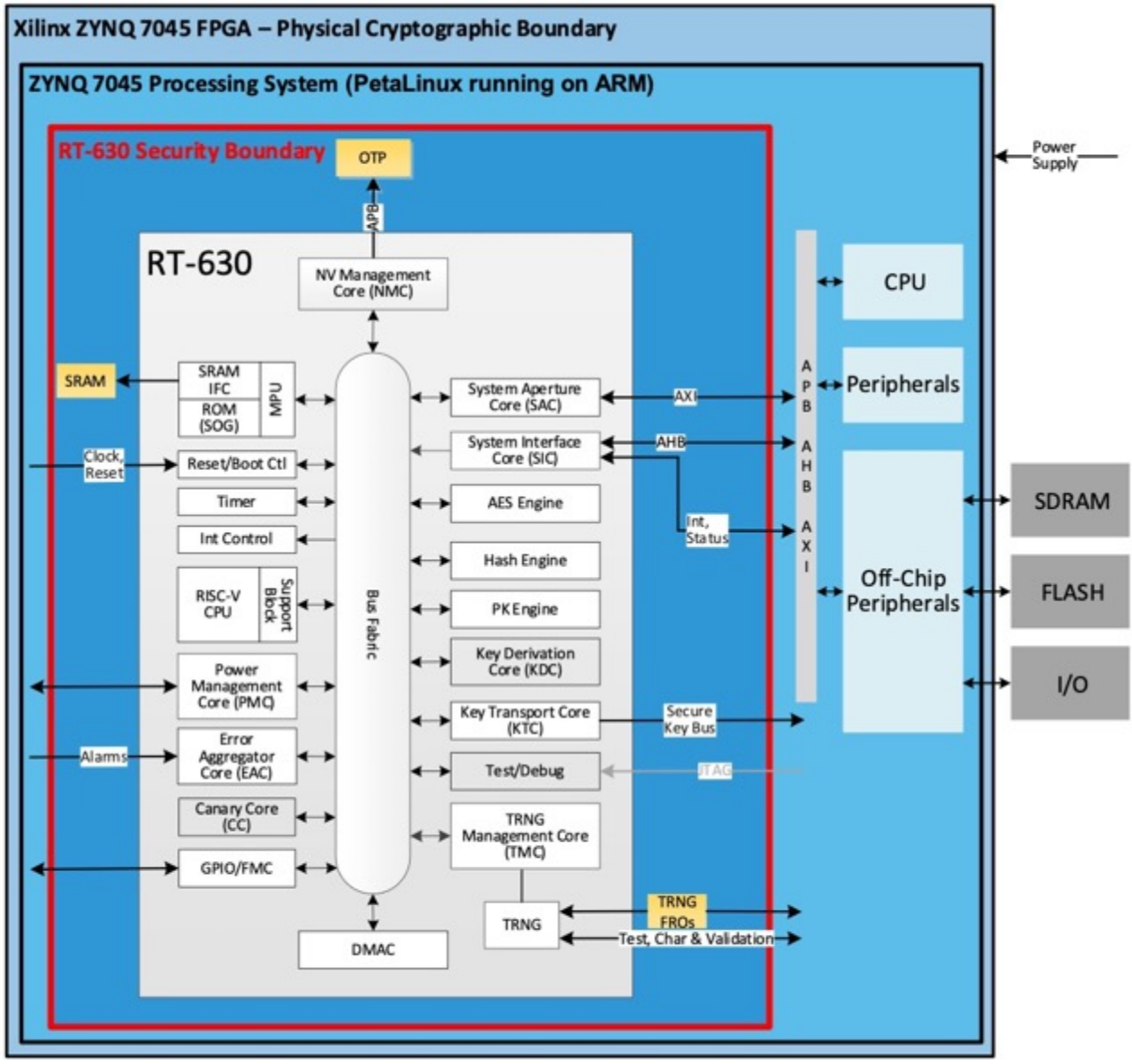

Figure 1 System Block Diagram

For the purpose of this validation, the physical cryptographic boundary is enclosed in the FPGA. The logical cryptographic module boundary is represented with the red line box. The white boxes represent the CMRT components that comprise the IP cores (the CMRT firmware is stored in Program ROM and Program RAM). The yellow boxes represent components that are provided in the IP core but must be replaced or adjusted during the synthesis process as they are technology dependent (OTP, SRAM, TRNG FROs).

2.6 Hardware Block Diagram

CMRT consists of two major components, the Verilog RTL (HW) and the Firmware running from ROM, SRAM, and OTP on the RISC-V processor. The RTL implements the cryptographic algorithms and basic public key big number mathematics. The Firmware handles the higher-

level operations, manages the keys and key storage and takes care of communications and data exchanges external to the CMRT.

Figure 2 shows the details of all interfaces that cross the security boundary and the first hierarchy levels of the CMRT RTL. The greyed-out interface names and arrows are either interfaces for testing purpose only (like jtag_ pins) and are disabled in production or are interfaces not in the scope of the module validation (like low power mode interfaces to PMC).

Firmware is located in the Program ROM to perform the boot process and in Program RAM once the boot process has completed and CMRT is in operational mode. The firmware has many routines; typically, the sources for each routine are located in an individual high level programming language.

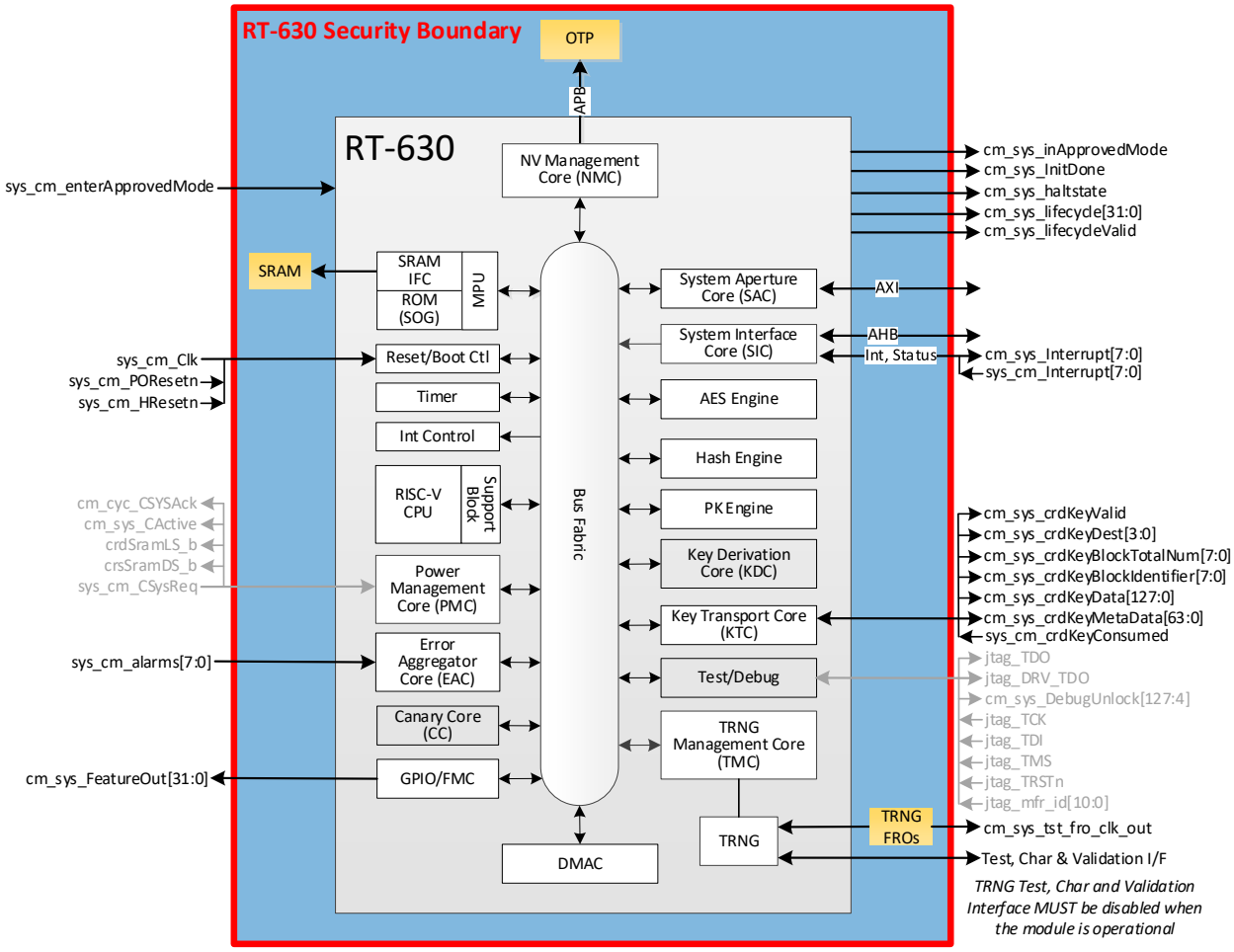

Figure 2 Hardware Block Diagram

2.7 CMRT module breakdown

The next lists show sub-blocks of CMRT and their corresponding version numbers.

- CMRT HW version 0x60000611 with the main embedded sub-blocks:
  - Timer
  - System Interface Core (SIC). SIC is a hardware core responsible for decoding all the external bus accesses to the module.

- System Aperture Core (SAC). SAC is a hardware core which provides a system aperture, or off-core memory interface.
- OTP Interface (NVM Management Core) configures as an APB interface
- Bus Fabric consists of 2 buses: the CPU Secure Bus (CSB) and the Inter-core Secure Bus (ISB).
- ROM (SOG). The Immutable ROM code is implemented in a 'sea of gates' that provides read-only-memory. It is used to store the fboot and its associated data as well as library calls usable by both security monitor and application. fboot is only executed following the successful completion of a CRC32 on the code. The CRC32 is automatically executed following power on or hard reset.
- SRAM IFC. SRAM is a library macro that provides read-write volatile memory. SRAM is used to store the security monitor, application FW, stack, and any other data needed for secure execution of the application. SRAM also contains the FW stack and data.
- Memory Protection Unit (MPU). This unit monitors accesses to the module's memory units – SRAM IFC and ROM, according to operational state. MPU differentiates between access privilege and R/W/X accesses.
- AES Engine with ECB, CBC, CTR, GCM, GMAC and CFB128.
- PKE (Public Key Engine) provides computationally intensive Public Key operations, such as modular exponentiations and Elliptic Curve Cryptography operations.
- HASH Engine, includes SHA-224, SHA-256, SHA-384 and SHA-512, SHA512/224, SHA2-512/256, and HMAC algorithms
- TRNG Management Core (TMC) is a NIST compliant (SP800-90A/B) hardware core that provides the requestor with a 256-bit random number using an ENT (P) and an AES-256 block operating in CTR mode as the DRBG. The entropy source consists of a collection of eight Free Running Oscillators (FROs).
- RISC-V processor running the FW code and CPU support block.
- Direct Memory Access Controller (DMAC). DMAC is managing the DMA a data transfer protocol requesting data reads or writes to or from the module's memory without the intervention of the RISC-V CPU.

- Firmware version 2022-02-21-gd74d034

The CMRT contains a three-stage bootloader that loads the FIPS compliant application firmware.

- CMRT Boot Firmware located in the Program ROM.

  This includes first-stage bootloader binary (fboot) and second-stage bootloader binary (sboot). The fboot located in the 'Sea-of-Gates' (SoG) ROM is the first stage of the secure boot process of the CMRT that verifies integrity, loads and transitions to the sboot. sboot augments fboot capabilities. sboot is located in the OTP non-volatile memory that verifies RAM integrity, loads and transitions to the tboot stage. The third boot stage (tboot) completes the secure boot process and passes control to the application firmware.

- CMRT application Firmware located in the Program RAM.

  This includes service management from and to the SIC, higher-level crypto operations, key generation, TRNG/ DRBG engine, asset management, DMA setup and generic engine control.

- Interfaces of the CMRT:

- System Interface Core (SIC - AHB Slave): interface that connects to the Host SoC using an AHB bus interface. The Host side is the master and the module is the slave.
- System Aperture Core (SAC - AXI Master): interface that connects external system memory to the module. On this interface the CMRT is master and the Host SoC is the slave.
- Key bus master (KTC): interface that output keys from the module to the Host SoC on a dedicated secure interface.

3 Ports and Interfaces

The CMRT module embeds a single slave interface and a single master interface. The slave interface is used to receive commands from one or more host CPU and send the appropriate response. The master interface is used for autonomous data reads and writes from and to an external memory, flash or interface.

Additionally, CMRT includes physical ports for showing the crypto module status establishing the role that is requesting services, and resetting the crypto module.

3.1 Physical ports

The summary of interface pins located on the physical boundary of CMRT is given in the tables below and shown in Figure 2. For clarity, signals are grouped by function in separate tables. Each port pin provides its name, direction, and function.

The set of signals in the table below is hardware related and drives the clock and reset signals.

Port name	Direction	Function
sys_cm_Clk	IN	System Clock
sys_cm_POResetn	IN	Power-on reset
sys_cm_enterApprovedMode	IN	This signal will be sampled by CMRT at the first clock edge after sys_cm_POResetn is released.
sys_cm_HResetn	IN	Hard reset (reset pin)

Table 2 Clock and Reset signals

The set of signals that indicates the status outputs from the module provides information about the module operation is in the table below.

Port name	Direction	Function
cm_sys_InitDone	OUT	Asserted (‘1’) when module initialization is complete; De-asserted by reset (sys_cm_HResetn or sys_cm_POResetn).
cm_sys_haltState	OUT	Asserted (‘1’) when the module is in ‘Halt’ state; De-asserted by reset (sys_cm_HResetn or sys_cm_POResetn).
cm_sys_lifecycle[31:0]	OUT	Module Lifecycle state
cm_sys_lifecycleValid	OUT	Asserted (‘1’) if the Module Lifecycle state information is valid; De-asserted if the Module Lifecycle state information is invalid
cm_sys_FeatureOut[31:0]	OUT	General purpose outputs.
cm_sys_inApprovedMode	OUT	Asserted (‘1’) when the module is in FIPS mode state; De-asserted by reset (sys_cm_HResetn or sys_cm_POResetn).

Table 3 Status signals

The set of signals passing through the Bus master KTC interface is in the table below. This interface is used to deliver keys (asset data) from the module to the Host SoC on a dedicated secure interface using the service *Output Key* (see Table 9). As the KTC has a 128-bit data bus, transfer of an entire key can take multiple cycles.

Port name	Direction	Function
sys_cm_crdKeyConsumed	IN	Asserted ('1') when a block (128 bits) of data has been transferred.
cm_sys_crdKeyValid	OUT	Asserted ('1') when the Key Bus has valid data
cm_sys_crdKeyDest[3:0]	OUT	Specifies the key destination
cm_sys_crdKeyBlockTotalNum[7:0]	OUT	Specifies the total number of blocks (of 128 bits) to be transferred to the key destination
cm_sys_crdKeyBlockIdentifier[7:0]	OUT	Identifies the current block (of 128 bits) being transferred to the key destination
cm_sys_crdKeyData[127:0]	OUT	Contains 1 block of 128 bits of key material
cm_sys_crdKeyMetaData[63:0]	OUT	Contains the key metadata. Constant until all blocks have been transferred

Table 4 Bus Master Key Transport Core (KTC) signals

The set of signals passing through the HOST SoC interface is in the table below.

Port name	Direction	Function
sys_cm_alarms[7:0]	IN	Alarm signals from the Host SoC to inform CMRT that there has been a system level alarm external to CMRT. The CMRT can choose how to respond to these signals.

Table 5 Host SoC interface signal

The set of signals passing through the SIC AHB slave Interface is in the table below. The SIC interface connects to the Host SoC using an AHB bus interface. Through this interface the Host's HLOS can communicate with the module.

Port name	Direction	Function
sys_cm_SicHready	IN	System ready. Indicates the completion of current transfer. De-assertion of this pin inserts wait states into the transfer. This signal is delivered to both AHB master & slave ports if both exist.
sys_cm_SicHaddr[31:0]	IN	System address bus. This 32-bit AHB slave address bus indicates the address to be used for a transfer. The address should be aligned according to the transfer size (sys_cm_Hsize[2:0]), even for IDLE transfers.
sys_cm_SicHburst[2:0]	IN	System burst type. This bus is driven by the system and is used to indicate the kind of burst being performed.
sys_cm_SicHprot[6:0]	IN	System protection. This 7-bit bus indicates the characteristics of the transfer, such as: - Instruction fetch or a data fetch - User or privileged access - Buffered or non-buffered - Cacheable or non-cacheable - Modifiable - Lookup - Allocate - Shareable
sys_cm_SicHsize[2:0]	IN	These pins define the size of the read or write transfer to be performed.

Port name	Direction	Function
sys_cm_SicHnonsec	IN	This pin is driven by the system to indicate a non-secure access
sys_cm_SicHtrans [1:0]	IN	System transfer type. This 2-pin bus indicates the type of transfer
sys_cm_SicHwdata[31:0]	IN	System 32-bit AHB write data bus. Data on this bus is driven by the system to the CryptoManager core.
sys_cm_SicHwrite	IN	System AHB write. This pin is driven by the slave to show the direction of data transfer
sys_cm_SicHsel	IN	System arbitration. The system drives this pin to indicate that it has a transaction waiting. The arbiter has an address-map decoder and drives this pin when the access is to the Crypto-Manager core.
sys_cm_SicHmastlock	IN	SIC AHB MASTER LOCK (currently unsupported)
sys_cm_SicHexcl	IN	SIC AHB EXCLUSIVE (currently unsupported)
cm_sys_SicHrdata[31:0]	OUT	32-bit read data bus. Data on this bus is driven by the CryptoManager core to the system.
cm_sys_SicHreadyout	OUT	Ready out. This pin is used to stall a transfer and insert idle cycles
cm_sys_SicHresp	OUT	This pin is driven by the CryptoManager core to indicate whether the transfer was successful.
sys_cm_Interrupt[7:0]	IN	When asserted (‘1’), indicates one or more interrupt requests to the module occurred.
cm_sys_Interrupt[7:0]	OUT	When asserted (‘1’), one or more of the enabled SoC_INTERRUPT_OUT events have occurred

Table 6 SIC (through AHB slave) signals

The set of signals passing through the master data (AXI) interface is in the table below. The SAC interface connects system memory to the CMRT.

Port name	Direction	Function
AXI Read Data Channel		
sys_cm_Rid[3:0]	IN	This signal is the identification tag for the read data group of signals generated by the slave.
sys_cm_Rdata[31:0]	IN	Read Data.
cm_sys_Rready	OUT	This signal indicates that the master can accept the read data and response information.
sys_cm_Rresp[1:0]	IN	This signal indicates the status of the read transfer.
sys_cm_Rvalid	IN	This signal indicates that the channel is signaling the required read data.
sys_cm_Rlast	IN	This signal indicates the last transfer in a read burst.
AXI Read Address Channel		
cm_sys_ARaddr[31:0]	OUT	The read address gives the address of the first transfer in a read burst transaction
cm_sys_ARburst[1:0]	OUT	The burst type and the size information determine how the address for each transfer within the burst is calculated.
cm_sys_ARcache[3:0]	OUT	This signal indicates how transactions are required to progress through a system.

Port name	Direction	Function
cm_sys_Arid[3:0]	OUT	This signal is the identification tag for the read address group of signals.
cm_sys_Arlen[7:0]	OUT	This signal indicates the exact number of transfers in a burst.
cm_sys_ARlock	OUT	This signal provides additional information about the atomic characteristics of the transfer. Currently this feature is not supported.
cm_sys_ARprot[2:0]	OUT	This signal indicates the privilege and security level of the transaction, and whether the transaction is a data access or an instruction access.
cm_sys_ARqos[3:0]	OUT	QoS identifier sent for each read transaction.
sys_cm_ARready	IN	This signal indicates that the slave is ready to accept an address and associated control signals.
cm_sys_ARregion[3:0]	OUT	Permits a single physical interface on a slave to be used for multiple logical interfaces. Currently this feature is not supported.
cm_sys_ARsize[2:0]	OUT	This signal indicates the size of each transfer in the burst. 1-4 byte sizes are supported.
cm_sys_Aruser[2:0]	OUT	Optional User-defined signal in the read address channel.
AXI Write Address Channel		
cm_sys_AWaddr[31:0]	OUT	The write address gives the address of the first transfer in a write burst transaction.
cm_sys_AWburst[1:0]	OUT	The burst type and the size information determine how the address for each transfer within the burst is calculated.
cm_sys_AWcache[3:0]	OUT	This signal indicates how transactions are required to progress through a system.
cm_sys_AWid[3:0]	OUT	This signal is the identification tag for the write address group of signals.
cm_sys_AWlen[7:0]	OUT	The burst length gives the exact number of transfers in a burst. This information determines the number of data transfers associated with the address.
cm_sys_AWlock	OUT	Provides additional information about the atomic characteristics of the transfer. Currently this feature is not supported.
cm_sys_AWprot[2:0]	OUT	This signal indicates the privilege and security level of the transaction, and whether the transaction is a data access or an instruction access.
cm_sys_AWqos[3:0]	OUT	The QoS identifier sent for each write transaction. Currently this feature is not supported.
sys_cm_AWready	IN	This signal indicates that the slave is ready to accept an address and associated control signals.
cm_sys_AWregion[3:0]	OUT	Permits a single physical interface on a slave to be used for multiple logical interfaces. Currently this feature is not supported.
cm_sys_AWsize[2:0]	OUT	This signal indicates the size of each transfer in the burst. 1-4 byte sizes are supported.
cm_sys_AWuser[2:0]	OUT	Optional User-defined signal in the write address channel.
cm_sys_AWvalid	OUT	This signal indicates that the channel is signaling valid write address and control information.

Port name	Direction	Function
cm_sys_AWaddr[31:0]	OUT	The write address gives the address of the first transfer in a write burst transaction.
cm_sys_Wlast	OUT	This signal indicates the last transfer in a write burst.
sys_cm_Wready	IN	This signal indicates that the slave can accept the write data.
cm_sys_Wstrb[3:0]	OUT	This signal indicates which byte lanes hold valid data. There is one write strobe bit for each eight bits of the write data bus.
cm_sys_Wuser[2:0]	OUT	Optional User-defined signal in the write data channel.
AXI Write Data Channel		
cm_sys_Wvalid	O	This signal indicates that valid write data and strobes are available.
cm_sys_Wdata[31:0]	O	Write Data
cm_sys_Wlast	O	This signal indicates the last transfer in a write burst.
sys_cm_Wready	IN	This signal indicates that the slave can accept the write data.
cm_sys_Wstrb[3:0]	O	This signal indicates which byte lanes hold valid data. There is one write strobe bit for each eight bits of the write data bus.
cm_sys_Wuser[2:0]	O	Optional User-defined signal in the write data channel.
AXI Write Response Channel		
cm_sys_Bready	O	This signal indicates that the master can accept a write response.
sys_cm_Bvalid	IN	This signal indicates that the channel is signaling a valid write response.
sys_cm_Bid[3:0]	IN	This signal is the ID tag of the write response.
sys_cm_Bresp[1:0]	IN	This signal indicates the status of the write transaction.

Table 7 SAC (through AXI Master) signals

3.2 Logical Interfaces

- The System Interface Core port communicates with the host through the AHB slave interface. The module provides status and communication through the SIC, the primary means of communication between the outside world -external host running High-Level Operation System (HLOS)- and the module. The host and the module communicate by sending and receiving messages through writing to / reading from SIC registers. The SIC interface provides support for data input, control input, data output, and status output interfaces.
- The SAC interface communicates with the external Host through the AXI master interface, providing off-core system memory interface. A request for data is initiated via the SAC interface. Depending on the direction of the SAC, input data is requested, or result data is available to be read. The SAC interface provides support for data input and data output interfaces.
- The module supports a Key transport core interface used to output key material to a high speed external cryptographic engine in the Host SoC. The KTC interface provides support for data output and control input interfaces.
- The Clock and Reset ports provide Control Input interface.
- The Status Signal ports provide the Status Output interface.

- Host SoC interface signal provides Control Input interface.
- The power port of the FPGA used for the validation provides power input to CMRT.

The following table shows the mapping between the ports available in CMRT and the logical interfaces:

Port Groups	Data Input	Data Output	Control Input	Status Output	Power Input
Clock and Reset ports			✓		
Status Signal ports				✓	
Key Transport Core (KTC) ports (Bus Master)		✓	✓		
System Interface Core -SIC- (AHB Slave)	✓	✓	✓	✓	
System Aperture Core -SAC- (AXI Master)	✓	✓			
Host SoC interface			✓		
FPGA power port					✓

Table 8 Ports and Logical Interfaces

4 Roles, Services and Authentication

4.1 Roles

The CMRT module is usually installed as part of a SoC, where applications running on the system can use the CMRT cryptographic services. Applications must identify and authenticate to CMRT through one of the following roles:

- **User Role**: This role performs general services, cryptographic operations, other approved security functions and asset managements.
- **Crypto Officer Role**: This role can perform the same functionality as the user role, but it also performs other services in the cryptographic module (e.g., create/delete Users, zeroize the module).

The CMRT module implements a role-based authentication method (see section 4.3). Internal mechanisms of the CMRT ensure that User and Crypto Officer service requests and assets (key material and other CSPs) are properly separated and protected. All services require an authorized role. The CMRT does not support concurrent operators.

4.2 Services

The following table presents the services provided by CMRT, including:

- The authorized roles: a checkmark in the role column indicates that the entity authenticated with that role can perform the service.
- The cryptographic algorithms involved in the service.
- The Keys and Critical Security Parameters (CSPs) involved in the service.
- How the service accesses the Keys and CSPs: (C)reate (create and fill the CSP), (R)ead (read an existing CSP), (D)elete (delete asset and/or zeroize memory where the asset is stored).

Service	Description	Roles		Cryptographic Algorithms	Keys and CSPs (asset data)	Access
		User	CO			
Login	This service establishes the session between the entity accessing the module and the CMRT.	✓	✓	SHA256, ECDSA P-256	ECDSA Public Key	R
Logout	This service closed the session	✓	✓	none	None	N/A
Create User	Create a new User in the OTP root table.		✓	none	ECDSA Public Key	C
Delete User	Delete User's public key and corresponding Hash in the OTP root table.		✓	none	ECDSA Public Key	D

Service	Description	Roles		Cryptographic Algorithms	Keys and CSPs (asset data)	Access
		User	CO			
Generate Symmetric Key	Generate a symmetric key	✓	✓	DRBG	AES Key, HMAC Key	C
Derive Symmetric Key, one-step	Derive a symmetric key of the requested length via SP800-108 or SP800-56C KDF	✓	✓	SP800-108 or SP800-56C KDF	HMAC Key	R
Derive Symmetric Key, two-steps	Derive a symmetric key of the requested length via the two-step process described in SP800-56C	✓	✓	SP800-56C KDF	HMAC Key	R
Generate EC Keypair	Generate a keypair for a requested elliptic curve.	✓	✓	ECDSA / ECDH with P-224, P-256, P-384, P-521	ECDSA Key Pair, ECDH Key Pair	C
Generate RSA Keypair	Generate RSA Key Pair	✓	✓	RSA PSS n-= 2048, 3072 or 4096	RSA key pair	C
Generate RSA Keypair	Generate RSA Key Pair	✓	✓	RSA PSS -CRT n-= 2048, 3072 or 4096	RSA-CRT key pair	C
Import Key	Import a key that is wrapped via the KWP method	✓	✓	AES-KWP	AES-KWP Key	R
Export Key	Export a key from static or dynamic storage.	✓	✓	AES-KWP	AES-KWP Key	R
Key Output	Output a key from static or dynamic storage on the Key Transport Core (KTC) bus in plaintext.	✓	✓	N/A	AES Key, HMAC Key,	R
AES-ECB Encrypt/ Decrypt	Executes AES-ECB mode encrypt or decrypt operation	✓	✓	AES-ECB	AES Key	R
Authenticated Encrypt	Execute an AES-GCM encrypt operation	✓	✓	AES-GCM	AES Key	C
Authenticated Decrypt	Execute an AES-GCM decrypt operation.	✓	✓	AES-GCM	AES Key	R

Service	Description	Roles		Cryptographic Algorithms	Keys and CSPs (asset data)	Access
		User	CO			
AES-CBC Encrypt/ Decrypt	Executes AES-CBC mode encrypt or decrypt operation	✓	✓	AES-CBC	AES Key	R
AES-CTR Encrypt/ Decrypt	Executes AES-CTR mode encrypt or decrypt operation	✓	✓	AES-CTR	AES Key	R
AES-CFB128 Encrypt/ Decrypt	Executes AES-CFB128 mode encrypt or decrypt operation	✓	✓	AES-CFB128	AES Key	R
ECDSA Sign	Sign a message with a specified EC private key	✓	✓	ECDSA P-224, P-256, P-384, P-521 SHA224 / 256 / 384 / 512 / 512_224 / 512_256	ECDSA Private Key	R
ECDSA Verify	Verify the signature of a message with a specified EC public key	✓	✓	ECDSA P-224, P-256, P-384, P-521 SHA224 / 256 / 384 / 512 / 512_224 / 512_256	ECDSA Public Key	R
ECDSA key verification	Test that an ECDSA public key is a point on the specified elliptic curve	✓	✓	ECDSA P-224, P-256, P-384, P-521	ECDSA Public Key	R
ECDH	Calculate a shared secret via the ECDH algorithm.	✓	✓	ECDH P-224, P-256, P-384, P-521	EC private Key, Shared Secret “Z” ECDH other's party public key	R C R
ECDH key agreement	Provide [SP800-56Ar3] ECDH KAS Ephemeral Unified using KDF [SP800-56C]	✓	✓	ECDH (P-224, P-256, P-384, P-521), [SP800-56C] KDF	EC key pair Shared Secret “Z” ECDH other's party public key Derived key	R C R C
RSA Sign	Generate a signature with PKCS#1 v2.1 PSS padding	✓	✓	n-= 2048, 3072 or 4096	RSA / RSA-CRT Private Key	R
RSA Verify	Verify a signature with PKCS#1 v2.1 PSS padding	✓	✓	n-= 2048, 3072 or 4096	RSA / RSA-CRT Public Key	R

Service	Description	Roles		Cryptographic Algorithms	Keys and CSPs (asset data)	Access
		User	CO			
MAC Generation	Generate an HMAC digest using the requested SHA2 algorithm.	✓	✓	HMAC-SHA224 / 256 / 384 / 512 / 512_224 / 512_256	HMAC Key	C
MAC Verification	Verify an HMAC digest using the requested SHA2 algorithm	✓	✓	HMAC-SHA224 / 256 / 384 / 512 / 512_224 / 512_256	HMAC Key	R
Hash	Generate a hash digest for the requested SHA2 algorithm	✓	✓	SHA224 / 256 / 384 / 512 / 512_224 / 512_256	None	N/A
Get TRNG	Get a random number from the TMC	✓	✓	AES-CTR_DRBG	Entropy Input, Seed Internal DRBG state	RC RC
List Assets	List the assets in either static or dynamic asset storage	✓	✓	N/A	AES Key, HMAC Key, RSA Public Key, RSA-PF Keypairs, RSA Key Pairs, RSA-CRT keypairs, ECDSA public keys, ECDSA Keypairs, ECDH Public Keys, ECDH Keypairs, Share Secret “Z”, AES-GCM Key, AES KWP key	N/A
Move Asset	Move an asset from dynamic to static storage	✓	✓	N/A	AES Key, HMAC Key, RSA Public Key, RSA-PF Keypairs, RSA Key Pairs, RSA-CRT keypairs, ECDSA public keys, ECDSA Keypairs, ECDH Public Keys, ECDH Keypairs, AES-GCM Key, AES KWP key	N/A

Service	Description	Roles		Cryptographic Algorithms	Keys and CSPs (asset data)	Access
		User	CO			
Delete Dynamic Asset	Delete an asset in dynamic storage. Delete only if the specific User "owns" the asset.	✓	✓	N/A	AES Key, HMAC Key, RSA Public Key, RSA-PF Keypair, RSA Key Pair, RSA-CRT keypair, ECDSA public key, ECDSA Keypair, ECDH Public Key, ECDH Keypair, Share Secret "Z", AES-GCM Key, AES KWP key	D
Delete Static Asset	Obliterate a static asset in OTP by writing all 0s to 1s in each data word. Delete only if the specific User "owns" the asset.	✓	✓	N/A	AES Key, HMAC Key, RSA Public Key, RSA-PF Keypair, RSA Key Pair, RSA-CRT keypair, ECDSA public key, ECDSA Keypair, ECDH Public Key, ECDH Keypair, AES-GCM Key, AES KWP key	D
Zeroize	Zeroize or obliterate all keys and CSPs in SRAM and /or OTP		✓	N/A	CSPs in the OTP and/or SRAM	D

Service	Description	Roles		Cryptographic Algorithms	Keys and CSPs (asset data)	Access
		User	CO			
Self-test_KAT	Execute the RAM firmware KATs listed in Table 16	✓	✓	AES-CBC Enc/Dec, AES-GCM Enc/Dec, AES-CTR Enc/Dec, AES-CFB Enc/Dec, SHA 256 / 512, HMAC SHA 256, HMAC SHA 256_DF, RSA (PSS) SigGen / SigVer, ECDH Z computation, ECDSA P-256 SigGen / SigVer, AES-CTR_DRBG, KBKDF	AES Key, HMAC Key, RSA Key pair, ECDSA Key pair	CR
Hard Reset	On-demand self-test (FW integrity tests and KATs)	✓	✓	N/A	N/A	N/A
Soft Reset	Soft-Reset the CMRT	✓	✓	N/A	N/A	N/A
Show Status	Return the hardware and firmware versions and approved mode status	✓	✓	N/A	N/A	N/A
DRBG	Send commands to the random number generator to support DRBG testing		✓	AES-CTR_DRBG	Seed Internal DRBG state Internal Counter	RC RC C

Table 9 CMRT Services

The following table shows the FIPS-Approved algorithms which are validated under the CAVP with certificate number **#A1102**.

Algorithm	Usage	Key lengths	Modes	Standards
AES	Encryption Decryption	128, 256 bits	ECB, CBC, CTR, CFB128	[FIPS197] [SP800-38A]
	Encryption Decryption	128, 256 bits	GCM	[SP800-38D]
	MAC	128, 256 bits	GMAC	[SP800-38B] [SP800-38D]
AES Key Wrapping	Key Wrapping	128, 256 bits	KWP	[SP800-38F]

Algorithm	Usage	Key lengths	Modes	Standards
				[RFC3394] [RFC5649]
DRBG	Random Number Generation		AES-256 in CTR mode, with derivation function, prediction resistance disabled / enabled	[SP800-90A] [SP800-38A]
ECDSA	Key Generation Key Verification	P-224, P-256, P-384, P-521	Extra bits for key generation	[FIPS186-4]
	Signature Generation Signature Verification		Hash Algorithm: SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	
	Integrity test of Firmware RAM (Signature Verification)	P-256	SHA-256	
EC	Key Pair Generation	P-224, P-256, P-384, P-521	Extra bits	[SP800-56Ar3] [FIPS186-4]
KAS-ECC with one-step and two-step KDF (KAS Cert.)	ephemeral unified model Key Agreement scheme	P-224, P-256, P-384, P-521	HMAC SHA256	[SP800-56Ar3] [SP800-56AC]
KAS-ECC-CDH component (KAS - SSC Cert.)	Shared secret computation primitive	P-224, P-256, P-384, P-521	N/A	[SP800-56A][1]
HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 HMAC SHA-512/224 HMAC SHA-512/256	MAC Generation MAC Verification	112-256 bits	HMAC Key	[FIPS198-1]
KBKDF	Key Derivation	8, 72, 128, 776, 3456, 4096 bits	Counter mode using HMAC-SHA-256	[SP800-108] [FIPS198-1] [SP800-38B]
SP800-56C KDF (KDA Cert.)	Key Derivation	Derived Key Length: 256 Shared Secret Length: 256-512 Increment 128	Counter mode using HMAC-SHA-256 (one step, two steps)	[SP800-56C] [FIPS198-1] [SP800-38B]

[1] CAVP testing for ECC-CDH component is same for SP 800-56A and SP 800-56ARev3.

Algorithm	Usage	Key lengths	Modes	Standards
RSA	Key Generation Signature Verification Signature Generation	modulus: 2048, 3072, 4096	RSA-PSS with standard key and CRT key format	[FIPS186-4] [PKCS#1]
SHA-224, SHA-256, SHA-384, SHA-512 SHA-512/224 SHA-512/256	Message Digest	N/A	N/A	[FIPS180-4]
ENT (P)	Random Number Generation	N/A	N/A	[SP800-90B]

Table 10 FIPS approved cryptographic algorithms

CMRT also supports the following algorithm for firmware integrity:

Algorithm	Usage
CRC-32	Integrity Test (fboot ROM Firmware)

Table 11 Firmware integrity algorithm

4.3 Identification and Authentication

The Login service must be executed before any other CMRT services can be requested. After logging in, an operator (Crypto Officer -CO- or User_0 to User_5) may assume a different role only after logging out followed by logging back in as the different role. The process to login is as follows:

1) An entity accessing the module requests a role (CO or User) and provides the ECDSA P-256 public key and 128-bit nonce.
2) The module calculates SHA256 hash of the public key and compares it to the value found in the OTP root table for the requested role. If the role hasn't been created, or the hash of the public key doesn't match the value expected, an error is returned.
3) If the hash of the public key matches the values found in the OTP root table, the module returns the session identifier and its own 128-bit nonce.
4) The entity accessing the module then returns the signature of the SHA256 hash digest of the concatenation of the role identifier (CO or User), both nonce, and the role's ECDSA P-256 public key. The purpose of this signature is to prove that the entity is in possession of the private key associated with the public key.
5) The module verifies the provided signature using the provided public key, upon successful verification, the authentication succeeds. If the signature verification fails, an error is returned.

At initial power-on of the module, the CO is the only role for which the hash value of its public key is provisioned in the OTP Root table stored in non-volatile memory. After a CO has created up to six new Users (User_0 to User_5), for each new user the SHA256 hashes of their respective

ECDSA P-256 public keys of will be added to the OTP Root table. Power cycling or hard reset of the module will retain the hash values.

A User's hash value of its public key is only cleared from the non-volatile memory when the CO requests the service "Delete User". If the CO requests the service "Zeroize", hash values of for all roles respective public keys will be cleared from the non-volatile memory (see 7.6).

Authentication status for the User or CO role is not maintained over the power cycle. The power cycle clears the previous authentication results. Therefore after a power cycle or reboot, each role needs to be re-authenticated using the "Login" process explained above.

4.4 Mechanism and Strength of Authentication

The ability to successfully authenticate the entity accessing the module depends on the ability to guess the signing ECDSA private key that matches the verified public key. The ECDSA curve size used for authentication is P-256. According to table 1 in FIPS IG 7.5, an ECDSA curve size of 256 bits provides 128 bits of security strength. Therefore, the chance of a random authentication attempt falsely succeeding is 1 /2^{128} $P_{role} = \frac{1}{2^{128}}$ which is less than the required 1/1,000,000.

Let's consider a conservative approach with a failed authentication rate of 1 per 1µs and 60,000,000 consecutive attempts per minute (60s / 0.000001s). The probability of successfully authentication is then less than or equal to 60,000 * 1 / 2^{128}(≤1.76324e-31) which is much less than the required 1 / 100,000 or 10e-5 false acceptance rate within one minute.

5 Physical Security

For the purpose of this validation, CMRT was synthesized in the Xilinx Zynq XC7Z045 FPGA. This FPGA is a single chip that includes standard passivation provided by an integrated heat spreader (IHS). The IHS serves as a protective shell around the processing silicon, a pathway for heat to be exchanged between the SoC and SoC cooler. The IHS lid and the substrate with solder ball grid array provide opacity in the visible spectrum and prevent any access to the interior of the chip. The FPGA conforms to Level 2 requirements for physical security.

6 Operational Environment

The module operates in a non-modifiable environment.

7 Cryptographic Key and CSP Management

The CMRT module manages two different type of assets: static asset stored in non-volatile memory file in OTP filesystems and dynamic asset stored in the memory file of the SRAM filesystems. The storage files are supported by those two different filesystems that are implemented with proprietary Rambus Trusted Execution Environment file systems (TEEFS).

Prior creation of an asset a session should be established successfully by one of the roles defined in section 4.1 (crypto officer or User_0-5). An asset is created by a service execution according to the correspondence between service and asset given in Table 9. Assets are stored in plaintext within the module and protected from disclosure and modification. The CMRT module does not support updating or modifying an asset, only assets can be deleted or zeroized (see section 7.6).

Table 12 summarizes the cryptographic keys used in CMRT with the key lengths supported, the available methods for key generation, entry to the module, output from the module and the way assets are stored.

Asset Name	Key Length / CSP Size	Generation				Entry	Storage		Output	
		KDF	Random	Key Pair	KAS	Keywrap	Static	Dynamic	KTC	KWP
AES keys	128 and 256 bits	✓	✓			✓	✓	✓	✓	✓
HMAC keys	SHA-224: 112 bits SHA-256: 128 bits SHA-384: 192 bits SHA-512: 256 bits SHA-512/224: 256 bits SHA-512/256: 256 bits	✓	✓			✓	✓	✓	✓	✓
RSA key pair	2048, 3072 and 4096 bits			✓		✓	✓	✓		✓
ECDSA key pair	224, 256, 384 and 521 bits			✓		✓	✓	✓		✓
EC Diffie-Hellman key pair[2]	224, 256, 384 and 521 bits			✓		✓[3]	✓	✓		✓
Entropy input string to seed DRBG								✓		
DRBG internal state and seed								✓		
ECDH Shared Secret (Z)	224, 256, 384 and 521 bits				✓	✓		✓		✓
Derived Key	256 bits	✓				✓		✓	✓	✓

Table 12 Life Cycle of Keys and CSPs

[2] Although the imported EC Diffie-Hellman key pair is included in both dynamic and static assets of CMRT, the imported ECDH key pair is not usable by any ECDH module's services. All the ECDH key pairs (including the unusable imported ECHD key pair or the internally generated ECDH key pair) can be exported using the *Export Key* service.

7.1 Key Generation

CMRT provides services for generating asymmetric and symmetric keys. The key generation methods implemented in the module are compliant with [SP800-133r2] section 4 (vendor affirmed).

CMRT implements symmetric key generation for AES and HMAC keys using random data obtained from a SP800-90A compliant Deterministic Random Bit Generator (DRBG).

CMRT implements key derivation methods ([SP800-108] or [SP800-56Cr2]) based on a HMAC-SHA-256 counter mode PRF that uses a key-derivation-key (a preexisting key) to generate symmetric key and HMAC key. The symmetric generated key is compliant with [SP800-133r2] section 6.2.2.

CMRT implements RSA and Elliptic Curve DSA (ECDSA) asymmetric key generation services compliant with [FIPS186-4]. A seed (i.e. the random value) used in asymmetric key generation is obtained from the [SP800-90A] DRBG.

Elliptic Curve Diffie-Hellman (ECDH) key pairs are generated by ECDSA.

Intermediate key generation values are not output from the cryptographic module during or after processing the service.

7.2 Key Derivation

CMRT provides key-based derivation in compliance with [SP800-108] and [SP800-56Cr2] one step KDF and [SP800-56Cr2] two step KDF (extraction and expansion). HMAC-SHA-256 counter mode PRF is used in those methods.

7.3 Key Entry and Output

Electronic key entry method is used to import the private and secret keys; there is no manual key import or export method used in the module. The keys are output to the calling entity within the same physical boundary.

Symmetric key, HMAC key, asymmetric key pair and secret assets are imported (*Import Key* service) and exported (*Export Key* service) encrypted with the AES-KWP function [SP 800-38F]. Key-Wrapping-Key used by the AES-KWP algorithm cannot be exported by any methods.

In addition, the module offers KTC port to output symmetric key, and HMAC key using the service *Key Output*. The keys are output in plaintext from the module through the KTC interface.

7.4 Key access control and usage

Asset is stored upon creation with data structures related to asset storage location (SRAM filesystem or OTP filesystem), asset name, asset type and usage policy (see Table 12), asset ownership and asset data (key and CSPs). When invoking an asset creation service, the asset ownership is matching the logged-in role (CO or User – see section 4.3). If there is a mismatch between the login role and the service requested, the service stopped, and an error is returned.

Once and asset is created, the ownership and usage attributes of the asset remain until the asset is deleted or the asset filesystem is zeroized. The asset usage is exclusively reserved for the owner of the asset and only the owner of the asset has access it to. The only exception is when an asset is specifically created with the attribute "FIPS_OWNER_ALL" in which case both CO and User roles can access it. Note however that only CO can create such asset.

The crypto officer can move any dynamic asset. A User can only move its own dynamic assets. The crypto officer can delete any asset. A User can only delete its own assets.

7.5 Key Agreement / Key Transport

CMRT provides SP800-56Arev3 compliant key agreement schemes according to FIPS 140-2 IG D.8 scenario X1 following:

- (path 1) with EC Diffie-Hellman shared secret computation.
- (path 2) with EC Diffie-Hellman key agreement scheme performing ECDH SSC with ephemeral Unified scheme followed by key derivation with [SP800-56C] compliant using one-step or two step KDF.

The key agreement schemes provide between 112 and 256 bits of security strength. The key agreement schemes only allow the use of internally generated EC key pair. The imported EC keys using the module's Import key service cannot be used for ECDH operation. The other party's public key enters the module as an input parameter to the ECDH service.

CMRT also provides key wrapping. As shown in Table 12 and explained in section 7.3, keys can be entered in encrypted form through the key wrapping method using AES in KWP mode with 128, or 256-bit keys, compliant with [SP800-38F]. AES Key establishment methodology (AES key wrapping) provides 128 or 256 bits of encryption strength.

The following table shows the maximum lengths and security strength of the keys that can be imported to and exported from the module using the key agreement or wrapping services.

Key	Maximum Length	Security Strength
AES key	256 bits	256 bits
HMAC key	256 bits	256 bits
RSA key pair	4096 bits	149 bits
ECDSA key pair	521 bits	256 bits
EC Diffie-Hellman key pair[2]	521 bits	256 bits

Table 13 Security Strength of Cryptographic Keys

It is the user's responsibility to use the establishment method with an appropriate key size to ensure FIPS compliance. Using an insufficient AES key size for AES Key Wrapping or an insufficient ECDH key size for KAS will reduce the security strength of the wrapped key or the established secret/ derived key respectively.

7.6 Key / CSP Zeroization

A dynamic asset is deleted from the SRAM filesystem using the *Delete Dynamic Asset* service executed by the owner of the asset. The service can be executed by the crypto officer on any of the asset. The SRAM filesystem and all the dynamic assets that it contains are zeroized following the execution of the *Zeroize* service with parameter "FIPS_ZEROIZE_DYNAMIC". This service can only be executed by the crypto officer. In addition, when the module is hard reset or power-off all the dynamic assets of the SRAM filesystem are deleted.

A static asset is deleted from the OTP filesystem using the *Delete Static Asset* service executed by the owner of the asset. The service can be executed by the crypto officer on any of the asset. The OTP filesystem and all the statics assets that it contains are zeroized following the execution of the *Zeroize* service with parameter "FIPS_ZEROIZE_STATIC". The static memory

becomes unusable. The process is not reversible and is ending the life of the module. The program RAM is still running. This service can only be executed by the crypto officer.

The filesystems and all the assets that they contain are zeroized following the execution of the *Zeroize* service with parameter “FIPS_ZEROIZE_ALL”. The execution of this service is the secure sanitization of the module and corresponds to the decommissioned of the module lifecyle. The module is no more functional after the execution of the service. This service can only be executed by the crypto officer

Zeroization of filesystems and root table is performed by writing the one's complement to each word present in the storage area. The operation is performed in a time that is not sufficient to compromise CSPs. Additionally, the module processes one input message at a time: when a zeroization/delete service is processed no other input message accessing the asset storage filesystems can be executed.

7.7 Random Number Generation

CMRT includes a Deterministic Random Bit Generator (DRBG) based on the CTR_DRBG (with and without prediction resistance; with derivation function) algorithm and AES-256 as the underlying cipher according to [SP800-90A]. CMRT uses this engine to:

- Generate symmetric keys in OTP and SRAM memories with the *Generate Symmetric Key* service.
- Generate asymmetric keys in OTP and SRAM memories with *Generate EC Keypair or Generate RSA Keypair* service.
- Provide random data with the *Get TRNG* service.

CMRT includes an ENT (P) that provides 512 bits of entropy input to seed the DRBG; the DRBG provides 256 bits of security strength. Using the *Get TRNG* service, the Crypto Officer and User can seed or re-seed the DRBG. The module also performs DRBG health tests according to section 11.3 of [SP800-90A].

7.8 True Random Number Generation

The ENT (P) utilizes Free Running Oscillators (FROs) to supply the entropy needed to generate random numbers. The ENT (P) meets the [SP800-90B] requirements and performs health tests (see sections 9.1.4 and 9.3 below).

8 Electromagnetic Interference / Electromagnetic Compatibility (EMI/EMC)

According to 47 Code of Federal Regulations, Part 15, Subpart B, Unintentional Radiators, CMRT is not subject to EMI/EMC regulations because it is a subassembly that is sold to an equipment manufacturer for further fabrication. That manufacturer is responsible for obtaining the necessary authorization for the equipment with CMRT embedded prior to further marketing to a vendor or to a user.

9 Self-Tests

9.1 Power-Up Tests

After successful initialization of the SoC in which CMRT is integrated, CMRT automatically performs power-up tests without user intervention to ensure that the module is not corrupted and that the cryptographic algorithms within the module work as expected.

During the execution of the power-up tests, services are not available, and no data output is possible.

If the power-up tests succeed, then CMRT becomes operational. If the power-up tests fail, then CMRT transitions to the Error state and becomes non-operational.

The POSTs are described in the sub-sections below and include the firmware integrity tests, KATs, and SP800-90B health tests at start-up.

9.1.1 Firmware ROM POST

9.1.1.1 fboot ROM integrity test and KATs

The first code executed by the RISC-V CPU in the module is from the first-stage bootloader (fboot). fboot is realized in a sea-of-gates (SoG) in Verilog RTL and as such is a part of the CMRT FIPS140-2 binary. At power-on the fboot ROM integrity is checked automatically by a hardware 32-bit CRC algorithm without operator intervention, under the control of a hardware state machine. At this point, no firmware has been executed.

If the CRC-32 integrity test succeeds, fboot ROM-based self-test that contains a single KAT for SHA256 (table below) is automatically executed; if the CRC-32 integrity test fails, the cryptographic module transitions into the Error state, becoming non-operational and the INTERNAL_HW_ERROR_INFO register is reading internalHwErrorInfoReg 0x00000002.

Cryptographic Algorithm	Test
CRC 32	ROM integrity
SHS	KAT SHA-256

Table 14 fboot ROM Power-up Test

9.1.1.2 sboot OTP integrity and KATs

The module's sboot verifies the integrity of the firmware sboot located in OTP by computing the SHA256 hash digest and compares it to the value stored in OTP.

If the SHA-256 integrity test succeeds, sboot OTP-based self-tests that contains SHA-256 and ECDSA KATs (Table 15) are automatically executed; if either test fails, the cryptographic module transitions into the Error state, becoming non-operational.

Cryptographic Algorithm	Test
SHS	Integrity of the firmware sboot
SHS	KAT SHA-256
ECDSA	KAT for ECDSA (NIST P-256) signature verification

Table 15 sboot OTP-based Power-up Tests

9.1.2 Firmware RAM Integrity tests

CMRT verifies the integrity of the RAM firmware image received by the host using ECDSA signature verification with a P-256 public key. If verification of the ECDSA digital signature succeeds, CMRT continues with the rest of the power-up tests located in the application firmware; if the integrity test fails, the module does not proceed with the power-up and instead enters the Error state.

9.1.3 Firmware RAM Cryptographic algorithm tests

CMRT performs self-tests (table below) on all FIPS-Approved cryptographic algorithms that can be used in the operational mode of the module.

Cryptographic Algorithm	Test
AES	KAT AES-CBC, 128-bit, encryption KAT AES-CBC, 128-bit, decryption KAT AES-GCM, 256-bit, encryption KAT AES-GCM, 256-bit, decryption KAT AES-CTR, 256-bit, encryption KAT AES-CTR, 256-bit, decryption KAT AES-CFB128, 256-bit, encryption KAT AES-CFB128, 256-bit, decryption
SHS	KAT SHA-256, KAT SHA-512
HMAC	KAT HMAC-SHA-256
RSA	KAT RSA 2048-bit (PSS) signature generation KAT RSA 2048-bit (PSS) signature verification
ECDH	KAT for Z computation (NIST P-256)
ECDSA	KAT ECDSA (NIST P-256) signature generation KAT ECDSA (NIST P-256) signature verification
DRBG AES-CTR (SP800-90A)	KAT AES-CTR-256 DRBG and Health test per SP800-90A section 11.3
KBKDF SP800-108 KDF	KAT SP800-108 KDF (PRF: HMAC-SHA-256 in Counter mode)
SP800-56Cr2 KDF	KAT SP800-56Cr2 one-step KDF (PRF: HMAC-SHA-256) KAT SP800-56Cr2 two-step KDF (PRF: HMAC-SHA-256)

Table 16 Application firmware Power-Up Tests

If any of the known answer tests fail CMRT transitions to the Error state and becomes non-operational.

9.1.4 ENT (P) start-up health tests

The SP800-90B health tests (Adaptive Proportion Test -APT- and Repetition Count Test -RCT) are performed at start-up on 1,024 consecutive samples.

9.2 On-demand self-tests

In order to perform the on-demand self-tests that initiates the POSTs, the Crypto-Officer shall power-off and power-on or do a hardware reset of the CMRT. The Self-test_KATs service (Table 10) performs all the KATs of the firmware application loaded in RAM.

9.3 Conditional Tests

CMRT performs conditional tests on the cryptographic algorithms shown in the following table:

Algorithm	Test
RSA key generation	Pair-wise consistency test.
EC key generation	Pair-wise consistency test.
ENT (P)	[SP800-90B] health tests: APT and RCT.

Table 17 Conditional Tests

If any of these tests fail, the module transitions to the Error state and becomes non-operational.

9.4 Module status

CMRT provides different interfaces to show its status:

- The cm_sys_inApprovedMode output signal is set to 1 to indicate that the cryptographic module is in FIPS mode, the only operational mode for the current validated module.
- The Show Status service provides the mode state indication (Fips:100000001 with bit 9 =1 indicating fips mode), general hardware and firmware version information. This service can be requested by both the User and Crypto-Officer roles.

9.5 Error state

When a FW or HW error occurs, CMRT transitions to the Error state and becomes non-operational. The module can transition to this state for the following reasons:

- Failure of the power-up tests (including failure of the integrity test)
- Failure of the conditional tests as listed in Table 17 above
- Failure of the ENT (P) SP800-90B health tests
- Failure of the service “Self-test_KAT”

Error indication is given by the cm_sys_haltState output port set to 1 when a halt error occurs. Both INTERNAL_HW_ERROR_INFO and INTERNAL_SW_ERROR_INFO registers have values different than 0x0 indicating fatal error and module not operational.

In the Error state, only the Show Status is available. No other services are available and data output is inhibited.

There are two options to clear the Error state:

- Hardware reset the module (sys_cm_HResetn pin).
- Power-off and power-on the module (sys_cm_POResetn).

Both actions will cause the assets in the SRAM (dynamic assets) to be zeroized. The module has to be restarted to return to the operational state.

10 Design Assurance

10.1 Configuration Management

Rambus uses a configuration management system to store all source code, test tools and verification scripts. A change and version control tool is used to identify and track each configuration item. For product documentation, a directory structure and labeling convention for filenames and directories are used to maintain documents under configuration management.

10.1.1 Cryptographic Module Identification

CMRT is uniquely identified by the filenames used to ship the IP core Verilog code (HW) and the application firmware image. The HW version (Builder Version Information) for the CMRT FIPS configuration is 0x60000611. This is a unique identifier for the configuration of the module. The convention adopted for the interpretation of the numbers is as follows :

- Bits [31:28] represents the Product Family of the module with CMRT being 0x6
- Bits [15:8] represents the Customer ID: an internal tracking number assigned by the Rambus design team. For the current module this value is 0x06 and this will be represented in the bits [15:8] as "0000 0110".
- Bits [7:4] represents the Product version for the module equals to 0x1 corresponding to the 1st version
- Bits [3:0] represents the Release milestone for the module equals to 0x1 corresponding to the 1st release.

10.1.2 Guidance Identification

Rambus uniquely identifies each document with the document name and revision number (e.g. DL-1201_RT-630_CryptoManager_RoT_External_Ref_Spec_Rev1.10).

10.1.3 Source Code Identification

Source code is identified and controlled by the configuration management tools. Specifically, RTL code is managed with SVN and the application FW code is managed with GIT.

10.2 Delivery and Operation

CMRT synthesized in the Xilinx Zynq XC7Z045 FPGA is a single chip hardware module. The chip is delivered from the vendor via a trusted delivery courier. Upon reception of CMRT, the customer should verify that the package does not have any irregular tears or openings.

The delivery packages (HW: 950-630913-100, FW: 951-602913-103) directly map the module HW version 0x60000611 and FW version 2022-02-21-gd74d034.

10.3 Guidance

As part of the installation step, the sys_cm_enterApprovedMode pin has to be tied in HW by the integrator or the external system must set the pin to ‘1’ for the module to be operated as a FIPS validated module.

In a FPGA configuration the Crypto Officer should execute the command the "Show Status" service to verify the version numbers of the code included in the chip with the version provided above in 10.2. The HW version shows 0x60000611. The FW version shows 2022-02-21-gd74d034.

The hash value of the Crypto Officer's public key which is used for signature verification as part of authentication procedure is embedded in the OTP Root table and the Create User service can be used to create the user roles.

For the purpose of this cryptographic module validation, CMRT is synthesized in the Xilinx Zynq XC7Z045 FPGA and validated as a single-chip hardware module. Nevertheless, CMRT, as a soft IP core written in RTL code, can also be synthesized in other single-chip implementations with different chip technologies and/or different non-security relevant functional subsystems. FIPS 140-2 IG 1.20 in [FIPS140-2_IG] provides guidance on how to take advantage of a sub-chip validated module and re-validate the module in other single-chip implementations.

Rambus provides the following documentation for integrators to synthesize CMRT in their chips:

Document Name	Document Number
RT-630 CryptoManager RoT External Ref Spec	DL-1201-00-EN-0002-00-GA
Security-IP-76 HW2.4, Integration Manual	007-076240-200
Security-IP-76_HW2.4_Hardware Reference & Programmer Manual	007-076240-207
RT-630 – FIPS 140-2 Embedded Software Architecture Specification	005-630120-320/913

Table 18 CMRT Documentation

The following sections provide further guidance for algorithms in operational mode.

10.3.1 AES GCM IV

CMRT is compliant with scenario 2 of FIPS 140-2 IG A.5 in [FIPS140-2_IG]. The internal IV is generated in the encryption operation using the RBG-based construction method as defined in section 8.2.2 of [SP800-38D]. The IV length is 96 bits; and the IV is generated from the random data obtained from the SP800-90A DRBG implemented in the module.

11 Mitigation of Other Attacks

The cryptographic module does not implement security mechanisms to mitigate other attacks.

A Appendixes

A.1 Glossary and Abbreviations

AES	Advanced Encryption Specification
APT	Adaptive Proportion Test
ASIC	Application Specific Integrated Circuit
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining (AES mode)
CFB	Cipher Feedback (AES mode)
CMRT	CryptoManager Root of Trust (CMRT)
CMVP	Cryptographic Module Validation Program
CPU	Central Processing Unit
CSP	Critical Security Parameter
CRC	cyclic redundancy check (an error-detecting code)
CRT	Chinese Remainder Theorem (RSA key)
CTR	Counter Mode (AES mode)
DMAC	Direct Memory Access Controller
ECDSA	Elliptic Curve Digital Signature Algorithm
EMI/ EMC	Electromagnetic Interference/ Electromagnetic Compatibility
ENT (P)	physical entropy source (from FRO)
FIPS	Federal Information Processing Standards Publication
FPGA	Field-Programmable Gate Array
FRO	Free Running Oscillator
GCM	Galois/Counter Mode (AES mode)
HLOS	High Level Operating System
HMAC	Hash Message Authentication Code
IHS	Integrated metal Heat Spreader
IP	Intellectual Property
KAT	Known Answer Test
KBKDF	Key-based Key Derivation Function
NIST	National Institute of Science and Technology
OTP	One-Time-Programmable memory
PKE	Public key Engine
PRF	Pseudorandom Function
PSS	Probabilistic Signature Scheme
RAM	Read access memory (volatile)
RCT	Repetition Count Test
ROM	Read Only Memory (non-volatile)
RSA	Rivest, Shamir, Addleman
RTL	Register Transfer Level
SHA	Secure Hash Algorithm
SAC	System Aperture Core
SIC	System Interface Core
SHS	Secure Hash Standard
SoC	System on a Chip
SOG	Sea of Gates
TEE	Trusted Execution Environment
TMC	TRNG Management Core
TRNG	True Random Number Generator

A.2 References

FIPS140-2 **FIPS PUB 140-2 - Security Requirements For Cryptographic Modules**
May 2001
http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

FIPS140-2_IG Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program
http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf

FIPS180-4 **Secure Hash Standard (SHS)**
March 2012
http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

FIPS186-4 **Digital Signature Standard (DSS)**
https://doi.org/10.6028/NIST.FIPS.186-4

FIPS197 **Advanced Encryption Standard**
November 2001
http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

FIPS198-1 **The Keyed-Hash Message Authentication Code (HMAC)**
July 2008
http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf

PKCS#1 **Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1**
February 2003
http://www.ietf.org/rfc/rfc3447.txt

RFC5649 **Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm**
September 2009
http://www.ietf.org/rfc/rfc5649.txt

SP800-38A **NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation - Methods and Techniques**
December 2001
http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

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

SP800-38F **NIST Special Publication 800-38F - Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping**
December 2012
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf

SP800-56Ar3 **NIST Special Publication 800-56A Revision 3 - Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography**
https://doi.org/10.6028/NIST.SP.800-56Ar3

SP800-56C **NIST Special Publication 800-56C Revision 2 - Recommendation for Key-Derivation Methods in Key-Establishment Schemes**
https://doi.org/10.6028/NIST.SP.800-56Cr2

SP800-67 **NIST Special Publication 800-67 Revision 1 - Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher**
January 2012
http://csrc.nist.gov/publications/nistpubs/800-67-Rev1/SP-800-67-Rev1.pdf

SP800-90A **NIST Special Publication 800-90A – Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators**
January 2015
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

SP800-90B **NIST Draft Special Publication 800-90B - Recommendation for the Entropy Sources Used for Random Bit Generation**
January 2018
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf

SP800-108 **NIST Special Publication 800-108 - Recommendation for Key Derivation Using Pseudorandom Functions**
October 2009
http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf