CryptoManager Root of Trust RT-660

FIPS 140-3 Non-Proprietary Security Policy

Last update: July 2024

Prepared by:
atsec information security corporation
4516 Seton Center Parkway, Suite 250
Austin, TX 78759
www.atsec.com

Table of Contents

Rambus Inc.

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

https://www.rambus.com/

1 General Information

1.1 Overview

This document is the non-proprietary FIPS 140-3 Security Policy for the CryptoManager Root of Trust RT-660 cryptographic module from Rambus®. 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 140-3 (Federal Information Processing Standards Publication 140-3) for a Security Level 2 module.
This document provides all tables and diagrams (when applicable) required by NIST SP 800-140B. The column names of the tables follow the template tables provided in NIST SP 800-140B.

1.2 Security Levels

Table 1 describes the individual security areas of FIPS 140-3, as well as the Security Levels of those individual areas.

ISO/IEC 24759 Section 6. [Number Below]	FIPS 140-3 Section Title	Security Level
1	General	2
2	Cryptographic Module Specification	2
3	Cryptographic Module Interfaces	2
4	Roles, Services, and Authentication	2
5	Software/Firmware Security	2
6	Operational Environment	N/A
7	Physical Security	2
8	Non-Invasive Security	N/A
9	Sensitive Security Parameter Management	2
10	Self-Tests	2
11	Life-Cycle Assurance	2
12	Mitigation of Other Attacks	2
Overall Security Level		2

Table 1 - Security Levels

2 Cryptographic Module Specification

Description

The CryptoManager Root of Trust RT-660 cryptographic module (hereafter referred to as “the module” or "CMRT") is a sub-chip hardware module and its embodiment is of the type of single chip. 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.

Tested Module Identification – Hardware:

Model	Hardware [Part Number and Version]	Firmware Version	Distinguishing Features
Xilinx Zynq XC7Z045 FPGA	0x6000_0931	2022-02-24-g801c166	N/A

Table 2 - Cryptographic Module Tested Configuration

CMRT is primarily integrated in the design of Application Specific Integrated Circuits (ASIC). However, it can also be configured in a Field Programmable Gate Array (FPGA). For the purpose of this Cryptographic Module Validation, CMRT is configured 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.

Modes of Operation

The module does not require any installation, configuration or initialization steps. Once the module is power on and the self-tests are successful, the module becomes operational and transitions to approved mode automatically.
The mode of operation is assumed based on the service invoked i.e., the module switches back and forth between approved and non-approved modes based on the service called. By default, the module is in approved mode. The non-approved mode of operation is entered when the module utilizes non-approved security functions in Table 9. The module switches back to approved mode of operation when the approved service in Table 8 is called.

Algorithms

Table 3 lists all security functions of the module, including specific keys employed for approved services, and implemented modes of operation.

CAVP Cert.	Algorithm and Standard	Mode / Method	Description / Key Size(s)/ Key strengths (bits)	Use / Function
#A2114	AES [FIPS 197, SP800-38A]	ECB, CBC, CTR, CFB128	128, 192, 256 bits / from 128 to 256	Encryption, Decryption
#A2114	AES [FIPS 197, SP800-38B]	CMAC	128, 192, 256 bits / from 128 to 256	MAC Generation and Verification

CAVP Cert.	Algorithm and Standard	Mode / Method	Description / Key Size(s)/ Key strengths (bits)	Use / Function
#A2114	AES [SP800-38D]	GCM	128, 192, 256 bits / from 128 to 256	Encryption, Decryption
#A2114	AES [SP800-38B, SP800-38D]	GMAC	128, 192, 256 bits / from 128 to 256	MAC Generation and Verification
#A2114	AES Key Wrapping (KTS) [SP800-38F, RFC3394, RFC5649]	KWP	128, 192, 256 bits / from 128 to 256	Key Wrapping
#A2114	DRBG [SP800-90ARev1, SP800-38A]	AES 256 in CTR mode, with derivation function, prediction resistance disabled / enabled	256 bits / 256	Random Number Generation
#A2114	ECDSA [FIPS186-4]	FIPS186-4 B.4.2 Testing Candidates	P-224, P-256, P-384, P-521 / from 112 to 256	Key Generation
#A2114	ECDSA [FIPS186-4]	N/A	P-224, P-256, P-384, P-521 / from 112 to 256	Key Verification
#A2114	ECDSA [FIPS186-4]	SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	P-224, P-256, P-384, P-521 / from 112 to 256	Signature Generation, Signature Verification selecting Hash Core 2
#A2115	ECDSA [FIPS186-4]	SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512	P-224, P-256, P-384, P-521 / from 112 to 256	Signature Generation, Signature Verification selecting Hash Core 1
#A2114	KAS-ECC-SSC [SP800-56ARev3]	ephemeralUnified: KAS Role: initiator, responder	P-224, P-256, P-384, P-521 / from 112 to 256	Shared Secret Computation
#A2114	KAS-ECC [SP800-56ARev3, SP800-56CRev2]	Function: Full Validation Scheme: ephemeral Unified: KAS Role: Initiator, Responder KDF Methods: with One-Step and Two-Step KDF, MAC Modes: HMAC-SHA-256	P-224, P-256, P-384, P-521 / from 112 to 256	Key Agreement
#A2114	HMAC [FIPS198-1]	HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512, HMAC-SHA-512/224, HMAC-SHA-512/256	112-512 bits key sizes with 112-256 bits strength	MAC Generation, MAC Verification selecting Hash Core 2
#A2115	HMAC [FIPS198-1]	HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512, HMAC SHA3-224, HMAC SHA3-256, HMAC SHA3-384, HMAC SHA3-512	112-512 bits key sizes with 112-256 bits strength	MAC Generation, MAC Verification selecting Hash Core 1
#A2114	KBKDF [SP800-108Rev1, FIPS198-1, SP800-38B]	Counter mode using HMAC-SHA-256 as PRF	8- 4096 bits Increment 8 / from 112 to 256	Key Derivation

CAVP Cert.	Algorithm and Standard	Mode / Method	Description / Key Size(s)/ Key strengths (bits)	Use / Function
#A2114	SP800-56CRev2 KDF [SP800-56CRRev2, FIPS198-1, SP800-38B] (KDA Cert.)	Counter mode (one step, two steps) using HMAC-SHA-256 as PRF	Derived key length: 256 Shared secret length: 256-512 Increment 128 / from 112 to 256	Key Derivation
#A2114	RSA [FIPS186-4, PKCS#1]	B.3.3 Probable prime with standard key and CRT key format	2048, 3072, 4096 / from 112 to 149	Key Generation
#A2114	RSA [FIPS186-4, PKCS#1]	RSA PKCSPSS with SHA-224, SHA-256, SHA-384, SHA-512, SHA2-512/224, SHA2-512/256	2048, 3072, 4096 / from 112 to 149	Signature Verification, Signature Generation selecting Hash Core 2
#A2115	RSA [FIPS186-4, PKCS#1]	RSA PKCSPSS with SHA-224, SHA-256, SHA-384, SHA-512	2048, 3072, 4096 / from 112 to 149	Signature Verification, Signature Generation selecting Hash Core 1
vendor affirmed	RSA [FIPS186-4, PKCS#1]. Vendor affirmed per IG C.C comments 2.c with SHA-3 #A2115	RSA PKCSPSS with SHA3-224, SHA3-256, SHA3-384, SHA3-512	2048, 3072, 4096 / from 112 to 149	Signature Verification, Signature Generation selecting Hash Core 1
vendor affirmed	CKG [SP800-133Rev2]	AES with mode in Table 3	128, 192, 256 bit / from 128 to 256	Cryptographic Key Generation
		HMAC key with mode in Table 3	112-512 bits key sizes with 112-256 bits strength	
		RSA key pair	2048, 3072, 4096 / from 112 to 149	
		ECDSA/ EC Diffie-Hellman key pair	P-224, P-256, P-384, P-521 / from 112 to 256	
#A2114	SHS [FIPS180-4]	SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256	N/A	Message Digest selecting Hash Core 2
#A2115	SHS [FIPS180-4] SHA3 [FIPS202]	SHA-224, SHA-256, SHA-384, SHA-512 SHA3-224, SHA3-256, SHA3-384, SHA3-512	N/A	Message Digest selecting Hash Core 1
N/A	ENT (P) [SP800-90B]	N/A	N/A	Random Number Generation

Table 3 - Approved Algorithms

The module does not implement any Non-Approved Algorithms Allowed in the Approved Mode of Operation with or without security claimed.

Table 4 lists Non-Approved security functions that are not Allowed in the Approved Mode of Operation:

Algorithm / Function	Use /Function
EC Diffie-Hellman shared secret computation	Shared secret computation with method and key size(s) described in Table 3, using imported private key (not SP800-56ARev3 compliant)
EC Diffie-Hellman key agreement	Key agreement with method and key size(s) described in Table 3, using imported private key (not SP800-56ARev3 compliant)
Derive Symmetric Key (extraction and expansion steps), KBKDF (expansion step)	Key derivation in counter mode using HMAC-SHA-256 as PRF derives keying material from a shared secret that was generated by the EC Diffie-Hellman key agreement scheme with imported key pair (not SP800-56ARev3/ 56CRev2 compliant)
Derive Symmetric Key (extraction)	Key derivation in counter mode using HMAC-SHA256 as PRF derives keying material from a shared secret that was generated by the EC Diffie-Hellman key agreement scheme with imported key pair (not SP800-56ARev3/ 56CRev2 compliant)

Table 4 - Non-Approved Algorithms Not Allowed in the Approved Mode of Operation

Hardware module Photograph

The module physical boundary is defined by the FPGA perimeter.

Figure 1: Xilinx Zynq XC7Z045 FPGA

Block Diagram

Figure 2 provides a block diagram in which the CMRT (RT-660) is configured: a FPGA with one or more CPUs that connects to a common bus system.

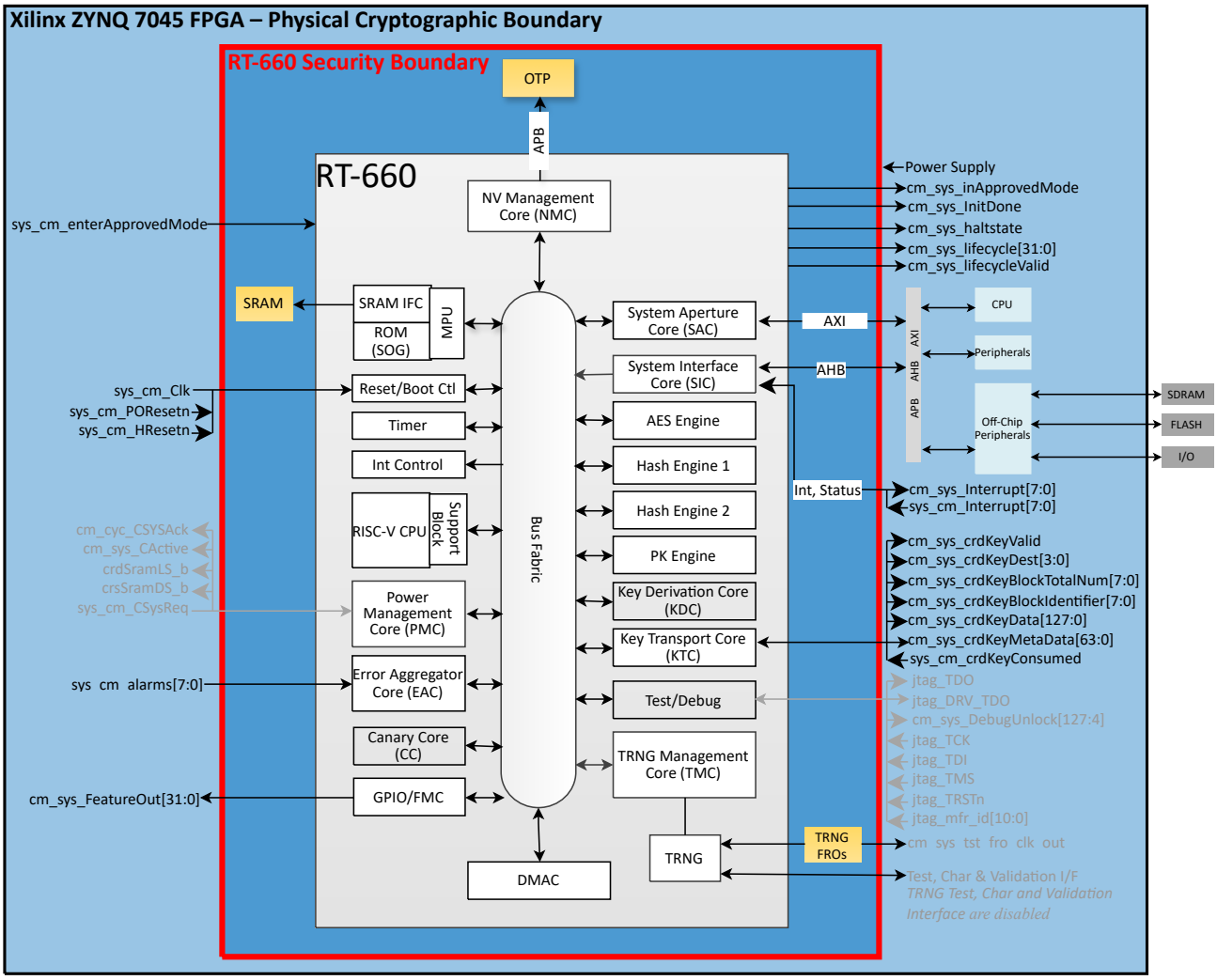

Figure 2 – Module Physical boundary, Module Cryptographic boundary, components and interfaces

In the block diagram, the physical boundary of the single chip is outlined with the black rectangle and the sub-chip cryptographic sub-system boundary, is outlined with the red rectangle. 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 configuration process as they are technology dependent (OTP, SRAM, TRNG FROs).
Also, the block diagram Figure 2 shows the details of all interfaces and the first hierarchy levels of the CMRT RTL. The greyed-out module components and interfaces are for testing purpose only (like jtag_ pins) and are disabled for the end user of the module.

Algorithm Specific Information

Hash core:
The Hash Core 1 (2) are components integrated in the modules' cryptographic boundary as shown in the block diagram with blocks "Hash Engine 1" and “Hash Engine 2" respectively. Hash Core (HC1) supports SHA2-224, SHA2-256, SHA2-384, SHA2-512,

SHA3-224, SHA3-256, SHA3-384, SHA3-512 and corresponding HMAC. Hash Core (HC2) supports SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 and corresponding HMAC. The KATs are run by the RAM firmware self-tests on both cores. The HC1 is selected by default and the user can select HC2 based on service input parameter. The algorithms implemented in each HC and higher-level algorithms with the approved hash have their own ACVP certificates (See Table 3) per IG C.B.

AES-GCM:
CMRT is compliant with scenario 2 of FIPS 140-3 IG C.H in [FIPS140-3_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 CTR_DRBG implemented in the module.

3 Cryptographic Module Interfaces

The 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 via status output interface, and resetting the crypto module via control input interface.
Table 5 describes all the cryptographic module's interfaces. There is a mapping between the physical ports and the corresponding logical interfaces with the data that pass over them.

Physical port	Logical Interface[1]	Data that passes over port/interface
AHB Slave Write)	Data Input, Control Input (SIC)	Data and arbitration from HLOS
AHB Slave (Read)	Data Output, Status Output (SIC)	Data and indicator to HLOS.
AXI Master (Read)	Data Input (SAC)	Data from system memory
AXI Master (Write)	Data Output (SAC)	Data to system memory
sys_cm_Clk	Control Input	Not accessible via API
sys_cm_POResetn		Not accessible via API
sys_cm_HResetn		Not accessible via API
sys_cm_enterApprovedMode		The signal is sampled by CMRT at the first clock edge after sys_cm_POResetn is released
sys_cm_Interrupt[7:0] cm_sys_Interrupt[7:0]	Control Input, Status Output	Interrupt requests to the module
sys_cm_alarms[7:0]	Control Input	Alarm signals from the Host SoC to RT-660
cm_sys_InitDone cm_sys_haltState cm_sys_lifecycle[31:0] cm_sys_lifecycleValid cm_sys_Interrupt[7:0] cm_sys_FeatureOut[31:0] cm_sys_inApprovedMode	Status Output	Information about the module operation
Bus Master (Write) cm_sys_crdKeyValid cm_sys_crdKeyDest[3:0] cm_sys_crdKeyBlockTotalNum[7:0] cm_sys_crdKeyBlockIdentifier[7:0] cm_sys_crdKeyData[127:0] cm_sys_crdKeyMetaData[63:0]	Data Output (Key Transport Core -KTC)	Data for KTC transfer. KTC is a dedicated secure interface to deliver keys to the Host SoC.
Bus Master (Read) sys_cm_crdKeyConsumed	Control Input (Key Transport Core)	Control signals for KTC transfer
FPGA Power port	Power Interface	Power

Table 5 - Ports and Interfaces

All data output via the Data Output interfaces is inhibited during zeroization and pre-operational self-tests.

[1] The module does not implement Control Output interface.

4 Roles, Services, and Authentication

4.1 Roles

The module provides a role-based authentication with session management. Cryptographic Officer (with role identifier CO) role and User role (2 different users can be created, with role identifier U0 to U1) are supported and all services require an authorized role. No concurrent operators are supported.
Table 6 describes the authorized role(s) in which the service can be performed with specification of the service input parameters and associated service output parameters.

Role	Service	Input service Parameters	Output parameters
CO, U0 to U1 (User)	Login	Login initialization: Role identifier, role authentication key (public key), client nonce	CMRT nonce, session identifier
		Login finalization: signature value calculated on the role authentication key, the CMRT nonce, client nonce and role identifier.	Confirmation of login upon success signature verification
CO, U0 to U1 (User)	Logout	None	N/A
CO	Create User	Role of the user to create, hash of the role authentication key	N/A
CO	Delete User	Role of the user to delete	N/A
CO, U0 to U1 (User)	Generate Symmetric Key[2]	Key size, key ownership, name of the key	Result
CO, U0 to U1 (User)	Derive Symmetric Key (one-step/ two-steps)	PRF algorithm, shared secret[3], L, salt, label, context, name of the derived key	Derived key
CO, U0 to U1 (User)	Generate asymmetric key pair	Key size, key ownership, name of the key	Public key
CO, U0 to U1 (User)	Import Key	Wrapped key (encrypted secret), name of the key or asset to import	N/A
CO, U0 to U1 (User)	Export Key	Reference to the key to be wrapped (the asset to export), name of the key or asset to export	Wrapped key or asset (encrypted secret)
CO, U0 to U1 (User)	Key Output	KTC destination, metadata, name of the key	AES key, HMAC key, Derived key
CO, U0 to U1 (User)	AES-ECB Encrypt	Plaintext, AES key	Cyphertext
CO, U0 to U1 (User)	AES-ECB Decrypt	Cyphertext, AES key	Plaintext
CO, U0 to U1 (User)	Authenticated Encrypt	Plaintext, AES key, AAD, tag length	Cyphertext, authentication tag, IV
CO, U0 to U1 (User)	Authenticated Decrypt	Cyphertext, authentication tag, AES key, IV, AAD, tag length	Plaintext
CO, U0 to U1 (User)	AES-CBC/ CTR/ CFB128 Encryption	Plaintext, IV, AES key	Cyphertext
CO, U0 to U1 (User)	AES-CBC/ CTR/ CFB128 Decryption	Cyphertext, IV, AES key	Plaintext
CO, U0 to U1 (User)	RSA / ECDSA Sign Generation	Message, hashing algorithm, hash core selection, private key	Computed signature

[2] The generated key is stored within the module and not output as part of service
[3] The internally generated shared secret is used for approved service and the imported shared secret is used for non-approved service.

Role	Service	Input service Parameters	Output parameters
CO, U0 to U1 (User)	RSA / ECDSA Sign Verification	Signature, hashing algorithm, hash core selection, public key, message	Verification result
CO, U0 to U1 (User)	ECDSA Key Verification	ECDSA public key	N/A
CO, U0 to U1 (User)	EC Diffie-Hellman SSC	EC Diffie-Hellman private key[4], EC Diffie-Hellman public key for remote peer	Shared secret
CO, U0 to U1 (User)	EC Diffie-Hellman Key Agreement	EC Diffie-Hellman private key[4], EC Diffie-Hellman public key for remote peer	Derived key
CO, U0 to U1 (User)	MAC Generation	Message, HMAC key or AES key, MAC algorithm, hash core selection, MAC length	Authenticated message
CO, U0 to U1 (User)	MAC Verification	Authenticated message, HMAC key or AES key, MAC algorithm, hash core selection, Message	Result of verification
CO, U0 to U1 (User)	Hash	Message, hashing algorithm, hash core selection	Hashed message
CO, U0 to U1 (User)	Get TRNG	Amount of random number	Random numbers
CO, U0 to U1 (User)	List Assets	Location from which to retrieve the list of assets	List of asset names
CO, U0 to U1 (User)	Move Asset	Dynamic asset reference	Result
CO, U0 to U1 (User)	Delete Dynamic Asset	Context containing SSPs in dynamic storage	N/A
CO, U0 to U1 (User)	Delete Static Asset	Context containing SSPs in static storage	N/A
CO	Zeroize	Context containing SSPs	N/A
CO, U0 to U1 (User)	Self-Test	None	N/A
N/A	Hard Reset	None	N/A
CO, U0 to U1 (User)	Soft Reset	None	N/A
CO, U0 to U1 (User)	Show Status	None	Version information, FIPS Mode: 1
CO	DRBG	None	None

Table 6 - Roles, Service Commands, Input and Output

4.2 Authentication Methods

Except for Hard Reset, the Login service must be executed before any other CMRT services can be requested. After logging in, an operator (Crypto Officer or U0 to U1 User) may assume a different role only after logging out followed by logging back in as the different role. The process to login is divided into two main stages: the initialization stage is included in the below bullets 1,2,3; the finalization stage is included in the below bullets 4,5. Precisely, during login the following happens:

1. An entity accessing the module requests a role identifier (CO or U0 to U1 User) and provides the ECDSA P-256 public key (the role authentication key) and 128-bit nonce.
2. The module calculates SHA2-256 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

[4] Internally generated EC Diffie-Hellman private key is used for approved ECDH services and imported EC Diffie-Hellman private key is used for non-approved ECDH services.

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 using SHA2-256 on the concatenation of the role identifier, both nonces, and the role authentication 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 signature verification, the authentication succeeds. If the signature verification fails, an error is returned.

At power-on of the module, the CO is the only available role and only the CO can create up to two new users.
Authentication status for the U0 to U1 User or CO role is not maintained over the power cycle (new login is required).

Role	Authentication Method	Authentication Strength
CO	Role-Based	ECDSA P-256 is used for authentication (digital signature sign and verify) with 128 bits of security strength. The chance of a random authentication attempt falsely succeeding is 1 /2^128 which is less than the claimed strength objective of 1/1,000,000. Let's consider a failed authentication rate of 1 per 1µs and 60,000,000 consecutive attempts per minute. The probability of successful authentication is then less than or equal to 60,000,000 * 1 / 2^128(≤1.76324e-31) which is much less than the claimed false acceptance rate of 1 / 100,000 or 10e-5 within one minute.
U0 to U1 (User)	Role-Based	ECDSA P-256 is used for authentication (digital signature sign and verify) with 128 bits of security strength. The chance of a random authentication attempt falsely succeeding is 1 /2^128 which is less than the claimed strength objective of 1/1,000,000. Let's consider a failed authentication rate of 1 per 1µs and 60,000,000 consecutive attempts per minute. The probability of successful authentication is then less than or equal to 60,000,000 * 1 / 2^128(≤1.76324e-31) which is much less than the claimed false acceptance rate of 1 / 100,000 or 10e-5 within one minute.

Table 7 - Roles and Authentication

4.3 Approved Services

Table 8 lists the approved services supported by the module. Each service provides an indicator, which corresponds to the bit 31 of the service return code. For approved security services, this bit is set and the indicator sent back to the caller is 1.

Service	Description	Approved Security Functions	Keys and/or SSPs		Roles	Access rights to Keys and/or SSPs	Indicator
Login	Establish the session between the operator and the module	SHA2-256	Login initialization:	role authentication key (public key)	CO, U0 to U1 (User)	E, W	1
				hash of role authentication key		E, G	

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSPs	Indicator
		SHA2-256, ECDSA P-256	Login finalization: signature with the role authentication key		E, W	1
Logout	Close the session	None	None	CO, U0 to U1 (User)	N/A	None
Create User	User Creation	None	hash of the role authentication key	CO	W	None
Delete User	Role's entry in the root table is deleted	None	None	CO	None	None
Generate Symmetric Key	Generate a symmetric key	CKG [SP800-133Rev2]	AES key HMAC key	CO, U0 to U1 (User)	G	1
Derive Symmetric Key (one-step/ two-steps)	Derive a symmetric key of the requested length via SP800-56CRev2 KDF (one step)	SP800-56CRev2 KDF	Internally generated shared secret	CO, U0 to U1 (User)	E,W	1
			Derived key		G, R	
	Derive a symmetric key of the requested length via the two-step process described in SP800-56CRev2	SP800-56CRev2 KDF with SP800-108Rev1 KBKDF	Internally generated shared secret	CO, U0 to U1 (User)	E,W	1
			Key-derivation key		G, E, Z	
			Derived key		G, R	
Generate asymmetric key pair	Generate a key pair for a requested elliptic curve.	CKG [SP800-133Rev2]	ECDSA key pair, Internally generated EC Diffie-Hellman key pair	CO, U0 to U1 (User)	G, R (pubic key only)	1
	Generate RSA key pair in standard format[5]	CKG [SP800-133Rev2]	RSA key pair	CO, U0 to U1 (User)	G, R (pubic key only)	1
	Generate RSA key pair in CRT format	CKG [SP800-133Rev2]	RSA-CRT key pair	CO, U0 to U1 (User)	G, R (pubic key only)	1
Import Key	Import a key or asset that is wrapped via the KWP method	AES-KWP	AES key-wrapping-key	CO, U0 to U1 (User)	E	1
			Imported key or asset		W	
Export Key	Export a key or asset from static or dynamic storage	AES-KWP	AES key-wrapping-key	CO, U0 to U1 (User)	E	1
			Exported key or asset		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, Derived key	CO, U0 to U1 (User)	R	None
AES-ECB Encrypt	Executes AES-ECB mode encrypt operation	AES-ECB	AES-ECB key	CO, U0 to U1 (User)	E	1
AES-ECB Decrypt	Executes AES-ECB mode decrypt operation					
Authenticated Encrypt	Execute an AES-GCM encrypt operation	AES-GCM/ AES-GMAC (when plaintext is zero)	AES-GCM, AES-GMAC key	CO, U0 to U1 (User)	E	1
Authenticated Decrypt	Execute an AES-GCM decrypt operation	AES-GCM/ AES-GMAC (when MAC is provided as input)	AES -GCM, AES-GMAC key	CO, U0 to U1 (User)	E	1

[5] If the key generation service is requested to return prime factors then resulting key will be identified by the module as "RSA-PF" instead of just "RSA" in all other cases.

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSPs	Indicator
AES-CBC / AES-CTR / AES-CFB128 Encryption	Executes AES-CBC / AES-CTR / AES-CFB128 mode encrypt operation	AES-CBC / AES-CTR / AES-CFB128	AES-CBC / AES-CTR / AES-CFB128 key	CO, U0 to U1 (User)	E	1
AES-CBC / AES-CTR / AES-CFB128 Decryption	Executes AES-CBC / AES-CTR / AES-CFB128mode decrypt operation	AES-CBC / AES-CTR / AES-CFB128	AES-CBC / AES-CTR / AES-CFB128 key	CO, U0 to U1 (User)	E	1
RSA / ECDSA Signature Generation	Sign a message with a specified EC Diffie-Hellman/ ECDSA private key, selecting HC1 or 2	ECDSA P-224, P-256, P-384, P-521 with Hash functions listed in Table 3	ECDSA private key	CO, U0 to U1 (User)	E, W	1
	Generate a signature with PKCS#1 v2.1 PSS padding, selecting HC1 or 2	2048, 3072 or 4096 with Hash functions listed in Table 3	RSA / RSA-CRT private key			
RSA / ECDSA Signature Verification	Verify the signature of a message with a specified EC Diffie-Hellman/ ECDSA public key, selecting HC1 or 2	ECDSA P-224, P-256, P-384, P-521 with Hash functions listed in Table 3	ECDSA public key	CO, U0 to U1 (User)	E, W	1
	Verify a signature with PKCS#1 v2.1 PSS padding, selecting HC1 or HC2	2048, 3072 or 4096 with Hash functions listed in Table 3	RSA / RSA-CRT public key			
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	CO, U0 to U1 (User)	E, W	1
EC Diffie-Hellman SSC	Calculate a shared secret via the EC Diffie-Hellman algorithm	EC Diffie-Hellman P-224, P-256, P-384, P-521	Internally generated EC Diffie-Hellman private key	CO, U0 to U1 (User)	E	1
			Shared secret		G, R	
			EC Diffie-Hellman remote peer's public key		W, E	
EC Diffie-Hellman Key Agreement	Provide [SP800-56ARev3] EC Diffie-Hellman KAS Ephemeral Unified using KDF [SP800-56CRev2]	EC Diffie-Hellman P-224, P-256, P-384, P-521, [SP800-56CRev2] KDF	Internally generated EC Diffie-Hellman private key	CO, U0 to U1 (User)	E	1
			EC Diffie-Hellman remote peer's public key		W, E	
			Shared secret		G, E, Z	
			Derived key		G, R	
MAC Generation	Generate an HMAC digest using the requested SHA2 or SHA3 or AES CMAC/GMAC algorithm	HMAC functions listed in Table 3, AES-CMAC, AES-GMAC	HMAC key, AES key	CO, U0 to U1 (User)	E, W	1
MAC Verification	Verify an HMAC digest using the requested SHA2 or SHA3 or AES or CMAC/ GMAC algorithm	HMAC functions listed in Table 3, AES-CMAC, AES-GMAC	HMAC key, AES key	CO, U0 to U1 (User)	E, W	1

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSPs	Indicator
Hash	Generate a hash digest for the requested SHA2 or SHA3 algorithm.	Hash functions listed in Table 3	None	CO, U0 to U1 (User)	N/A	1
Get TRNG	Get a random number from the SP800-90ARev1 DRBG seeded with the TRNG Management Core	CTR_DRBG	Entropy Input	CO, U0 to U1 (User)	E, G	1
			Seed		E, G	
			DRBG internal state		E, G	
List Assets	List the assets in either static or dynamic asset storage	N/A	N/A	CO, U0 to U1 (User)	N/A	None
Move Asset	Move an asset from dynamic to static storage and zeroize the dynamic entry	N/A	AES key, HMAC key, RSA key pair, ECDSA key pair, EC Diffie-Hellman key pair, Derived key	CO, U0 to U1 (User)	Z	None
Delete Dynamic Asset	Delete an asset in dynamic storage by writing all 0s in each data word. Delete only if the specific User “owns” the asset.	N/A	AES key, HMAC key, RSA key pair, ECDSA key pair, EC Diffie-Hellman key pair, Shared secret, Key-derivation key, Derived key	CO, U0 to U1 (User)	Z	None
Delete Static Asset	Obliterate a static asset in OTP by writing to 1s in each data word. Delete only if the specific User “owns” the asset.	N/A	AES key, HMAC key, RSA key pair, ECDSA key pair, EC Diffie-Hellman key pair, Derived key	CO, U0 to U1 (User)	Z	None
Zeroize	Zeroize or obliterate all keys and SSPs in SRAM and /or OTP	N/A	SSPs in the OTP and/or SRAM	CO	Z	1
Self-test	Execute the RAM firmware KATs	Functions listed under FW RAM CASTs in Table 13	Keys listed under FW RAM CASTs in Table 13	CO, U0 to U1 (User)	N/A	1
Hard Reset	On demand self-test (FW integrity tests and CASTs)	N/A	N/A	N/A	N/A	None
Soft Reset	Soft Reset the CMRT	N/A	N/A	CO, U0 to U1 (User)	N/A	None
Show Status	Return the hardware version and firmware version and approved mode status	N/A	N/A	CO, U0 to U1 (User)	N/A	None
DRBG	Triggers a reseed of the SP800-90ARev1 DRBG where the entropy input is taken from the built-in Entropy Source implemented in the TRNG Management Core	CTR_DRBG	Entropy input	CO	E, G	1
			Seed		E, G	
			DRBG internal state		E, G	

Table 8 - Approved Services

G = **Generate**: The module generates or derives the SSP.
R = **Read**: The SSP is read from the module (e.g. the SSP is output).
W = **Write**: The SSP is updated, imported, or written to the module.

E = **Execute**: The module uses the SSP in performing a cryptographic operation.
Z = **Zeroise**: The module zeroises the SSP.

4.4 Non-Approved Services

Table 9 lists the non-approved services supported by the CryptoManager Root of Trust RT-660 module that can only be used in the non-Approved mode of operation. The indicator bit 31 of the service return code for non-approved services output a 0.

Service	Description	Algorithms Accessed	Roles	Indicator
Derive Symmetric Key	Derive the key from the imported shared secret	KDF (one step)	CO, U0 to U1 (User)	0
Derive Symmetric Key (two-steps)	Derive the key from the imported shared secret	KDF (two steps), KBKDF	CO, U0 to U1 (User)	0
EC Diffie-Hellman shared secret	Calculate the EC Diffie-Hellman shared secret with Imported EC Diffie-Hellman private key (P-224, P-256, P-384, P-521)	EC Diffie-Hellman algorithm	CO, U0 to U1 (User)	0
EC Diffie-Hellman key agreement	Execute the KAS-ECC scheme with Imported EC Diffie-Hellman private key (P-224, P-256, P-384, P-521), followed with a key derivation method	EC Diffie-Hellman followed by KDF one step or KDF two steps	CO, U0 to U1 (User)	0

Table 9 - Non-Approved Services

5 Software/Firmware Security

5.1 Integrity Techniques

The integrity tests (EDC and approved integrity techniques) are listed in section 10.1.1 below. Integrity tests are performed as part of the Pre-Operational Self-Tests.

5.2 Initiate on Demand

The module provides the Hard Reset service to perform self-tests on demand. Pre-Operational Self-Tests can also be performed by powering off and powering on the module.

5.3 Executable Code

Verilog RTL has been used as hardware design language for hardware components. The First Stage Bootloader (fboot), Second Stage Bootloader (sboot) and the Application Firmware are written in C language. The module includes the following executable codes:

- First-stage bootloader realized in a sea-of-gates (located in the Program ROM)
- Second-stage bootloader binary executable code (sboot) located in OTP
- Application Firmware binary executable (Supervisor Application) running in RAM

6 Operational Environment

The module operates in a non-modifiable environment and is validated at a Security Level 2 in Physical Security. Once the module is operational, it does not allow the loading of any additional software or firmware.
There are no further requirements for this security area.

7 Physical Security

The module is a sub-chip hardware module configured in a Xilinx Zynq XC7Z045 FPGA and is defined as a single-chip embodiment.
The FPGA is covered with a tampered-evident coating. The integrated heat spreader (IHS) serves as a protective shell for the processing silicon chip. The IHS lid, the substrate with solder ball grid array and the silicon chip covered with Thermal Interface material in (TMI) provide opacity in the visible spectrum.

Physical Security Mechanism	Recommended Frequency of Inspection / Test	Inspection/ Test Guidance Details
Tampered-evident coating covering the FPGA components: integrated heat spreader, substrate with solder ball grid array, silicon chip with TMI.	N/A	N/A

Table 10 - Physical Security Inspection Guidelines

8 Non-Invasive Security

Until NIST SP 800-140F that replaces the ISO/IEC 19790:2012 Annex F defines the non-invasive security mechanisms, the non-invasive mechanisms per IG 12.3 are addressed in the below section 12 "Mitigation of other attacks". The non-invasive Security area is N/A.

9 Sensitive Security Parameters Management

Key/ SSP Name /Type	Strength (bits)	Security Function / Cert. Number	Generation	Import / Export	Establishment	Storage	Zeroization	Use and related keys
AES key	128 to 256	AES-ECB, AES-CBC, AES-CTR, AES-CFB128, AES-GCM #A2114	SP800-56CRev2 KDF (one step) SP800-56CRev2 (two steps) CKG SP800-132Rev2	**Import**: Import Key **Export**: Export Key, Key Output	N/A	Static Dynamic	Delete Dynamic Asset, Delete Static Asset, Zeroize	**Use:** Generate Symmetric Key, AES-mode Encrypt/ Decrypt **Related SSPs:** DRBG internal state
		AES-GMAC AES-CMAC #A2114						**Use:** Generate Symmetric Key, Authenticated Encrypt/ Decrypt, MAC Generation, MAC Verification **Related SSPs:** DRBG internal state
AES key-wrapping-key	128 to 256	AES-KWP #A2114	SP800-56CRev2 KDF (one step) SP800-56CRev2 (two steps) CKG SP800-132Rev2	Import: Import Key Export: N/A	N/A	Static Dynamic	Delete Dynamic Asset, Delete Static Asset, Zeroize	**Use:** Generate Symmetric Key, Export Key, Import Key **Related SSPs:** DRBG internal state
HMAC key	112 to 256	HMAC #A2114 #A2115	SP800-56CRev2 KDF (one step SP800-56CRev2 (two steps) CKG SP800-132Rev2	Import: Import Key **Export**: Export Key, Key Output	N/A	Static Dynamic	Delete Dynamic Asset, Delete Static Asset, Zeroize	**Use:** Generate Symmetric Key, MAC Generation, MAC Verification, Derive Symmetric Key, Derive Symmetric Key (two-steps) **Related SSPs:** DRBG internal state
RSA key pair (public, private keys)	112 to 149	RSA, RSA-CTR #A2114 RSA #A2115	FIPS 186-4	Import: Import Key **Export**: Export Key	N/A	Static Dynamic	Delete Dynamic Asset, Delete Static, Asset Zeroize	**Use:** RSA Sign, RSA Verify, Generate RSA Key Pair **Related SSPs:** DRBG internal state
ECDSA key pair (public, private keys)	112 to 256	ECDSA #A2114 #A2115	FIPS 186-4	Import: Import Key **Export**: Export Key	N/A	Static Dynamic	Delete Dynamic Asset, Delete Static Asset, Zeroize	**Use:** ECDSA Sign, ECDSA Verify, Generate EC Key Pair **Related SSPs:** DRBG internal state

Key/ SSP Name /Type	Strength (bits)	Security Function / Cert. Number	Generation	Import / Export	Establishment	Storage	Zeroization	Use and related keys
Internally generated EC Diffie-Hellman key pair[6] (public, private keys)	112 to 256	KAS-ECC #A2114	SP800-56ARev3	Import: N/A **Export**: Export Key	N/A	Static Dynamic	Delete Dynamic Asset, Delete Static Asset, Zeroize	**Use:** EC Diffie-Hellman SSC, EC Diffie-Hellman Key Agreement **Related SSPs:** Shared secret
EC Diffie-Hellman remote peer's public key	112 to 256	KAS-ECC #A2114	N/A	**Import**: EC Diffie-Hellman SSC, EC Diffie-Hellman Key Agreement Export: N/A	N/A	N/A	N/A	**Use:** EC Diffie-Hellman SSC, EC Diffie-Hellman Key Agreement
Entropy input	256	Random number generation	Obtained from ENT (P)	N/A	N/A	Dynamic	Zeroize	**Use:** Get TRNG, DRBG **Related SSPs:** DRBG seed and internal state
DRBG internal state and seed	256	CTR_DRBG #A2114	Using SP800-90ARev1 CTR_DRBG	N/A	N/A	Dynamic	Zeroize	**Use:** Get TRNG, DRBG **Related SSPs:** Entropy input
Internally generated shared secret	112 to 256	EC Diffie-Hellman Shared Secret Computation #A2114	N/A	**Import**: Import Key **Export**: Export Key	KAS or KAS-SSC SP800-56ARev3 per IG D.F.	Dynamic	Delete Dynamic Asset, Zeroize	**Use:** Derive Symmetric Key, Derive Symmetric Key (two-steps), EC Diffie-Hellman SSC, EC Diffie-Hellman Key Agreement **Related SSPs:** EC Diffie-Hellman key pair, Derived key
Derived key	112 to 256	SP800-56ARev3 or SP800-56CRev2 KDF (one step and two steps with SP800-108Rev1) #A2114	SP800-56CRev2 (one step KDF or expansion step of two step KDF using SP800-108Rev1)	Import: N/A **Export**: Export Key, Key Output	KAS SP800-56ARev3 per IG D.F.	Static, Dynamic	Delete Dynamic Asset, Delete Static Asset, Zeroize	**Use:** EC Diffie-Hellman Key Agreement, Derive Symmetric Key, Derive Symmetric Key (two-steps) **Related SSPs:** Shared secret (one step KDF), Key-derivation key (two steps KDF)
Key-derivation key	112 to 256	SP800-56CRev2	SP800-56CRev2	Import: N/A Export: N/A	N/A.	Dynamic	Delete Dynam	**Use:** Derive Symmetric Key (two-steps)

[6] Although the module supports imported EC Diffie-Hellman key pair it is not used by approved services and hence is not considered as CSP. As listed in the table 9 EC Diffie-Hellman using imported key pair is indicated as non-approved.

Key/ SSP Name /Type	Strength (bits)	Security Function / Cert. Number	Generation	Import / Export	Establishment	Storage	Zeroization	Use and related keys
		KDF two steps #A2114	(extraction step)				ic Asset, Zeroize	**Related SSPs:** Derived key (two steps KDF), Shared secret
role authentication key (public key)	128	ECDSA #A2114 #A2115	N/A	**Import**: from calling application Export: N/A	N/A	Dynamic	Zeroize	**Use:** Login, Create User **Related SSPs:** hash of the role authentication key
hash of role authentication key	128	SHA-256 #A2114 #A2115	N/A	**Import**: during module initialization for CO, Create User for User Export: N/A	N/A	Static	Zeroize	**Use:** Login, Create User **Related SSPs:** role authentication key

Table 11 – SSPs

From above table, the module manages two different types of assets: static and dynamic. Static assets are stored in the non-volatile memory file in the OTP filesystems and dynamic asset are stored in the memory file of the SRAM filesystems. Assets are stored in plaintext within the module and protected from unauthorized disclosure and modification.

9.1 Random Bit Generators

CMRT includes a Deterministic Random Bit Generator based on the CTR_DRBG (with and without prediction resistance; with derivation function) algorithm and AES-256 as the underlying cipher according to [SP800-90ARev1]. 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 the *Generate EC Key Pair or Generate RSA Key Pair* services.
- Provide random data with the *Get TRNG* service.

CMRT includes a [SP800-90B] compliant True Random Number Generator (TRNG) to provide entropy input to the CTR_DRBG. This TRNG automatically seeds the CTR_DRBG, as part of the TRNG Management Core (TMC) driver initialization process. This process includes the gathering of enough entropy. After the initialization process, the Crypto Officer and U0 to U1 (User) can use the *Get TRNG* service to obtain random numbers from the TMC.
The CTR_DRBG provides 256 bits of security strength.
The module also performs CTR_DRBG health tests according to section 11.3 of [SP800-90ARev1].
Table 12 describes the Entropy Source available in CMRT, called ENT (P).

Entropy Source	Minimum number of bits of entropy	Details
ENT (P)	512 bits of entropy input to seed the CTR_DRBG provides 256 bits of entropy	The True Random Number Generator engine is a hardware entropy source based on eight free running ring oscillators.

Table 12 - Non-Deterministic Random Number Generation Specification

9.2 SSP Generation

The symmetric and asymmetric key generation methods (vendor affirmed) implemented in the module are compliant with [SP800-133Rev2] section 4 example 1.

- CMRT implements symmetric key generation for AES and HMAC keys using random data (Random in Generation column Table 11) obtained from the SP800-90ARev1 compliant CTR_DRBG. The method in in accordance with SP800-133r2 section 6.1 i.e. the Direct Generation of Symmetric Keys.
- RSA and Elliptic Curve DSA (ECDSA) asymmetric key pairs are generated in accordance with [SP800-133Rev2] section 5.1 i.e. which maps to FIPS 186-4. A seed (i.e. the random value) used in asymmetric key generation is obtained from the [SP800-90ARev1] CTR_DRBG.
- Elliptic Curve Diffie-Hellman key pairs are generated in accordance with [SP800-133Rev2] section 5.2 i.e. key generation method specified in [SP800-56ARev3] section 5.6.1.2. used by approved key-establishment schemes which maps to FIPS 186-4.

CMRT implements key derivation methods (*KDF* in *Generation* column Table 11, in compliance with [SP800-108Rev1] and [SP800-56CRev2] one step and two step KDF) based on a HMAC-SHA-256 counter mode PRF that uses a shared secret and key-derivation key to generate symmetric and HMAC keys. The symmetric generated key is compliant with [SP800-133Rev2] section 6.2.2. The [SP800-56CRev2] one step and two step KDF use a shared secret computed with ECDSA and EC Diffie-Hellman key pair internally generated by the module.
Intermediate key generation values are not output from the cryptographic module during or after processing the service.

9.3 SSP Entry and Output

There is no manual key import or export method used in the module.
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 [SP 800-38F]. The AES 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.

9.4 SSP Storage

Assets are stored upon creation with data structures related to asset storage location (SRAM filesystem or OTP filesystem), asset name, asset type and usage policy, asset ownership and asset data (key and SSPs). When invoking an asset creation service, the asset ownership is matching the logged-in role (CO or U0 to U1 User). If there is a

mismatch between the login role and the service requested, the service is stopped and an error is returned.
Once an 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.

9.5 SSP Establishment

CMRT provides the EC Diffie-Hellman key agreement scheme compliant with [SP800-56ARev3] and scenario 2 of IG D.F

- path (1) EC Diffie-Hellman SSC with ephemeral Unified scheme and
- path (2) EC Diffie-Hellman SSC with ephemeral Unified scheme followed by key derivation with [SP800-56CRev2] compliant using one-step or two step KDF.

The key agreement scheme provides between 112 to 256 bits of security strength. The key agreement scheme uses EC Diffie-Hellman key pairs internally generated by the module. The usage of imported EC Diffie-Hellman / ECDSA key pair is considered a non-approved service.

CMRT also provides key wrapping. As explained in section 9.3, keys can be entered in encrypted form through the key wrapping method using AES in KWP mode [SP800-38F] with keys between 128 to 256 bits, compliant with IG D.G approved Key Transport Method. SSP establishment methodology (AES key wrapping) provides between 128 and 256 bits of encryption strength.

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 EC Diffie-Hellman key size for KAS will reduce the security strength of the wrapped key or the established secret/ derived key respectively.

9.6 SSP Zeroization

A dynamic asset is deleted from the SRAM filesystem using the Delete Dynamic Asset service executed by the owner of the asset. This service can also be executed by the crypto officer on any of the assets. 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 powered 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. This service can also be executed by the crypto officer on any of the assets. 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 assets is performed by writing zeroes (in the case of dynamic assets) and ones (in the case of static assets) to the SRAM or OTP location of the asset. Zeroization of the SRAM filesystem is performed by writing zeroes to each word present in the SRAM buffer. Zeroization of the OTP filesystem and root table is performed by writing ones to each word present in the OTP storage area. The operation is performed in a time that is not sufficient to compromise SSPs. Additionally, the module processes one input message at a time: when a Zeroization or Delete Asset service is processed no other input message accessing the asset storage filesystems can be executed.
Finally, temporary SSPs generated for use during other services are zeroized when they are no longer needed, by overwriting the memory location of the SSPs with zeroes.
The zeroization indicator is provided by the Zeroize service. When the Zeroize service is called, its return value indicates the output status. If the return value is 0, the zeroization has completed successfully. Otherwise, the zeroization has not completed successfully.
When temporary SSPs are generated and zeroized during a service of this module, this is implicitly indicated by the successful completion of this service.

10 Self-Tests

10.1 Pre-Operational Self-Tests

After successful installation of the FPGA in which CMRT is configured, the CMRT automatically performs the initialization process. During initialization the pre-operational self-tests are performed 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 self-tests, services are not available and no data output is possible.
If the pre-operational self-tests succeed, then the CMRT proceeds with performing conditional algorithm self tests as specified in section 10.2.1. If the pre-operational self-tests fail, then CMRT transitions to the Error state and a corresponding error indication is given.
CMRT permits the initiation of the pre-operational or conditional self-tests on demand for periodic testing of the module. In order to perform the on demand self-tests that initiates the firmware integrity tests and KATs, the Crypto Officer shall power-off and power-on or do a hard reset of the module. The Self-tests service listed Table 8 performs all the KATs of the firmware application loaded in RAM.

10.1.1 Pre-Operational Firmware Integrity Test

The integrity tests are performed for fboot (first stage bootloader), sboot (second stage bootloader), RAM firmware image (the application firmware).
At power-on, the following happens:

1. the ROM integrity is checked automatically by a hardware 32 bit CRC
2. if the integrity check succeeds, the fboot, located in ROM, is executed and the KAT SHA2-256 is performed
3. if the KAT succeeds, the integrity of sboot, located in OTP, is checked by computing the SHA2-256 hash digest of the sboot firmware and compared to the value stored in OTP
4. if the integrity check succeeds, sboot is executed and the SHA2-256 and ECDSA KATs are performed
5. if the KATs succeed, the signature of the RAM firmware image (stored in the footer of the image) is verified using ECDSA P-256 with SHA-256
6. if this integrity test succeeds, the rest of the CASTs located in the application firmware is performed

If one of the KATs or integrity checks fail, the CMRT transitions to the Error state and a corresponding error indication is given.
SP800-90B health tests (APT and RCT) are performed at start-up on 1,024 samples and at runtime.

10.2 Conditional Self-Tests

If one of the Conditional Tests fail, the CMRT transitions to the Error state and a corresponding error indication is given.

10.2.1 Conditional Cryptographic Algorithm Self-Tests

After successful completion of the pre-operational self test, the module automatically performs all cryptographic algorithm self tests listed in the below table without any user intervention. The CASTs consist in Known Answer Tests for all the approved cryptographic

algorithms including their separate implementations (HC1 and HC2) and SP800-90ARev1, SP800-90B Health Tests for CTR_DRBG and ENT (P) respectively.

Algorithm	Conditional Algorithms Self-Tests
• Firmware ROM POST - fboot ROM KAT	
SHS	KAT SHA2-256 (prior OTP integrity test) (in Hash Core 1)
• Firmware ROM POST - sboot OTP KATs	
SHS	KAT SHA2-256 (in Hash Core 1)
ECDSA	KAT for ECDSA (NIST P-256 with SHA-256) signature verification (prior RAM integrity test) (in Hash Core 1)
• Firmware RAM Cryptographic algorithm tests	
AES	KAT AES-CBC, 128bit, 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 SHA2-256 (in Hash Core 1 and Hash Core 2) KAT SHA2-512 (in Hash Core 1 and Hash Core 2) KAT SHA3-256 (in Hash Core 1)
HMAC	KAT HMAC-SHA2-256 (in Hash Core 1 and Hash Core 2) KAT HMAC-SHA3-256 (in Hash Core 1)
RSA	KAT RSA 2048-bit with SHA-256 (PSS) signature generation (in Hash Core 1 and Hash Core 2) KAT RSA 2048-bit with SHA-256 (PSS) signature verification (in Hash Core 1 and Hash Core 2)
EC Diffie-Hellman	KAT for shared secret computation (NIST P-224)
ECDSA	KAT ECDSA (NIST P-224) with SHA-256 signature generation (in Hash Core 1 and Hash Core 2) KAT ECDSA (NIST P-224) with SHA-256 signature verification (in Hash Core 1 and Hash Core 2)
DRBG	KAT AES-CTR-256 DRBG Health test per SP800-90ARev1 section 11.3
KBKDF SP800-108Rev1 KDF	KAT SP800-108Rev1 KDF (PRF: HMAC-SHA-256 in Counter mode) (in Hash Core 1 and Hash Core 2)
SP800-56CRev2 KDF	KAT SP800-56CRev2 one-step KDF (PRF: HMAC-SHA-256) (in Hash Core 1 and Hash Core 2) KAT SP800-56CRev2 two-step KDF (PRFs: HMAC-SHA-256) (in Hash Core 1 and Hash Core 2)
ENT (P)	[SP800-90B] RCT and APT health tests

Table 13 - Conditional Algorithm Self-Tests

10.2.2 Conditional Pairwise Consistency Test

CMRT performs Pairwise Consistency Tests for generated RSA and EC keypairs. Both EC and RSA key pair generation are tested by the generation and verification of a digital signature using newly generated keys. This is compliant with IG 10.3.A Additional Comment 1.

10.3 Error States

Error State	Cause of Error	Status Indicator
Only the Show Status is available. Cryptographic functions and data output are inhibited. The only options to clear the Error state are hard reset or power-off and power-on the module.	Failure of the integrity tests	cm_sys_haltState output port set to 1 Both INTERNAL_HW_ERROR_INFO and INTERNAL_SW_ERROR_INFO registers have values different than 0x0 indicating fatal error and module not operational.
	Failure of the conditional tests (KAT and PCT)	
	Failure of SP800-90B health tests	
	Failure of the TRNG-FROs	
	Failure of the service “Self-test”	

Table 14 - Error States

From Table 14 there are two options to clear the Error state:

- Hard 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.

11 Life-Cycle Assurance

11.1 Delivery and Operation

CMRT configured 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: 950-660931-13 and FW: 951-602931-131) directly map the module HW and FW versions.

When the CMRT module is delivered as part Xilinx Zynq XC7Z045 FPGA with the above listed module versions, the signals greyed out in Figure 2 are disabled and cannot be enabled again. Customers who intend to purchase the soft IP core to be added to their own SoC should note that these signals can be enabled by updating the RTL code in order to facilitate module testing before finalizing the production version. Specifically, TRNG test, Char & Validation I/F when enabled through RTL code, can be used to allow exercising specific test features for algorithm testing and trigger error for the purposes of functional testing during FIPS validation.

11.2 Guidance Documents

Rambus provides the following documentation part of the delivered module's package:

- RT-660 CryptoManager RoT External Ref Spec [Last Revision May27.2021; 007-660130-222/914]
- CMRT RT-660 FPGA Hardware Reference Manual [Version: 1.0; Doc Number: 007-660130-412/931]
- RT-660 – FIPS 140-3 Embedded Software Architecture Specification [Version 1.02; Last Revision Novembter15, 2021; Doc Number: 005-660130-320/931]
- CMRT HLOS Programmer's Guide [007-602131-321/911 Rev B, 2021-08-31]

11.2.1 Administrator Guidance

The module is configured as a FIPS140-3 module at factory for the Xilinx Zynq XC7Z045 FPGA tested implementation. In this FPGA configuration the Crypto Officer should execute the "Show Status" service to verify:

- the hardware version output as *BUILDER_VERSION = 0x60000931*
- the firmware version output as *Sboot version: 2022-02-24-g801c166*
- the supervisor version output as *2022-02-24-g801c166*
- the module approved mode status output as "*FIPS Mode: 1*"

The hash value of the Crypto Officer’s authentication key is provisioned in the OTP root table. Once the CO is authenticated, they can call the “Create User” service to add two new Users (U1, U2).

11.2.2 Non-Administrator Guidance

The module is operating in approved mode, using the approved services listed in section 4 Table 8.

11.2.3 Rules of Operation

The Crypto Officer shall consider the following requirements and restrictions when using the module.

- AES-GCM see section 2.

12 Mitigation of Other Attacks

The module is designed to mitigate side-channel attacks which involve statistically analyzing power consumption measurements and injection of fault. The module supports Differential Power Analysis (DPA) protections and Fault Injection Attack (FIA) protections as countermeasures to mitigate those attacks.

Appendix A. Glossary and Abbreviations

AES	Advanced Encryption Standard
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CFB	Cipher Feedback
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CTR	Counter Mode
DSA	Digital Signature Algorithm
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
FIPS	Federal Information Processing Standards Publication
FSM	Finite State Model
GCM	Galois Counter Mode
HMAC	Hash Message Authentication Code
KAS	Key Agreement Scheme
KAT	Known Answer Test
KDF	Key Derivation Function
KWP	AES Key Wrap with Padding
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
OFB	Output Feedback
PR	Prediction Resistance
PSS	Probabilistic Signature Scheme
RNG	Random Number Generator
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SSP	Sensitive Security Parameter

Appendix B. References

FIPS140-3 FIPS PUB 140-3 - Security Requirements For Cryptographic Modules
March 2019
https://doi.org/10.6028/NIST.FIPS.140-3

FIPS140-3_IG Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program
https://csrc.nist.gov/Projects/cryptographic-module-validation-program/fips-140-3-ig-announcements

FIPS180-4 Secure Hash Standard (SHS)
August 2015
http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

FIPS186-4 Digital Signature Standard (DSS)
July 2013
http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

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

FIPS202 SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions
August 2015
http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf

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

RFC3394 Advanced Encryption Standard (AES) Key Wrap Algorithm
September 2002
http://www.ietf.org/rfc/rfc3394.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

SP800-38B NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication
May 2005
http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.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

SP800-56ARev3	NIST Special Publication 800-56A Revision 3 - Recommendation for PairWise Key Establishment Schemes Using Discrete Logarithm Cryptography April 2018 https://doi.org/10.6028/NIST.SP.800-56Ar3
SP800-56CRev2	Recommendation for Key Derivation through Extraction-then-Expansion August 2020 https://doi.org/10.6028/NIST.SP.800-56Cr2
SP800-90ARev1	NIST Special Publication 800-90A - Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators June 2015 http://dx.doi.org/10.6028/NIST.SP.800-90Ar1
SP800-90B	(Second DRAFT) NIST Special Publication 800-90B - Recommendation for the Entropy Sources Used for Random Bit Generation January 2016 http://dx.doi.org/10.6028/NIST.SP.800-90B
SP800-108Rev1	NIST Special Publication 800-108 - Recommendation for Key Derivation Using Pseudorandom Functions August 2022 https://doi.org/10.6028/NIST.SP.800-108r1
SP800-133r2	NIST Special Publication 800-133 Revision 2 - Recommendation for Cryptographic Key Generation June 2020 http://dx.doi.org/10.6028/NIST.SP.800-133r2
SP800-140B	NIST Special Publication 800-140B - CMVP Security Policy Requirements March 2020 http://dx.doi.org/10.6028/NIST.SP.800-140B