THALES
Building a future we can all trust

Thales CipherTrust Manager Core Security Module

NON-PROPRIETARY SECURITY POLICY

FIPS 140-2, Level 1

Document Information

Document Part Number	002-000403-001
Release Date	November 21, 2022

Revision History

Revision	Date	Reason
A	June 10, 2021	Initial version.
B	July 30, 2021	Updated following test-lab review.
C	August 2, 2021	Added ACVP certificate numbers.
D	August 6, 2021	Updated following test-lab review.
E	September 3, 2021	Updated following test-lab review.
F	September 15, 2021	Updated following test-lab review.
G	October 13, 2021	Updated following test-lab review.
H	November 9, 2021	Updated following test-lab review.
J	June 20, 2022	• Updated for consistency with module firmware version 1.0.3. • Updated CAVP references for Alt library. • Updated for consistency with Entropy documentation.
K	August 24, 2022	• Updated Table 2-2 to note CVLs. • Updated Sections 2.5 and 10.3 to clarify the FIPS mode of operation.
L	November 21, 2022	Updated Section 10.3 to clarify that there is a non-approved mode of operation.

Trademarks, Copyrights, and Third-Party Software

Disclaimer

All information herein is either public information or is the property of and owned solely by Thales, and/or its subsidiaries who shall have and keep the sole right to file patent applications or any other kind of intellectual property protection in connection with such information.

Nothing herein shall be construed as implying or granting to you any rights, by license, grant or otherwise, under any intellectual and/or industrial property rights of or concerning any of Thales's information.

This document can be used for informational, non-commercial, internal and personal use only provided that:

> The copyright notice below, the confidentiality and proprietary legend and this full warning notice appear in all copies.

> This document shall not be posted on any network computer or broadcast in any media other than on the NIST CMVP validation list and no modification of any part of this document shall be made.

Use for any other purpose is expressly prohibited and may result in severe civil and criminal liabilities.

The information contained in this document is provided "AS IS" without any warranty of any kind. Unless otherwise expressly agreed in writing, Thales makes no warranty as to the value or accuracy of information contained herein.

Thales hereby disclaims all warranties and conditions with regard to the information contained herein, including all implied warranties of merchantability, fitness for a particular purpose, title and non-infringement. In no event shall Thales be liable, whether in contract, tort or otherwise, for any indirect, special or consequential damages or any damages whatsoever including but not limited to damages resulting from loss of use, data, profits, revenues, or customers, arising out of or in connection with the use or performance of information contained in this document.

Thales does not and shall not warrant that this product will be resistant to all possible attacks and shall not incur, and disclaims, any liability in this respect. Even if each product is compliant with current security standards in force on the date of their design, security mechanisms' resistance necessarily evolves according to the state of the art in security and notably under the emergence of new attacks. Under no circumstances, shall Thales be held liable for any third party actions and in particular in case of any successful attack against systems or equipment incorporating Thales products. Thales disclaims any liability with respect to security for direct, indirect, incidental or consequential damages that result from any use of its products. It is further stressed that independent testing and verification by the person using the product is particularly encouraged, especially in any application in which defective, incorrect or insecure functioning could result in damage to persons or property, denial of service or loss of privacy.

CONTENTS

ACRONYMS AND ABBREVIATIONS

Term	Definition
AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard – New Instructions
AIC	Advanced Industrial Computers
ANSI	American National Standards Institute
API	Application Programming Interface
ASN	Advanced Shipping Notification
CA	Certificate Authority
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CMVP	Cryptographic Module Validation Program
CPU	Central Processing Unit
CSR	Certificate Signing Request
CSP	Critical Security Parameter
CTR	CounTeR
CVL	Component Validation List
DH	Diffie-Hellman
DRAM	Dynamic Random Access Memory
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
FIPS	Federal Information Processing Standard

Term	Definition
GCM	Galois Counter Mode
HDD	Hard Disk Drive
HKDF	HMAC-based Key Derivation Function
HMAC	Keyed-Hash Message Authentication Code
IG	Implementation Guidance
I/O	Input/Output
IV	Initialization Vector
JSON	JavaScript Object Notation
JWE	JSON Web Encryption
JWT	JSON Web Token
KAS	Key Agreement Scheme
KAT	Known Answer Test
KDF	Key Derivation Function
KEK	Key Encryption Key
KMIP	Key Management Interoperability Protocol
KTS	Key Transport Scheme
KW	Key Wrap mode
KWP	Key Wrap with Padding mode
MAC	Message Authentication Code
MKEK	Master Key Encryption Key
NAE	Network Attached Encryption
NIST	National Institute of Science and Technology
N/A	Not Applicable
OE	Operational Environment
PAA	Processor Algorithm Accelerator
PBKDF	Password Based Key Derivation Function

Term	Definition
PFS	Perfect Forward Secrecy
PKCS	Public-Key Cryptography Standards
POST	Power-On Self-Test
PSK	Pre-Shared Key
PSK-DHE	Pre-Shared Key – Diffie-Hellman Ephemeral
PSS	Probabilistic Signature Scheme
PWD	Password
REST	Representational State Transfer (web services using REST architectural principals are known identified as being RESTful).
RFC	Request for Comments
RNG	Random Number Generator
RSA	Rivest Shamir Adleman
SHA	Secure Hash Algorithm
SSC	Shared Secret Computation
SSH	Secure Shell
TDES	Triple – Data Encryption Standard
TLS	Transport Layer Security
URL	Uniform Resource Locator
XML	eXtensible Markup Language

REFERENCES

[FIPS 140-2] Federal Information Processing Standards Publication (FIPS PUB) 140-2, 'Security Requirements for Cryptographic Modules', May 25, 2001 (including change notices 12-02-2002).

[FIPS 140-2 IG] NIST, Implementation Guidance for FIPS 140-2 and the Cryptographic Module Validation Program, March 14, 2022.

[FIPS 180-4] Federal Information Processing Standards Publication 180-4, Secure Hash Standard (SHS), NIST, August 2015.

[FIPS 186-4] Federal Information Processing Standards Publication 186-4, Digital Signature Standards (DSS), NIST, July 2013.

[FIPS 197] Federal Information Processing Standards Publication 197, Specification for the Advanced Encryption Standard (AES), November 26, 2001.

[FIPS 202] Federal Information Processing Standards Publication 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, August 2015.

[FIPS 198-1] Federal Information Processing Standards Publication 198-1, The Keyed-Hash Message Authentication Code (HMAC), July 2008.

[SP800-38A] NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation – Methods and Techniques, Morris Dworkin, December 2001.

[SP800-38B] NIST Special Publication 800-38B, Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication, May 2005 (with October 2016 updates).

[SP800-38D] NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, November 2007.

[SP800-38F] NIST Special Publication 800-38F, Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping, December 2012.

[SP800-52r2] NIST Special Publication 800-52 Rev 2, Guidelines for the Selection, Configuration, and Use of Transport Lauer Security (TLS) Implementations, August 2019.

[SP800-56Ar3] NIST Special Publication 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, Revision 3, April 2018.

[SP800-56Br2] NIST Special Publication 800-56B, Recommendation for Pair-Wise Key-Establishment Schemes Using Integer Factorization Cryptography, Revision 2, March 2019.

[SP800-56Cr2] NIST Special Publication 800-56C, Recommendation for Key-Derivation Methods in Key-Establishment Schemes, Revision 1, April 2018.

[SP800-67r2] NIST Special Publication 800-67, Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, Rev 1, January 2012.

[SP800-90Ar1] NIST Special Publication SP800-90A, Recommendation for Random Number Generation Using Deterministic Bit Generators, Rev1, June 2015.

[SP800-90B] NIST, SP800-90B, “Recommendation for the Entropy Sources Used for Random Bit Generation”, Version 1.0, January 2018.

[SP800-131Ar2] NIST Special Publication 800-131A revision 2, Transitioning the Use of Cryptographic Algorithms and Key Lengths, March 2019.

[SP800-132] NIST Special Publication 800-132, Recommendation for Password-Based Key Derivation: Part 1: Storage Applications, December 2010.

[SP800-133] NIST Special Publication 800-133 revision 2, Recommendation for Cryptographic Key Generation, June 2020.

[SP800-135r1] NIST Special Publication 800-135, Recommendation for Existing Application-Specific Key Derivation Functions, December 2011.

[PKCS #1] PKCS #1: RSA Cryptographic Standard, RSA Laboratories, v2.1.

[RFC5246] RFC 5246, The Transport Layer Security (TLS) Protocol, Version 1.2, August 2008.

[RFC5288] RFC 5288, AES Galois Counter Mode (GCM) Cipher Suites for TLS, August 2008.

[RFC5639] Lochter M, Merkle J, ‘Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation’, Internet Engineering Task Force, RFC 5639, March 2010.

[RFC7516] RFC 7516, JSON Web Encryption (JWE), May 2015.

[RFC8446] RFC 8446, The Transport Layer Security (TLS) Protocol Version 1.3, ISSN: 2070-1721, August 2018.

[SEC 2] Certicom Research, ‘Standards for Efficient Cryptography - SEC2: Recommended Elliptic Curve Domain Parameters’, Version 2.0, January 27, 2010.

[KMIP 2.1] Key Management Interoperability Protocol Specification Version 2.1, Committee Specification 01, 07 May 2020.

PREFACE

This document deals only with operations and capabilities of the Thales CipherTrust Manager Core Security Module in the technical terms of [FIPS 140-2].

General information on Thales products is available from the following sources:

- the Thales internet site contains information on the full line of available products at https://cpl.thalesgroup.com
- product updates and technical support literature is available from the Thales Customer Support Portal at https://supportportal.thalesgroup.com/csm
- online manuals for the product can be found at https://www.thalesdocs.com/ctp/cm/2.4/
- technical or sales representatives of Thales can be contacted through one of the channels listed on https://cpl.thalesgroup.com/contact-us

> NOTE: You require an account to access the Customer Support Portal. To create a new account, go to the portal and click on the **REGISTER** link.

1 General

1.1 Security Level

The cryptographic module meets all Level 1 security requirements for [FIPS 140-2] as summarized in the table below:

Table 1-1: FIPS 140-2 Security Levels

Security Requirements Section	Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles and Services and Authentication	1
Finite State Machine Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	3
Mitigation of Other Attacks	N/A
Cryptographic Module Security Policy	1

2 Cryptographic Module Specification

2.1 Module Overview

The Thales CipherTrust Manager Core Security Module is module type: **software module,** with the embodiment: **multi-chip standalone**. The module is a sub-component in the Thales CipherTrust Manager product.

The cryptographic boundary of the module encompasses elements of Thales CipherTrust Manager, which provide cryptographic and key management services either internally to the CipherTrust Manager product or to clients, which are employing services offered to the CipherTrust Manager via its external interfaces.

The module provides secure key generation and protection for symmetric keys and asymmetric key pairs along with support for a broad range of other cryptographic services.

Keys are stored outside the logical boundary of the module but within the physical boundary. Memory management is completely handled by the operating system

Access to services offered by Thales CipherTrust Manager Core Security Module is exclusively through a number of Application Programming Interfaces (API) offered by the Thales CipherTrust Manager Core Security Module. These API can be accessed by other applications running internal to the physical boundary of the module or, in some instances, can be accessed by remote client over dedicated TLS tunnels.

The module supports the option to configure use of a separate module as a 'Root-of-Trust'. When configured, the Root-of-Trust can optionally be used to seed the noise source for the module DRBG.

2.2 Module Description

Thales CipherTrust Manager Core Security Module is a software module designed to execute on a general purpose computer hardware platform. It may either reside on a Thales supplied physical host or on a user supplied equivalent.

The module consists of a suite of six binary files that are bound and launched together to form the module. Each binary is responsible for running its own firmware integrity test on launch and where a single binary is responsible for launching the other five binary and for ensuring that all power-on test have completed ahead of the module being available for external use.

Each binary contains code required for it to complete its role in the module alongside a local copy of one or two cryptographic libraries depending on the cryptography it is locally required to perform. Known-Answer Test (KAT) are run at power-on for each binary on its own copy of the relevant cryptographic libraries.

The module supports a single Deterministic Random Bit Generator (DRBG) instance accessed and used by each of the binaries.

Each binary communicates with other binaries forming the module over shared sockets and using REST calls. Should any of the binaries stop as a result of an error following startup, all binaries are stopped and where recovery requires a repeat of the start-up process for the module.

The module is accessible for use either using a series of externally available REST commands on its RestAPI or can be accessed remotely where one binary supports the use of TLS for applications running external to the physical boundary of the module to access the module. The TLS tunnel supports the use of the Key Management Interoperability Protocol (KMIP) or Network Attached Encryption XML (NAE-XML) to interface with the module.

In the case of remote access, all management of the network interface for the GPC (up until the passage of raw TLS data received on target sockets to the module) is performed by the operating environment and in particular the host OS.

Thales offers two variants of physical hosts in the form of the CipherTrust Manager K470 and the CipherTrust K570 appliances. The two platforms are nearly identical with the K570 containing an embedded Thales Luna PCIe HSM not present on the K470 platform.

If the module is deployed as a virtual appliance on a user supplied host, the Operational Environment (OE) includes an additional layer in the software stack where VMware ESXi is used to virtualize all software running on the appliance (including the Thales CipherTrust Manager Core Security Module).

Thales supplied physical embodiments of the module are shown below:

Figure 2-1: Thales CipherTrust Manager K470

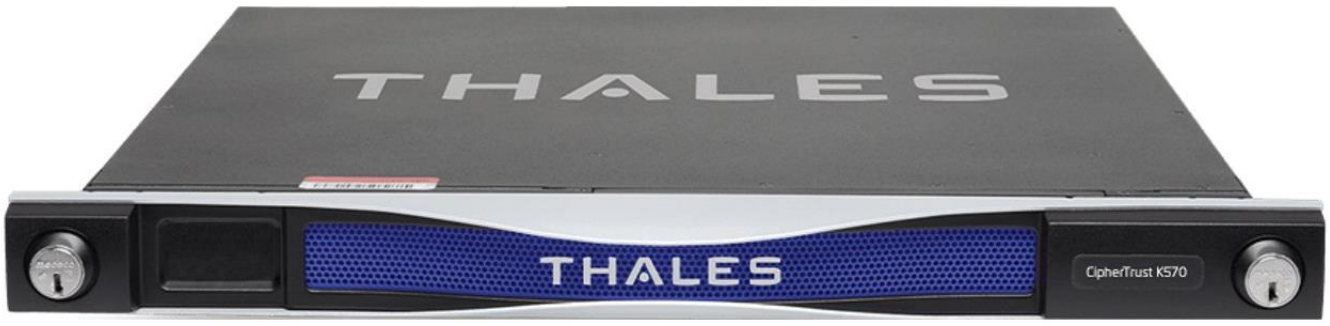


Figure 2-2: Thales CipherTrust Manager K570

The physical cryptographic boundary for the module is defined as the general-purpose computer on which the module is run. Physical interfaces are as supported by the motherboard. A block diagram of the physical operating environment for the module including the physical cryptographic boundary is shown below:

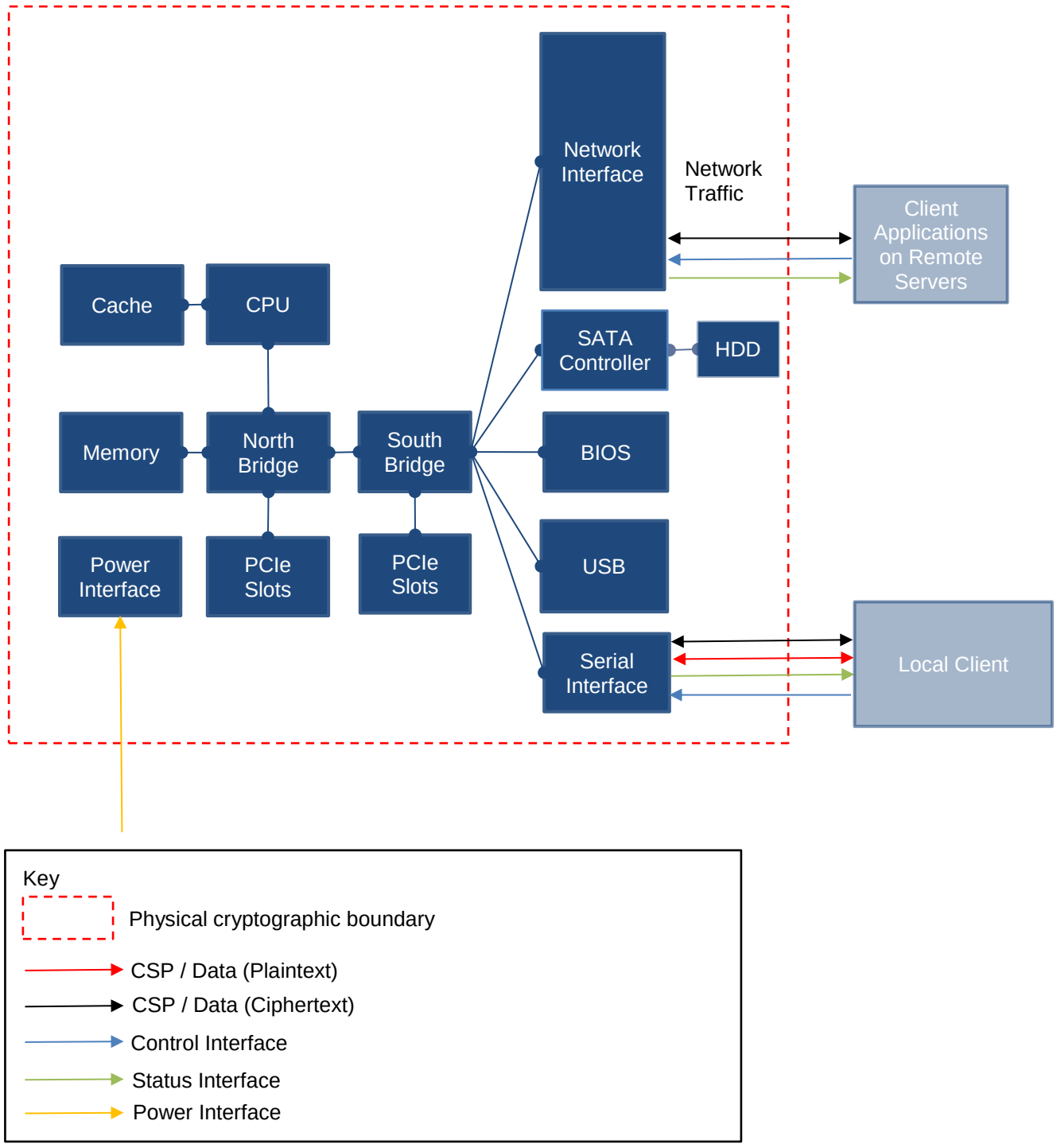


Figure 2-3: Physical Module Cryptographic Boundary

The logical cryptographic boundary of the software module consists of the Thales CipherTrust Manager Core Security Module as shown the following figure:

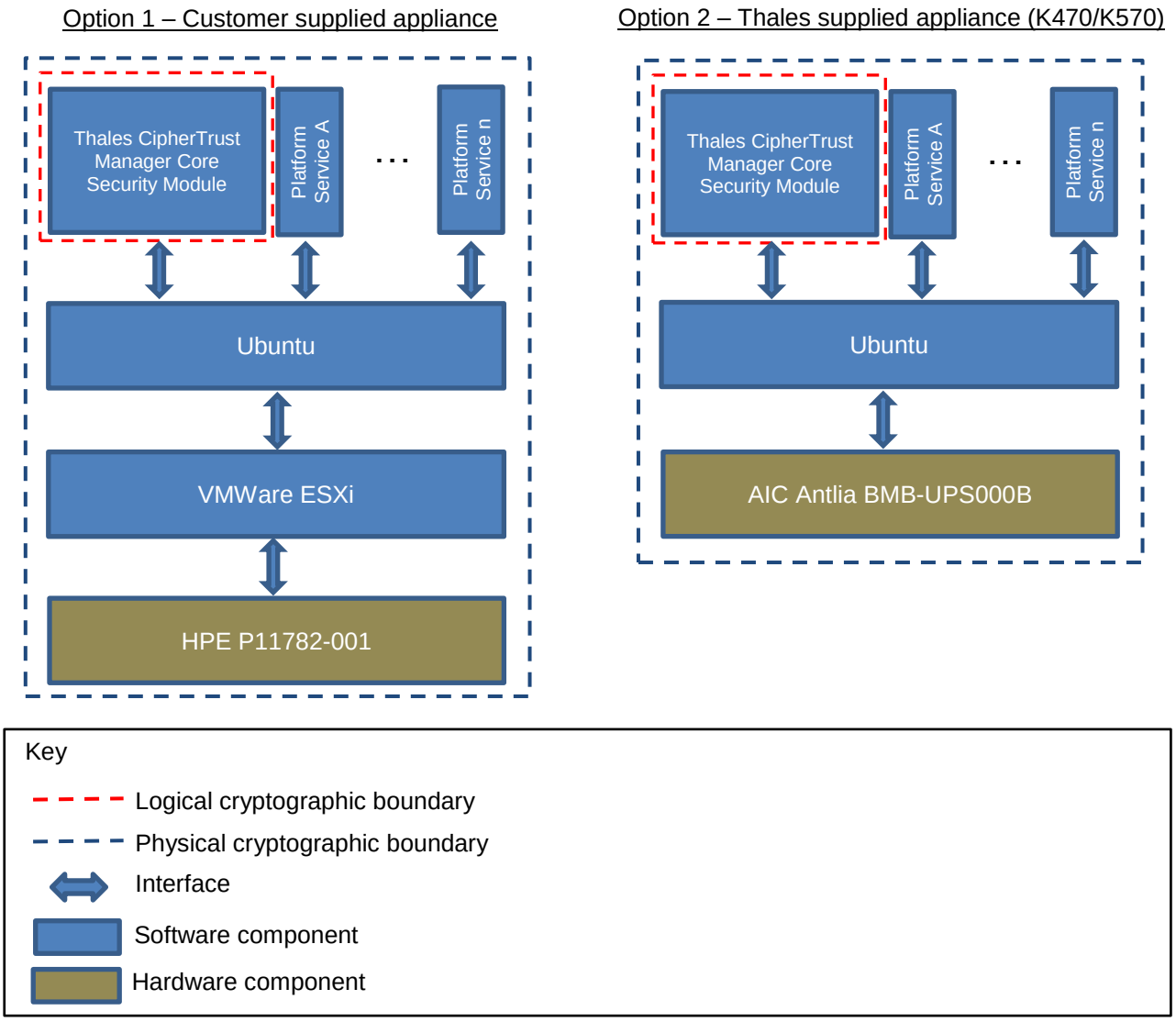


Figure 2-4: Logical Cryptographic Boundary.

The logical boundary of the module consists of the following files with each file being a linux executable:

> sheepdog, darkstar, enigma, minerva, nae and sallyport.

Each file has a corresponding signature file[1] that contains its signature as performed as part of the modules firmware integrity test.

The structure of the Thales CipherTrust Manager Core Security Module as split between the six binaries including the mapping of interfaces to each binary is shown in the following figure:

[1] Signature files for the module are: sheepdog.signature.json, darkstar.signature.json, sallyport.signature.json, minerva.signature.json, enigma.signature.json and nae.signature.json.

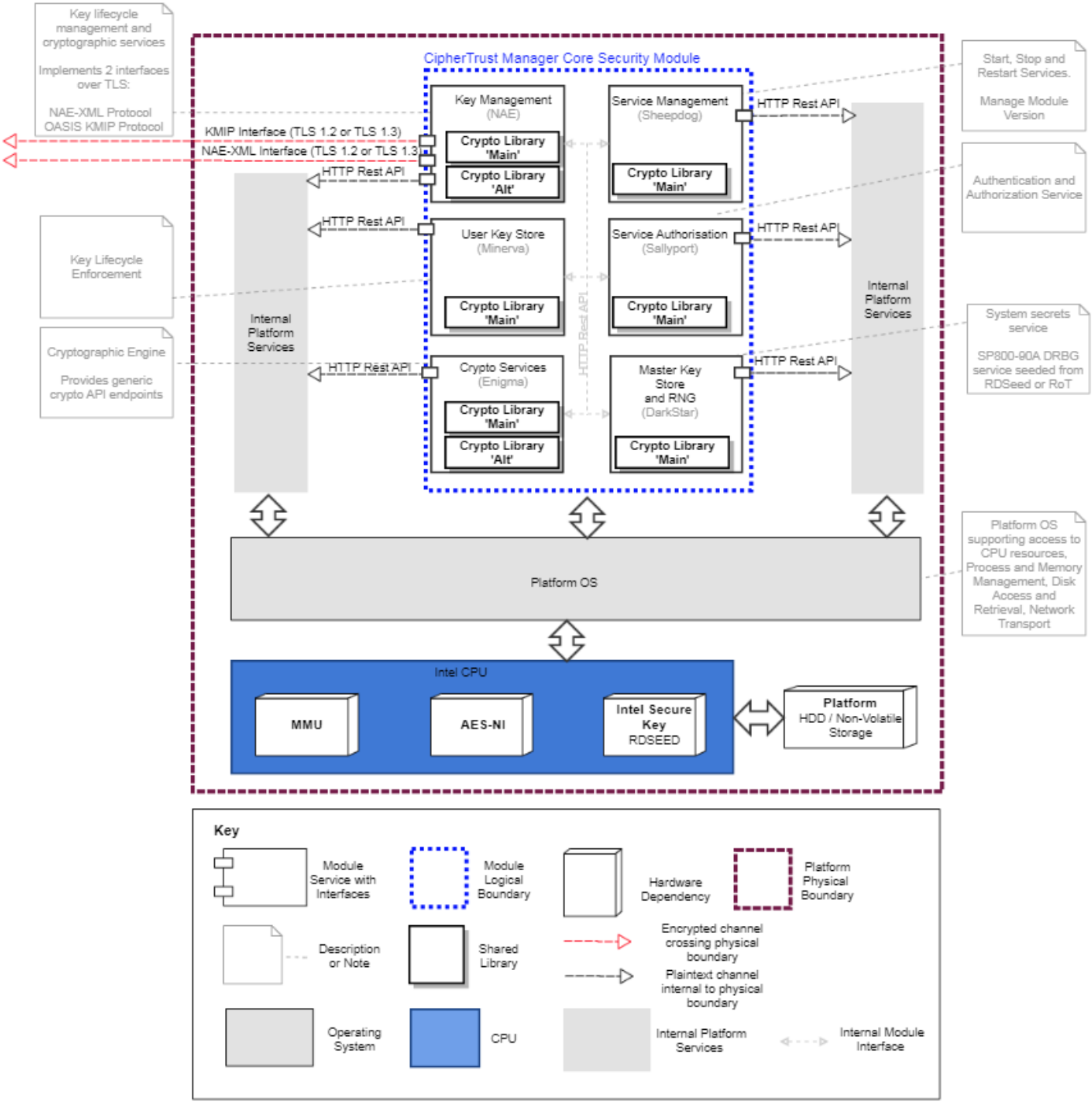


Figure 2-5: CipherTrust Core Security Manager – sub-systems, interfaces and hardware dependencies.

The module is only able to enter its approved mode of operation when the self-contained services supported across these files are all operational and where each of the binaries has independently completed its integrity check and Power-on Self-Test (POST).

CSP are stored outside the logical boundary of the module but within the physical boundary. Memory management is completely handled by the operating system.

The module relies on the operating environment to provide separation of memory used to execute each binary from other services running on the CipherTrust Manager appliance.

2.3 Tested Configurations

The following tested configurations are covered in this security policy:

Table 2-1: Cryptographic module tested configuration.

Platform	Hardware Identifier	Operating System	Software Version	Distinguishing Features
General purpose x86 based server (AIC Antlia BMB-UPS0000B).	BMB-UPS0000B (includes: Intel® XEON® E3 1275 v6 CPU).	Ubuntu Version: 18.04.	Thales CipherTrust Manager Core Security Module, Software Version: 1.0.3.	Standard configuration typically used with the Thales supplied K470 or K570 host.
General purpose x86 based server (HPE P11782-001).	P11782-001 (includes: Intel® XEON® Gold 6252 CPU).	Ubuntu Version: 18.04. VMWare ESXi Version: 6.5.	Thales CipherTrust Manager Core Security Module, Software Version: 1.0.3.	Virtualized configuration used with customer supplied host.

2.4 Approved Algorithms

The following software cryptographic libraries and associated CAVP certificates are used by the cryptographic module:

> **CipherTrust Manager Core Library Main (PAA off)** (Cert #A1779);
  - Supported Algorithms: AES, Triple-DES, SHA2, SHA3, ECDSA, RSA, HMAC, DRBG, KAS-ECC-SSC, KDF (OneStep, HKDF, TLS 1.2, TLS 1.3 and X9.63), KTS-RSA, PBKDF.
> **CipherTrust Manager Core Library Main (PAA on)** (Cert #A1778);
  - Supported Algorithms: AES only.
> **CipherTrust Manager Core Library Alt (PAA off)** (Cert #A2634);
  - Supported Algorithms: AES, Triple-DES, SHA, ECDSA, HMAC, KAS-ECC-SSC.
> **CipherTrust Manager Core Library Alt (PAA on)** (Cert #A2635).
  - Supported Algorithms: AES only.

NOTE The modules cryptographic libraries include PAA support for AES-NI when loaded on compatible Intel® CPU and have been tested with both AES-NI enabled and disabled.

Libraries have been tested on both configurations of the module as covered in section 2.3, 'Tested Configuration'.

NOTE In addition to the software cryptographic libraries, the modules uses a hardware implementation of the CBC-MAC conditioning function covered under Cert #A1791 for the Intel® Xeon® Gold 6252 Processor and Cert #A2206 for the Intel® Kaby Lake E3-1275 V6 Processor.

The approved algorithms implemented by the module with their mapping to the certificates above and algorithms use by service are listed in the Table 2-2.

Non-approved algorithms allowed in the approved mode of operation are covered in Table 2-3, 'Non-approved algorithms allowed in the approved mode of operation'.

Table 2-2: Approved Algorithms

CAVP Cert	Algorithm and Standard	Mode / Methods	Description / Key Size(s) / Key Strengths	Use / Function
Asymmetric Cryptography				
#A1779.	**Algorithm:** RSA. **Standard:** [FIPS 186-4].	**Method:** Key Generation, Signature Generation, Signature Verification. **Signature Type:** PKCS #1-v1.5 1.5, PKCS-PSS. Key Generation: Probable Primes ([FIPS 186-4], Appendix B.3.3). **Hash options:** Signature Generation (PKCS #1-v1.5 and PKCS-PSS): SHA2-256, SHA2-384, SHA2-512. Signature Verification (PKCS #1-v1.5 and PKCS-PSS): SHA-1, SHA2-256, SHA2-384, SHA2-512.	Modulus length: 2048, 3072, 4096 – Key Generation, Signature Generation and Signature Verification. 1024 – Signature Verification only.	Used to support the following services: > Generate User Key; > Sign Data; > Signature Verify; > Client to Module Trusted Channel; and > Issue / validate Internal JWT. In addition as an internal module function, RSA is used to support software integrity during power-on self-test.
#A1779.	**Algorithm:** ECDSA. **Standard:** [FIPS 186-4].	**Methods:** Key Generation, Signature Generation, Signature Verification. **Hash options (Signature Generation / Signature Verification):** SHA2-224, SHA2-256, SHA2-384, SHA2-512. **Hash options (Signature Verification):** SHA1.	**Curves:** P-224, P-256, P-384, P-521.	Used to support the following services: > Generate System Key; > Generate User Key; > Issue / validate Internal JWT; and > Client to Module Trusted Channel.

CAVP Cert	Algorithm and Standard	Mode / Methods	Description / Key Size(s) / Key Strengths	Use / Function
#A2634.	**Algorithm:** ECDSA. **Standard:** [FIPS 186-4].	**Methods:** Key Generation, Signature Generation, Signature Verification. Hash options: SHA1, SHA2-224, SHA2-256, SHA2-384, SHA2-512.	**Curves:** P-224, P-256, P-384, P-521.	Used to support the following services: > Wrap/Unwrap User Key > Sign Data; and > Signature Verify.
Hashing				
#A1779.	**Algorithm:** SHA. **Standards:** [FIPS 186-4] and [FIPS 202].	**Methods:** SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512.	N/A.	Used to support the following services: > Generate User Key; > Derive Key; > Wrap/Unwrap User Key; > Sign Data; > Signature Verify; > MAC Generate; > MAC Verify; > Secure MKEK Distribution; > Issue / validate Internal JWT; and > Client to Module Trusted Channel. In addition as an internal module function, SHA2-512 is used to support software integrity during power-on self-test.
#A2634.	**Algorithm:** SHA. **Standards:** [FIPS 186-4] and [FIPS 202].	**Methods:** SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512.	N/A.	Used to support the following services: > Encrypt Data; and > Decrypt Data.

CAVP Cert	Algorithm and Standard	Mode / Methods	Description / Key Size(s) / Key Strengths	Use / Function
Message Authentication Code				
#A1779.	**Algorithm:** HMAC. **Standard:** [FIPS 198-1].	**Methods:** HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512.	**Mac Sizes:** 10-to-64 bytes (dependent on hash). **Key Sizes:** key size < block size, key size = block size, key size > block size.	Used to support the following services: > Secure MKEK Distribution; > Issue / validate Internal JWT; > MAC Generate; > MAC Verify; and > Get Entropy.
#A2634.	**Algorithm:** HMAC. **Standard:** [FIPS 198-1].	**Methods:** HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512.	**Mac Sizes:** 10-to-64 bytes (dependent on hash). **Key Sizes:** key size < block size, key size = block size, key size > block size.	Used to support the following service: > Wrap/Unwrap User Key.
Symmetric				
#A1779, #A1778.	**Algorithm:** AES. **Standards:** [FIPS 197], [SP800-38A], [SP800-38D] and [SP800-38F].	**Mode:** CBC, CTR, ECB, GCM[2], KW, KWP.	**Key size:** 128, 192 and 256-bits.	Used to support the following services: > Secure MKEK Distribution; > Wrap/Unwrap User Key; > Client to Module Trusted Channel; > Encrypt Data (NAE-XML and KMIP API); and > Decrypt Data (NAE-XML and KMIP API). In addition, as an internal module function GCM mode with 256-bit key is used to support storage encryption of keys.

[2] details on supported IV generation methods is provided in the note boxes following Table 2-2.

CAVP Cert	Algorithm and Standard	Mode / Methods	Description / Key Size(s) / Key Strengths	Use / Function
#A2634, #A2635.	**Algorithm:** AES. **Standards:** [FIPS 197], [SP800-38A] and [SP800-38D].	**Mode:** CBC, ECB, GCM[3].	**Key size:** 128, 192 and 256-bits.	Used to support the following services: > Encrypt Data (REST API); > Decrypt Data (REST API).
#A1779.	**Algorithm:** Triple-DES. **Standards:** [SP800-67r2] and [SP800-38A].	**Mode:** CBC, ECB.	**Key size:** 162-bits (3-key).	Used to support the following services: > Encrypt Data (NAE-XML and KMIP API); and > Decrypt Data (NAE-XML and KMIP API). Further guidance on maintaining a user enforced block count for Triple-DES is provided in section 10.6, 'Security Rules'. **This algorithm is only permitted for use until Dec 31st 2023 after which point it is disallowed for all new encryption operations based on planned algorithm transitions outlined in [SP800-131Ar2].**
#A2634.	**Algorithm:** Triple-DES. **Standards:** [SP800-67r2] and [SP800-38A].	**Mode:** CBC, ECB.	**Key size:** 162-bits (3-key).	Used to support the following services: > Encrypt Data (REST API); > Decrypt Data (REST API). Further guidance on maintaining a user enforced block count for Triple-DES is provided in section 10.6, 'Security Rules'. **This algorithm is only permitted for use until Dec 31st 2023 after which point it is disallowed for all new encryption operations based on planned algorithm transitions outlined in [SP800-131Ar2].**

[3] details on supported IV generation methods is provided in the note boxes following Table 2-2.

CAVP Cert	Algorithm and Standard	Mode / Methods	Description / Key Size(s) / Key Strengths	Use / Function
Key Agreement Scheme				
#A1779.	**Algorithm:** KAS-ECC-SSC. **Standards:** [SP800-56Ar3].	**Methods:** ephemeralUnified, onePassDH. **Role:** Initiator and Responder. **Hash Function:** SHA2-224, SHA2-256, SHA2-384, SHA2-512.	**Curves:** P-224, P-256, P-384, P-521.	Outputs from this function are used to feed approved KDF to derive encryption and MAC keys used to support the following services: > Secure MKEK Distribution; > Client to Module Trusted Channel; > Wrap/Unwrap User Key. Further details on the specific cryptography used can be found in the Table 4-3, 'Services' from section 4.3, 'Services'.
#A2634.	**Algorithm:** KAS-ECC-SSC. **Standards:** [SP800-56Ar3].	**Methods:** onePassDH. **Role:** Initiator and Responder. **Hash Function:** SHA2-224, SHA2-256, SHA2-384, SHA2-512.	**Curves:** P-224, P-256, P-384, P-521.	Outputs from this function are used to feed approved KDF to derive encryption keys used to support the following services: > Encrypt Data; and > Decrypt Data. Further details on the specific cryptography used can be found in the Table 4-3, 'Services' from section 4.3, 'Services'.
Key Transport				
#A1779.	**Algorithm:** KTS-RSA. **Standards:** [SP800-56Br2] and [SP800-56Cr2].	**Method:** KTS-OAEP-basic. **Key generation method:** rsakpg1-basic and rsakpg1-prime-factor. **KTS method hash algorithm:** SHA2-256, SHA2-384, SHA2-512.	**Modulus length:** 2048, 3072, 4096. **Caveat:** key establishment methodology provides between 112 and 150 bits of encryption strength.	Used to support the following service: > Wrap/Unwrap User Key.

CAVP Cert	Algorithm and Standard	Mode / Methods	Description / Key Size(s) / Key Strengths	Use / Function
#A1779 #A1778.	**Algorithm:** KTS (AES Certs. #A1779, #A1778). **Standards:** [FIPS 197] and [SP800-38F].	**Modes:** KW, KWP and GCM[4].	**Key size:** 128, 192, 256-bits. **Caveat:** key establishment methodology provides between 128 and 256 bit of encryption strength.	Used to support the following service: > Wrap/Unwrap User Key.
#A1779, #A1778.	**Algorithm:** KTS (AES Certs. #A1779, #A1778, HMAC Certs #A1779). **Standards:** [FIPS 197] and [SP800-38F].	**Modes:** CTR. **MAC options:** HMAC-SHA2-256.	**Key size:** 128. **Caveat:** key establishment methodology provides 128 bits of encryption strength.	Used to support the following service: > Secure MKEK Distribution.
#A2634, #A2635.	**Algorithm:** KTS (AES Certs. #A2634, #A2635, HMAC Certs #A2634). **Standards:** [FIPS 197] and [SP800-38F].	**Modes:** CBC. **MAC options:** HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512.	**Key size:** 128, 192, 256-bits. **Caveat:** key establishment methodology provides between 128 and 256 bit of encryption strength.	Used to support the following service: > Wrap/Unwrap User Key.

[4] details on supported IV generation methods is provided in the note boxes following Table 2-2.

CAVP Cert	Algorithm and Standard	Mode / Methods	Description / Key Size(s) / Key Strengths	Use / Function
Key Derivation Function				
#A1779.	**Algorithm:** PBKDF. **Standards:** [SP800-132].	**Methods:** HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512, HMAC-SHA2-512/224, HMAC-SHA2-512/256, HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512.	**Derived Key Length:** 112 to 4096-bits. **Password Length:** 8 to 128-bytes. **Salt Length:** 128 to 4096-bits.	Used as an internal module service to derive the storage key for MKEK from MKEK PWD when a separate root-of-trust is not configured. Available as a key derivation option with Wrap/Unwrap User Key but where this may exclusively be used to wrap keys for storage by the end-user and not for transport. Further guidance on permitted use of PBKDF is provided in section 10.6, 'Security Rules'.
#A1779.	**Algorithm:** TLS KDF (CVL). **Standards:** [SP800-135r1].	**Methods:** TLS 1.2 KDF, TLS 1.3 KDF **PRF (TLS 1.2 KDF):** SHA2-256, SHA2-384. **Hash (TLS 1.3 KDF):** SHA2-256, SHA2-384. Running Modes (TLS 1.3 KDF): DHE, PSK, PSK-DHE.	N/A.	Used to support the following service: > Client to Module Trusted Channel.
#A1779.	**Algorithm:** KDA. **Standards:** [SP800-56Cr2].	**Method:** OneStep KDF. **Hash:** SHA2-256.	**Shared secret length:** 224-65536, increment 8 bits. **Derived Key length:** 1024 bits **MAC salt methods (HKDF only):** default, random.	Used to support the following service: > Secure MKEK Distribution.
#A1779.	**Algorithm:** KDA. **Standards:** [SP800-56Cr2].	**Method:** HKDF **Hash:** SHA1, SHA2-224, SHA2-256, SHA2-384, SHA2-512.	**Shared secret length:** 224-65536, increment 8 bits. **Derived Key length:** 1024 bits **MAC salt methods (HKDF only):** default, random.	Used to support the following services: > Wrap/Unwrap User Key; and > Derive Key.

CAVP Cert	Algorithm and Standard	Mode / Methods	Description / Key Size(s) / Key Strengths	Use / Function
#A1779.	**Algorithm:** X9.63 KDF (CVL). **Standards:** [SP800-135r1].	**Methods:** SHA2-224, SHA2-256, SHA2-384, SHA2-512.	**Shared secret length:** 256-1024 bits.	Used to support the following service: > Wrap/Unwrap User Key.
Random Number Generation				
N/A	**Algorithm:** ENT (P). **Standards:** [SP800-90B].	**Methods:** Live noise source with CBC-MAC with AES-128 as the vetted conditioning function (this is defined in Appendix F to [SP800-90B]).	**Security Strength:** Full Entropy.	Used as an internal module service to seed the platform DRBG (HMAC_DRBG) when the Intel® secure key RDSEED is selected as the entropy source during secure initialization as covered in section 10.3, Approved Mode of Operation.
#A1791, #A2206.	**Algorithm:** Conditioning component AES-CBC-MAC SP800-90B. **Standards:** [SP800-90B].	None.	**Key Length:** 128. **Payload Length:** 128.	Used as an internal module service to condition entropy from the output of the digitized noise source following online testing when the Intel® secure key RDSEED is selected as the entropy source during secure initialization as covered in section 10.3, Approved Mode of Operation.

CAVP Cert	Algorithm and Standard	Mode / Methods	Description / Key Size(s) / Key Strengths	Use / Function
#A1779.	**Algorithm:** HMAC_DRBG. **Standard:** [SP800-90Ar1].	**Mode:** HMAC-SHA2-512.	**Security strength:** 256-bits.	Used to support the following services: > Generate User Key; > Generate System Key; > Derive Key; > Wrap/Unwrap User Key; > Secure MKEK Distribution; > Client to Module Trusted Channel; > Encrypt Data; > Sign Data; and > Get Entropy.
Key Generation				
Vendor affirmed.	**Algorithm: CKG.** **Standard:** [SP800-133].	**Method:** symmetric keys and seed for asymmetric key generation are created based on the direct output of the module DRBG (#A1779).	**Security strength:** 256-bits.	Used to support the following services: > Generate User Key; > Generate System Key; > Secure MKEK Distribution; and > Wrap/Unwrap User Key (when using JWE format objects for export).

NOTE When used with TLS, the GCM implementation meets Scenario 1 of [FIPS 140-2 IG] A.5. It is used in a manner compliant with [SP800-52r2] and in accordance with:

- Section 4 of [RFC5288] for TLS 1.2 key establishment, and ensures when the nonce_explicit part of the IV exhausts all possible values for a given session key, that a new TLS handshake is initiated per sections 7.4.1.1 and 7.4.1.2 of [RFC5246].
- Section 5.5 from [RFC8446] for TLS 1.3 where on exhaustion of the defined limits a KeyUpdate is performed as covered in section 4.6.3 of [RFC8446].

AES GCM keys are erased when the module is power-cycled. For each new TLS session, a new AES GCM key is established.

NOTE When GCM is used by the module for all applications not involving use to support TLS:

- All encrypt operations performed in the approved mode of operation use IV generated from the approved DRBG and with a minimum length of 96-bits. This is compliant with Scenario 2 of [FIPS 140-2 IG] A.5.
- Use of external IV is exclusively permitted in the approved mode of operation for GCM decrypt operations.

Table 2-3: Non-approved algorithms allowed in the approved mode of operation

Algorithm	Caveat	Use / Function
Key Transport		
RSA	Key wrapping; key establishment methodology provides between 112-bits and 150-bits of encryption strength. Uses allowances in [FIPS 140-2 IG], D.9, Key transport methods, for key wrapping using RSA based key transport that uses the PKCS #1-v1.5 padding scheme.	Available for use with the Wrap/Unwrap User Key service for import and export of user keys protected by Thales CipherTrust Manager Core Security Module to other trusted systems. This algorithm is only permitted for use until Dec 31st 2023 after which point it is disallowed for all new encryption operations based on planned algorithm transitions outlined in [SP800-131Ar2].

2.5 Non-Approved Algorithms

Non-FIPS Approved security functions are not permitted to be used when the module is being operated in FIPS-approved mode (see section 10 for further details on configuring the approved mode of operation).

Table 2-4: Non-approved algorithms

Non-Approved Algorithm	Use / Function
AES with FF3 and FF3-1	Available as an encrypt/decrypt option with the Network Attached Encryption XML interface and REST API. This service is available to users when using user owned AES keys protected by the Thales CipherTrust Manager Core Security Module.
AES in GCM mode for encryption with externally generated IV.	Available for use with the Wrap/Unwrap User Key, Encrypt Data and Decrypt Data services when called from the NAE-XML, KMIP or Rest API and where an external IV is supplied with an encryption or wrapping request. Services can be used for encryption with GCM using these interfaces by not supplying an external IV with API calls which will trigger generation of an internally generated IV compliant with Scenario 2 of [FIPS 140-2 IG], A.5.
ARIA	Available as an encrypt/decrypt option with the Network Attached Encryption XML interface and REST API. This service is available to users when using user owned ARIA keys protected by the Thales CipherTrust Manager Core Security Module.
CHACHA20	Available for use as an encryption algorithm with the TLS 1.3 cipher-suite: TLS_CHACHA20_POLY1305_SHA256.
ECDSA (non-approved for signature verification when using keys less than 112-bits of security strength).	Available for use in validating signatures on certificate chains used with TLS.
Poly1305	Available for use as a Message Authentication Code (MAC) with the TLS 1.3 cipher-suite: TLS_CHACHA20_POLY1305_SHA256.
RSA (non-approved for key generation and signature generation when using keys less than 2048-bits).	Available for use as an option with the signing service available to users when using user owned RSA keys protected by the Thales CipherTrust Manager Core Security Module. Available for use in validating generating Certificate Authority (CA) keys on certificate chains used with TLS.
RSA (non-approved for key generation and signature verification when using keys less than 1024-bits).	Available for use in validating signatures on certificate chains used with TLS.
RSA (key wrapping; key establishment methodology; non-compliant with less than 112-bits of encryption strength).	Available for use with the key migration service for import and export of user keys protected by Thales CipherTrust Manager Core Security Module to other trusted systems.

Non-Approved Algorithm	Use / Function
SEED	Available as an encrypt/decrypt option with the Network Attached Encryption XML interface and REST API. This service is available to users when using user owned SEED keys protected by the Thales CipherTrust Manager Core Security Module.
TLS 1.0 KDF and TLS 1.1 KDF	Available as the KDF when using version 1.0 or 1.1 of the TLS protocol to connect to the module. By default, support for TLS 1.0 and TLS 1.1 is disabled.
Triple-DES, Key Generation (non-approved for key generation of less than 168-bit).	Available for use as with the module key generation service available to user for creating user owned keys protected by Thales CipherTrust Manager Core Security Module.
Triple-DES, Data Encryption/Decryption (non-approved for keys less than 168-bit).	Available as an encrypt/decrypt option with the Network Attached Encryption XML interface and REST API. This service is available to users when using user owned Triple-DES keys protected by the Thales CipherTrust Manager Core Security Module.
ECDSA (non-approved when using non-NIST elliptic curves)	Used to support the following services: > Generate System Key; > Generate User Key; > Client to Module Trusted Channel; > Wrap/Unwrap User Key > Sign Data; and > Signature Verify.
KAS-ECC-SSC (non-approved when using non-NIST elliptic curves)	Outputs from this function are used to feed approved KDF to derive encryption and MAC keys used to support the following services: > Client to Module Trusted Channel; > Wrap/Unwrap User Key; > Encrypt Data; and > Decrypt Data.

3 Cryptographic Module Interfaces

3.1 Ports and Interface Overview

As a software-only module, the module does not have physical ports. For the purpose of the [FIPS 140-2] validation, the physical ports are interpreted to be the physical ports of the GPC hosting the module.

The logical interfaces are the API through which applications request services.

The following logical interfaces are supported by the module:

- **Network Attached Encryption XML (NAE-XML) API** – provides XML based support for management of users, groups and keys alongside support access to cryptography as a service;
- **Key Management Interoperability Protocol (KMIP) API** – provides access to key management and cryptographic services in a form compatible with [KMIP 2.1]; and
- **REST API** – RESTful interface used exclusively by other applications running internal to the physical boundary of the module. This interface supports all functions available on the NAE-XML API and KMIP API alongside a number of additional lower-level functions.

The following table maps the supported interfaces to the FIPS 140-2 defined interfaces:

Table 3-1: Mapping of FIPS 140-2 Interfaces to Physical and Logical Interfaces

FIPS 140-2 Interface	Logical Interface Mapping			
	NAE-XML API	KMIP	REST API	Description
Data Input	x	x	x	Parameters passed to the module via API calls.
Data Output	x	x	x	Data returned from the module via API calls.
Control Input	x	x	x	API method calls and/or parameters passed to API calls.
Status Output	x	x	x	Information received in response to API calls. Messages sent to logs and system standard output maintained by the host platform executing the module.
Power Interface	-	-	-	There is no separate power or maintenance access interface beyond the power interface provided by the module host platform itself.

The RestAPI is supported by all binaries forming the module. The NAE-XML and KMIP API are exclusive to the Key Management (NAE) binary as shown in Figure 2-5, 'CipherTrust Core Security Manager – sub-systems, interfaces and hardware dependencies.' in section 2.2, 'Module Description'.

3.2 Trusted Channel

Thales CipherTrust Manager Core Security Module supports a TLS 1.2 or TLS 1.3[5] logically enforced trusted channel between the cryptographic module and any external network connected client using the NAE-XML API or KMIP API.

The following TLS cipher-suites from [RFC5288] and [RFC8446] are supported by default in the FIPS approved module. Cipher-suites are ordered as prioritized when negotiating the target cipher suite when setting up the trusted channel:

- **TLS 1.3:**
  - TLS_AES_256_GCM_SHA384;
  - TLS_AES_128_GCM_SHA256;
- **TLS 1.2:**
  - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384;
  - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384;
  - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256;
  - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256.

In addition, the following TLS 1.2 cipher-suites are supported by the module but are disabled by default. These cipher-suites are permitted for use in the FIPS approved mode of operation but do not offer perfect forward secrecy. Theses suites are typically only used to support legacy compatibility with older modules:

- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256;
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256;
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA;
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA;
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA;
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA;
- TLS_RSA_WITH_AES_128_CBC_SHA256;
- TLS_RSA_WITH_AES_256_CBC_SHA; and
- TLS_RSA_WITH_AES_128_CBC_SHA.

[5] CMVP has not reviewed the conformance of the TLS 1.2 and TLS 1.3 protocol implementations against the TLS standards. Testing performed during the module validation is limited to CAVP testing of cryptography used by these protocols (AES (#A1778 and #A1779), ECDH (#A1779), ECDSA (#A1779), RSA (#A1779), SHA (#A1779), TLS 1.2 KDF (#A1779) and TLS 1.3 KDF(#A1779)).

4 Roles, Services and Authentication

4.1 Roles

The module supports the definition of multiple ‘Users’ that can create, configure and use keys protected by the module. Users are authenticated based on one of a number of identity based access control schemes.

In addition to the User role, the module implicitly supports an ‘Administrator’ role. Role assumption is implied by the service being requested and the parameters supplied.

A third role is defined named ‘Internal Service’ that is implicit and maps to other internal applications internal to the CipherTrust Manager platform that use the module for protection and management of platform keys. Role assumption is implicit when access to the module is performed over the Rest API based on the service being requested and parameters supplied.

Table 4-1: Thales CipherTrust Manager Core Security Module Roles

CipherTrust Manager Role	[FIPS 140-2] mapped Role
Administrator	Crypto Officer
User	User Role / Crypto Officer Role depending on assigned privileges.
Internal Service	User Role

Services performed by each role are as covered and mapped in section 4.3, ‘Services’ in Table 4-3, ‘Services’.

Thales CipherTrust Manager Core Security Module does not support a maintenance role.

Multiple users may concurrently have authenticated session with the Thales CipherTrust Manager Core Security Module based on ownership of a valid JSON Web Token (JWT).

4.2 Authentication

The User role is the only explicitly authenticated role with other roles being assumed implicitly. There are two ways for the User role to be authenticated depending on the API being used covered in the following sections.

4.2.1 JWT based authentication - RestAPI

When using the Rest-API, the user can be authenticated through validation of the signature or MAC applied to a JSON Web Token (JWT). The key and algorithm used to sign the JWT depends on the service and context for the authentication event with two types of JWT supported:

- **Internal JWT** – these are used to validate messages sent on behalf of end-users between services internal to the module as part of performing an operation. Internal JWT are issued by the module following successful authenticating of a message from outside the modules logical boundary using one of the other listed authentication methods (i.e. external JWT or certificate based authentication).

  Internal JWT are signed and validated using RSA PKCS #1 v1.5 signatures and the 2048-bit, Internal JWT signing key and Internal JWT validation key.

> **External JWT** – these are used to validate applications running internal to the physical boundary of the module using either:

- ECDSA and P-256 with either the Platform Service JWT signing key or Platform Service JWT validation key; or
- HMAC-SHA2-256 and the External JWT signing key.

4.2.2 Certificate based authentication – KMIP and NAE-XML over TLS

Where roles exclusively access the module through a configured TLS tunnel, authentication can be configured (for a given role) to be based on the X.509 certificates used to authenticate a configured TLS end-point.

4.2.3 Authentication Mechanism Strength

The strength of each authentication option is covered in the following table:

Table 4-2: Strength of Authentication Mechanism

Mechanism	Supported Algorithm	Strength
External JWT	HMAC-SHA2-256	Probability of success with a single authentication attempt is $1/2^{256}$ = 8.63e-78. Number of authentication attempts possible in a 1 minute period: 1,800,000. Probability of success in a 1 minute period: $1.8x10^6/2^{256}$ = 1.55e-71.
	ECDSA with P-256.	Probability of success with a single authentication attempt when using P-256 is $1/2^{128}$ = 2.94e-39. The maximum number of failed attempts in a 1 minute period is: 1,200,000. Probability of success in a 1 minute period: $1.2x10^6/2^{128}$ = 1.76e-33.
Internal JWT	RSA PKCS #1 v1.5 signature using 2048-bit key.	Probability of success with a single authentication attempt is $1/2^{112}$ (7.45e-155). Number of authentication attempts possible in a 1 minute period: 780,000. Probability of success in a 1 minute period: $7.8x10^5/2^{112}$ = 1.5e-28.
X.509 Certificate	RSA ECDSA	Minimum allowed RSA signature key length gives probability of success with a single authentication attempt is $1/2^{112}$ = 7.45e-155. Minimum supported ECDSA signature key length (P-224) gives probability of success with a single authentication attempt is $1/2^{112}$ = 7.45e-155. When using both signature mechanism, the maximum number of failed attempts is when using the smallest allowed/supported keys where the maximum rate is achieved with ECDSA signed certificates in a 1 minute period is: 1,200,000. Probability of success in a 1 minute period: $1.2x10^6/2^{112}$ = 2.3e-28.

NOTE Based on the calculated false probability rate and maximum number of failed attempts, Thales CipherTrust Manager Core Security Module satisfies the [FIPS 140-2] requirements:

- *For each attempt to use the authentication mechanism, the probability shall be less than one in 1,000,000 that a random attempt will succeed or a false acceptance will occur (e.g., guessing a password or PIN, false acceptance error rate of a biometric device, or some combination of authentication methods).*
- *For multiple attempts to use the authentication mechanism during a one-minute period, the probability shall be less than one in 100,000 that a random attempt will succeed or a false acceptance will occur.*

4.3 Services

Thales CipherTrust Manager Core Security Module supports the services listed in Table 4-3, 'Services'.

All services listed in the following four tables below can be accessed in FIPS 140-2 Approved mode and, when in this mode, exclusively use the security functions listed in Table 2-2, 'Approved Algorithms' and Table 2-3, 'Non-approved algorithms allowed in the approved mode of operation'.

When the module is operating in this mode, Security Functions in section 2.5, 'Non-Approved Algorithms' are not permitted for use.

Services listed exercise at least one of the six binaries that form the module and where interdependencies exist between services. Successful use of most of the services involves exercising most of the modules six binaries.

For a complete description of CSPs referenced from the table, please see Table 6-1, 'Summary of CSPs'.

In the 'Cryptographic Keys and CSPs alongside access rights' column:

- G = Generate: The module generates or derives the CSP.
- R = Read: The CSP or key is read from the module.
- W = Write: The CSP is updated, imported, or written to the module.
- E = Execute: The module uses the CSP in performing a cryptographic operation.

Table 4-3: Services

Service	Approved Security Functions	Cryptographic Keys and CSPs alongside access rights	Role			Notes
			Admin	User	Internal Service	
Key Management Operations						
Generate User Key	**RSA** (#A1779) – Key Generation. **ECDSA** (#A2634) – EC Key Generation. **DRBG** (#A1779) – HMAC_DRBG with SHA2-512. **AES** (#A1779, #A1778) – GCM mode with 256-bit key.	**W:** (depending on requested key) User AES Key, User EC Key, User HMAC Key, User RSA Key, User Triple-DES Key, DRBG_Key, DRBG_V. **E:** DRBG_Key, DRBG_V, Account KEK for user keys.	-	x	-	Used to generate symmetric (AES, Triple-DES or HMAC) or asymmetric (RSA, EC) user keys to be protected by Thales CipherTrust Manager Core Security Module. Following creation, keys are encrypted for storage using AES in GCM mode with the Account KEK for user keys and a 16-byte random IV generated from the approved DRBG. The module returns the resulting key-ID for the stored object alongside a Fingerprint (SHA2-256 hash) or the complete key object. Keys are generated using output from the module DRBG.
Generate System Key	**RSA** (#A1779) – Key Generation. **ECDSA** (#A2634) – EC Key Generation. **DRBG** (#A1779) – HMAC_DRBG with SHA2-512. **AES** (#A1779, #A1778) – GCM mode with 256-bit key.	**W:** System Key. **E:** DRBG_Key, DRBG_V, MKEK.	-	-	x	Used by applications running internal to the physical boundary of the module to generate system keys. Generated keys can be retrieved by the application that created them and used to support security functions outside the logical boundary of the module but internal to its physical boundary.

Service	Approved Security Functions	Cryptographic Keys and CSPs alongside access rights	Role			Notes
			Admin	User	Internal Service	
Derive Key	**KDA** (#A1779) – HKDF with SHA1, SHA2-224, SHA2-256, SHA2-384, SHA2-512. **DRBG** (#A1779) – HMAC_DRBG with SHA2-512. **AES** (#A1779, #A1778) – GCM mode with 256-bit key.	**W:** (depending on target output key) User AES Key, User HMAC Key, User Triple-DES Key DRBG_Key, DRBG_V. **E:** (depending on derivation algorithm requested) User AES Key, User HMAC Key, User Triple-DES Key. Account KEK for user keys. DRBG_Key, DRBG_V.	-	x	-	Used to derive a key based on other user keys protected by Thales CipherTrust Manager Core Security Module. Following creation, keys are encrypted for storage using AES in GCM mode with the Account KEK for user keys and a 16-byte random IV generated from the approved DRBG. The module returns the resulting key-ID for the stored object alongside a fingerprint (SHA2-256 hash) or the complete key object.
Manage Key Owner	None.	None.	-	x	-	Used to manage the owner of a given stored key object protected by Thales CipherTrust Manager Core Security Module.

Wrap/Unwrap User Key	**Basic Wrap/Unwrap Operation:** **AES** (#A1779, #A1778) – CBC, ECB, GCM, KW and KWP modes with 128,192 or 256-bit key. **HMAC** (#A2634) - HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512. **Additional security functions when key wrapping method includes key agreement to derive a transport key:** **KAS-SSC-ECC** (#A1779)- One-Pass Diffie-Hellman, C(1e,1s) with curves P-224, P-256, P-384, P-512. **ECDSA** (#A2634) – Key Generation. **CVL** (#A1779) – X9.63 KDF with SHA2-224, SHA2-256, SHA2-384, SHA2-512. **SHA** (#A1779) – SHA2-256. **DRBG** (#A1779) – HMAC_DRBG with SHA2-512. **Additional security functions when key wrapping method includes derivation of a transport key based on another user key or user supplied CSP:** **KDA** (#A1779) – HKDF with SHA1, SHA2-224, SHA2-256, SHA2-384, SHA2-512. **PBKDF** (#A1779) – HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512, HMAC-SHA2-512/224, HMAC-SHA2-512/256, HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512.	**R:** (depending on requested key) User AES Key, User EC Key, User HMAC Key, User RSA Key, User Triple-DES Key. **W:** (for key import) User AES Key, User EC Key, User HMAC Key, User RSA Key, User Triple-DES Key. (for key export using ephemeral keys) DRBG_Key, DRBG_V. **G:** (for key export using ephemeral methods) Ephemeral EC Key for User Key Export, User Key Wrapping Key. **E:** User Key Wrapping PWD, User AES Key, User Triple-DES Key, User HMAC Key, DRBG_Key, DRBG_V, Account KEK for user keys.	-	x	-	Used to import or export a user key on request of the user. Imported keys to be protected by Thales CipherTrust Manager Core Security Module are encrypted for storage using AES in GCM mode with the Account KEK for user keys. The module returns the resulting key-ID for the stored object alongside a Fingerprint (SHA2-256 hash) or the complete key object. Exported keys are recovered from storage and returned to the client requesting the export. Further details of all key wrapping methods supported for user key import or export including cryptography used are covered in section 6.3, 'Key Import and Export Methods'.

Service	Approved Security Functions	Cryptographic Keys and CSPs alongside access rights	Role			Notes
			Admin	User	Internal Service	
Get User Key Status / attributes	**AES** (#A1779, #A1778) – GCM mode with 256-bit key.	**R:** (depending on requested key) User AES Key, User EC Key, User HMAC Key, User RSA Key, User Triple-DES Key. **E:** (when reading sensitive key attributes) Account KEK for user keys.	-	x	-	Used to recover information about a user key object protected by Thales CipherTrust Manager Core Security Module. Can separately be used to read plaintext user key values, which are returned to the requesting client.
List User Keys	None.	None.	-	x	-	Used to list user keys protected by Thales CipherTrust Manager Core Security Module.
Get System Key Status / attributes	**AES** (#A1779, #A1778) – GCM mode with 256-bit key.	**R**: MKEK, Cluster node private key, External JWT signing key, Platform Service JWT signing key, TLS Cert Private Keys.	-	-	x	Used to recover information about a system key protected by Thales CipherTrust Manager Core Security Module. System keys are keys used exclusively internal to the physical boundary of the module to support CipherTrust Manager services. This service can separately be used to read plaintext key values, which are returned to the requesting application. In the context of this service System Key includes the following named CSP in addition to other keys created by applications internal to the modules physical boundary: MKEK, Cluster node private key, External JWT signing key, Platform Service JWT signing key, TLS Cert Private Keys.

Service	Approved Security Functions	Cryptographic Keys and CSPs alongside access rights	Role			Notes
			Admin	User	Internal Service	
Configure key attributes	None.	None.	-	x	-	Used to set the value of a user key attributes for user key material protected by Thales CipherTrust Manager Core Security Module. This command cannot be used to set sensitive key attributes containing the actual key values, which can only be set using the Wrap/Unwrap User Key, Generate User Key and Derive Key services.
Update User Key State (Destroy / Archive / Recover / Revoke / Reactivate)	None.	None.	-	x	-	Used to trigger life-cycle events that update the state of a given version of a user key protected by Thales CipherTrust Manager Core Security Module. States supported across the KMIP and REST API include: “Pre-Active”, “Active”, “Deactivated”, “Compromised”, “Destroyed”, “Destroyed Compromised”. States supported by NAE-XML API include: “Pre-Active”, “Active”, “Restricted/Retired”.
Delete User Key	None.	None.	-	x	-	Used to erase a User Key protected by the Thales CipherTrust Manager Core Security Module.
Delete System Key	None.	None.	-	-	x	Used to erase System Key created by applications running internal to the physical boundary of the module and where the keys are managed by the Thales CipherTrust Manager Core Security Module.

Service	Approved Security Functions	Cryptographic Keys and CSPs alongside access rights	Role			Notes
			Admin	User	Internal Service	
Copy User Key	**AES** (#A1779, #A1778) – GCM mode with 256-bit key. **DRBG** (#A1779) – HMAC_DRBG with SHA2-512.	**R:** (depending on requested key) User AES Key, User EC Key, User HMAC Key, User RSA Key, User Triple-DES Key. **W:** (for key import) User AES Key, User EC Key, User HMAC Key, User RSA Key, User Triple-DES Key, DRBG_Key, DRBG_V. **E:** DRBG_Key, DRBG_V, Account KEK for user keys.	-	x	-	Used to create a copy of an existing User Key object protected by Thales CipherTrust Manager Core Security Module. A new copy of the key is created and encrypted for storage using AES in GCM mode with the Account KEK for user keys and a 16-byte random IV generated from the approved DRBG. The module returns the resulting key-ID for the stored object alongside a Fingerprint (SHA2-256 hash) or the complete key object.
Module Management Operations						
Configure / Register User and Client Identity	None.	None.	-	x	-	Register/de-register a client as a user on the NAE-XML or KMIP API.
Configure Noise Source	None.	None.	x	-	-	Used to configure which noise source is to be used to seed the platform DRBG.
Initialize Root-of-Trust	None.	None.	x	-	-	Used to initialize an external HSM as a root-of-trust for the Thales CipherTrust Manager Core Security Module.
Configure Root-of-Trust	None.	None.	-	x	-	This service manages the process of configuring a root-of-trust once initialized.

Service	Approved Security Functions	Cryptographic Keys and CSPs alongside access rights	Role			Notes
			Admin	User	Internal Service	
Secure MKEK Distribution	**KAS-ECC-SSC** (#A1779) **KDA** (#A1779) – OneStep KDF with SHA2-256. **AES** (#A1779, #A1778) – CTR mode with 128-bit key. **HMAC** (#A1779) – HMAC-SHA2-256 with 128-bit key. **DRBG** (#A1779) – HMAC_DRBG with SHA2-512.	**R:** MKEK. **G/E:** EC Key for MKEK Export, MKEK Transport Key, MKEK Transport MAC Key. **E:** Cluster node certificate**,** Cluster node private key**,** DRBG_Key, DRBG_V. **W:** DRBG_Key, DRBG_V.	-	-	x	Used to securely: 1. transfer the MKEK to other instances of the Thales CipherTrust Manager Core Security Module operating within a cluster servicing a shared pool of keys. 2. package the MKEK for storage during a backup or restore operation of the CipherTrust Manager appliance. This service generates the MKEK Transport Key and MKEK Transport MAC Key using ECDH and OneStep KDF with SHA2-256. Following derivation of the transport keys, the service encrypts the MKEK under these ahead of returning the encrypted object to the calling service. Key encapsulation of the MKEK under the derived keys is covered in further detail in section 6.3, 'Key Import and Export Methods'.
Launch/Stop/ Restart Service	None.	None.	x	x	-	This service runs in response to system inputs by either the Crypto-Officer or User and is responsible for Launching, restarting or stopping services that underpin the Thales CipherTrust Manager Core Security Module. In the case Thales CipherTrust Manager Core Security Module encounters a critical error during operation (e.g. Self-test failure) this service is responsible for stopping all module services.

Service	Approved Security Functions	Cryptographic Keys and CSPs alongside access rights	Role			Notes
			Admin	User	Internal Service	
Client to Module Trusted Channel	**RSA (**#A1779) – Signature Generation and Verification. **ECDSA** (#A1779) – Signature Generation and Verification. **KAS-ECC-SSC** (#A1779) – (Cofactor) Ephemeral Unified C(2e,0s, ECC CDH) with curve P-224, P-256, P-384 or P-512. **AES (**#A1779, #A1778) – CBC or GCM mode with 128 or 256-bit keys. **KDF (**#A1779) – TLS 1.2 or TLS 1.3 KDF. **HMAC** (#A1779) - HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512. **DRBG** (#A1779) – HMAC_DRBG with SHA2-512.	**R:** TLS Cert Private Keys. **G/E:** TLS ECDHE Private Components, TLS Master Secret, TLS Pre-master Secret, TLS Encryption Keys, TLS HMAC Keys. **W:** DRBG_Key, DRBG_V.	-	x	-	This service supports the client to module TLS tunnel used with both the NAE-XML and KMIP interfaces. Details of supported TLS versions and cipher-suite are provided separately in section 3.2, 'Trusted Channel'.

Service	Approved Security Functions	Cryptographic Keys and CSPs alongside access rights	Role			Notes
			Admin	User	Internal Service	
Validate External JWT	**HMAC** (#A1779) – HMAC-SHA2-256. **ECDSA** (#A1779) – Sign and Verify with curve P-256.	**E:** External JWT signing key, Platform Service JWT validation key. MKEK, Local Storage MKEK.	-	-	-	This is an internal module service used to validate external JSON Web Tokens (JWT). JWT are included in messages sent to the module to authenticate roles and system components. External JWT signed with HMAC are received from external clients. External JWT signed with ECDSA are received from applications running within the physical boundary of the module. For more information on JWT as used for authentication, this is covered in section 4.2, 'Authentication'.
Issue / validate Internal JWT	**RSA** (#A1779) – PKCS#1 v1.5 Sign and Verify with 2048-bit modulus.	**E:** Internal JWT signing key, Internal JWT validation key. Local Storage MKEK. **W:** Internal JWT (to destination service when issuing JWT).	-	-	-	This is an internal module service used to issue and validate Internal JWT. JWT are included in messages on the REST API to authenticate roles and system components. For more information on JWT as used for authentication, this is covered in section 4.2, 'Authentication'.
Get Module Status	None.	None.	x	x	x	This service is used to access status information on the Thales CipherTrust Manager Core Security Module as part its deployment as part of the CipherTrust Manager.

Service	Approved Security Functions	Cryptographic Keys and CSPs alongside access rights	Role			Notes
			Admin	User	Internal Service	
User Cryptographic Services						
Encrypt Data	**Basic Encrypt/Decrypt Operation:** **AES** (#A1779) – ECB, CBC, GCM. **Triple-DES** (#A1779) – ECB, CBC. **KTS-RSA** (#A1779) – encrypt and decrypt with 2048, 3072 and 4096-bit modulus. **HMAC** (#A1779) - HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512. **Additional security functions when an encrypt operation includes key agreement to derive an encryption key:** **KAS-ECC-SSC** (#A2634) – One-Pass Diffie-Hellman, C(1e,1s) with curves P-224, P-256, P-384, P-512. **ECDSA** (#A2634) – Key Generation. **CVL** (#A2634) – X9.63 KDF with SHA2-224, SHA2-256, SHA2-384, SHA2-512. **SHA** (#A1779) – SHA2-256. **DRBG** (#A1779) – HMAC_DRBG with SHA2-512.	**E:** User AES Key, User RSA Key, User Triple-DES Key, User EC Key. DRBG_Key, DRBG_V. Account KEK for user keys. **W:** DRBG_Key, DRBG_V.	-	x	-	User service to encrypt supplied data using user keys (e.g. User AES Key) stored and protected by the Thales CipherTrust Manager Core Security Module. The service supports encryption options using both symmetric alongside asymmetric cryptography. When using symmetric encryption, IV are randomly generated by the module if not supplied by the client with data to be encrypted. User keys used for encrypt operations are decrypted from storage using the Account KEK for user keys ahead of use.

Service	Approved Security Functions	Cryptographic Keys and CSPs alongside access rights	Role			Notes
			Admin	User	Internal Service	
Decrypt Data	<As per Encrypt Data above.>	**E:** User AES Key, User RSA Key, User Triple-DES Key, User EC Key.	-	x	-	User service to decrypt supplied data using user keys (e.g. User AES Key) stored and protected by the Thales CipherTrust Manager Core Security Module. The service supports decryption options using both symmetric alongside asymmetric cryptography. User keys used for decrypt operations are decrypted from storage using the Account KEK for user keys ahead of use.
Sign Data	RSA (#A1779) – PKCS #1, v1.5 signing, PKCS #1 v1.5 PSS signing. ECDSA (#A2634) – Signing with curves P-224, P-256, P-384, P-512).	**E:** User RSA Key, User EC Key. DRBG_Key, DRBG_V. Account KEK for user keys. **W:** DRBG_Key, DRBG_V.	-	x	-	User service to sign supplied data using user keys (e.g. User EC Key) stored and protected by the Thales CipherTrust Manager Core Security Module. When using ECDSA, additional inputs to the operation are randomly generated by the module. User keys used for signing operations are decrypted from storage using the Account KEK for user keys ahead of use.
Signature Verify	<As per Sign Data above.>	**E:** User RSA Key, User EC Key.	-	x	-	User service to verify a supplied signature using user keys (e.g. User EC Key) stored and protected by the Thales CipherTrust Manager Core Security Module.
MAC Generate	HMAC (#A1779) - HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512.	**E:** User HMAC Key. Account KEK for user keys.	-	x	-	User service to generate a MAC over user supplied data using User HMAC Key stored and protected by the Thales CipherTrust Manager Core Security Module. User keys used for MAC are decrypted from storage using the Account KEK for user keys ahead of use.

Service	Approved Security Functions	Cryptographic Keys and CSPs alongside access rights	Role			Notes
			Admin	User	Internal Service	
MAC Verify	<As per MAC Generate above.>	**E:** User HMAC Key. Account KEK for user keys.	-	x	-	User service to verify a MAC over user supplied data using User HMAC Key stored and protected by the Thales CipherTrust Manager Core Security Module. User keys used for MAC are decrypted from storage using the Account KEK for user keys ahead of use.
Get Entropy	DRBG (#A1779) – HMAC_DRBG with SHA2-512.	**E:** DRBG_Key, DRBG_V. **W:** DRBG_Key, DRBG_V.	-	x	x	User service to supply raw entropy using the outputs from the platform DRBG (HMAC_DRBG).

5 Operating Environment

The module supports a **modifiable operating environment** as defined in section 4.6 from [FIPS 140-2].

6 CSP Management

6.1 Critical Security Parameter

The following table lists Critical Security Parameters (CSP) used to perform approved security function supported by the cryptographic module:

Table 6-1: Summary of CSPs

Key / CSP name and Type	Security Function and Cert Number	Generation or Establishment	Import / Export	Storage	Use and Related Keys
Binary Validation Key 4096-bit RSA public key	RSA (#A1779).	N/A.	N/A – public RSA key embedded in the binary during manufacture.	Plaintext storage, hardcoded in the module image.	This key is used to validate the integrity of the module binary during module integrity test run as part of power-on self-test.
MKEK PWD 32-bytes	PBKDF (#A1779).	[SP800-90Ar1] HMAC_DRBG with HMAC-SHA2-512.	N/A – passed in from the operating system responsible for reading and writing the key to HDD.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	The key is used to derive a KEK using PBKDF that is used to wrap the MKEK for storage when a separate root-of-trust isn't configured for the module.

Key / CSP name and Type	Security Function and Cert Number	Generation or Establishment	Import / Export	Storage	Use and Related Keys
MKEK 256-bit AES Key	AES (#A1779, #A1778).	[SP800-90Ar1] HMAC_DRBG with HMAC-SHA2-512.	**Option 1:** Imported/exported encrypted[6] between instances of the module using AES in CTR mode with 128-bit key MKEK Transport Key and HMAC-SHA2-256, using the MKEK Transport MAC Key as part of the Secure MKEK Distribution service. Transport keys are derived using X9.63 KDF from shared-secret output from ECDH and using two instances of the EC Key for MKEK Export. **Option 2:** Imported/exported plaintext to/from other application running internal to the physical boundary of the module when an independent root-of-trust is configured. **Option 3:** Imported wrapped by a KEK derived from the MKEK PWD using PBKDF. Stored key is wrapped using AES in GCM mode with 256-bit key and 12-byte random IV generated from the approved DRBG.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	System Master KEK. Used to protect the top hierarchy of keys for all services. This key is used with AES in GCM mode with a 16-byte random IV when encrypting objects for storage. This key is used to encrypt/decrypt the following key: > Master KEK for user keys; > Platform Service JWT signing key; > Platform Service JWT validation key; > System Key.

Key / CSP name and Type	Security Function and Cert Number	Generation or Establishment	Import / Export	Storage	Use and Related Keys
Local Storage MKEK 256-bit AES Key	AES (#A1779, #A1778).	[SP800-90Ar1] HMAC_DRBG with HMAC-SHA2-512.	N/A – passed in from the operating system responsible for reading and writing the key to HDD.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	This is the system Local Storage Master KEK. Used to provide a level of obfuscation to system top hierarchy secrets that are not to be replicated to other cluster nodes – also known as 'local secrets'. This key is used to encrypt/decrypt the following key: > Internal JWT signing key; > Internal JWT validation key; > TLS Cert Private Keys.
Cluster node certificate	KAS-ECC-SSC (#A1779).	N/A.	Transferred in plaintext from/to other applications internal to the physical boundary of the module using its Rest API.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Used as static input to ECDH when deriving the MKEK Transport Key as part of the Secure MKEK Distribution service and acting in the initiator role. Key is stored outside the module boundary and only imported for use.
Cluster node private key	KAS-ECC-SSC (#A1779).	N/A.	Transferred in plaintext from/to other applications internal to the physical boundary of the module using its Rest API.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Used as static input to ECDH when deriving the MKEK Transport Key as part of the Secure MKEK Distribution service and acting in the responder role. Key is generated and stored outside the module boundary and only imported for use and then erased.

[6] When a root-of-trust is configured – the copy of the MKEK transferred is the pre-encrypted copy of the key created by the root-of-trust.

Key / CSP name and Type	Security Function and Cert Number	Generation or Establishment	Import / Export	Storage	Use and Related Keys
EC Key for MKEK Export ECC private key on curve P-256.	KAS-ECC-SSC (#A1779).	[FIPS 186-4], Appendix B.4.2.	N/A.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Ephemeral keys used for exporting and importing MKEKs between instances of Thales CipherTrust Manager Core Security Module running on different physical platforms. Keys are erased following use for a single import/export operation and then erased.
MKEK Transport Key 128-bit AES Key.	AES (#A1779, #A1778).	Derived using ECDH [SP800-56Ar3], Ephemeral unified with [SP800-56Cr2] OneStep KDF using SHA2-256.	N/A.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Used to wrap the MKEK for transport between instances of Thales CipherTrust Manager Core Security Module running on different physical platforms. Keys are erased following use for a single import/export operation.
MKEK Transport MAC Key 128-bit HMAC Key	HMAC (#A1779).	Derived using ECDH [SP800-56Ar3], Ephemeral unified with [SP800-56Cr2] OneStep KDF using SHA2-256.	N/A.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Used to add MAC to the MKEK for transport between instances of Thales CipherTrust Manager Core Security Module running on different physical platforms. Keys are erased following use for a single import/export operation.

Key / CSP name and Type	Security Function and Cert Number	Generation or Establishment	Import / Export	Storage	Use and Related Keys
External JWT signing key 64 byte HMAC key	HMAC (#A1779).	[SP800-90Ar1] HMAC_DRBG with HMAC-SHA2-512.	**Option 1:** Import/Export encrypted under the MKEK using AES in GCM mode with 256-bit key and 16-byte random IV generated from the approved DRBG. **Option 2:** Transferred in plaintext when using the REST API from other applications internal to the physical boundary of the module.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Used to sign and verify external JWT tokens. This key used to create JWT for clients and where the resulting JWT can be presented as part of authenticating to a given service. Signed JWT are authenticated on presentation to the module and a resulting Internal JWT created on successful authentication. The Internal JWT are used for onward use within the CipherTrust Manager to access trusted services.
Internal JWT signing key 2048-bit RSA private key.	RSA (#A1779).	[FIPS 186-4], Appendix B.3.3.	N/A - passed in from the operating system responsible for reading and writing the key to HDD.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Used to sign internal JWT tokens. The key is used to sign all internal JWTs that can be used to make requests with internal services.
Internal JWT validation key 2048-bit RSA public key.	RSA (#A1779).	[FIPS 186-4], Appendix B.3.3.	N/A - passed in from the operating system responsible for reading and writing the key to HDD.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Used to validate internal JWT tokens. The key is used to verify all internal JWTs that can be used to make requests with internal services. Services use the public key to verify the incoming requests are properly signed as a valid request prior to accepting to process them.

Key / CSP name and Type	Security Function and Cert Number	Generation or Establishment	Import / Export	Storage	Use and Related Keys
Platform Service JWT signing key ECC private key on curve P-256.	ECDSA (#A1779).	[FIPS 186-4], Appendix B.4.1.	**Option 1:** Import/Export encrypted under the MKEK using AES in GCM mode with 256-bit key and 16-byte random IV generated from the approved DRBG. **Option 2:** Transferred in plaintext when using the REST API from other applications internal to the physical boundary of the module.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Generated on first boot. Each platform service owns a signing key used to sign its own JWTs and act as a service account on behalf of its end-points. JWT are used to authenticate internal messages passed between components of the CipherTrust Manager.
Platform Service JWT validation key ECC public key on curve P-256.	ECDSA (#A1779).	[FIPS 186-4], Appendix B.4.1.	**Option 1:** Import/Export encrypted under the MKEK using AES in GCM mode with 256-bit key and 16-byte random IV generated from the approved DRBG. **Option 2:** Transferred in plaintext when using the REST API from other applications internal to the physical boundary of the module.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Public key used to validate JWT created by application running internal to the physical boundary of the module.
Internal JWT JWT Token.	RSA (#A1779).	N/A.	Transferred in plaintext when using the REST API between services internal to the physical boundary of the module.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	JSON Web Token used for authentication between communicating services internal to the physical boundary of the module.

Key / CSP name and Type	Security Function and Cert Number	Generation or Establishment	Import / Export	Storage	Use and Related Keys
External JWT JWT Token.	ECDSA (#A1779). HMAC (#A1779).	N/A.	**Option 1:** Import/Export over TLS (when authenticating remote users). **Option 2:** Transferred in plaintext when using the REST API from other applications internal to the physical boundary of the module.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	JSON Web Token used for authenticating remote users communicating with the module through other applications outside the logical boundary of the module but inside it's physical boundary OR applications running internal to the physical boundary of the module.
Master KEK for user keys 256-bit AES Key	AES (#A1779, #A1778).	[SP800-90Ar1] HMAC_DRBG with HMAC-SHA2-512.	Import/Export encrypted under the MKEK using AES in GCM mode with 256-bit key and 16-byte random IV generated from the approved DRBG.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	This key is used to encrypt/decrypt the following key: > Account KEK for user keys.

Key / CSP name and Type	Security Function and Cert Number	Generation or Establishment	Import / Export	Storage	Use and Related Keys
Account KEK for user keys 256-bit AES Key	AES (#A1779, #A1778).	[SP800-90Ar1] HMAC_DRBG with HMAC-SHA2-512.	Import/Export with API calls encrypted under the Master KEK for user keys using AES in CBC mode[7] with 256-bit key and 16-byte random IV generated from the approved DRBG.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Used to encrypt/decrypt user keys stored and protected by the Thales CipherTrust Manager Core Security Module. A unique instance of the Account KEK for user keys is used for each domain created on a given CipherTrust Manager. This key is used to encrypt/decrypt the following key: > User AES Key; > User Triple-DES Key; > User HMAC Key > User RSA Key; and > User EC Key.

[7] for the purposes of the certified module, this key is considered to be exported in plaintext.

Key / CSP name and Type	Security Function and Cert Number	Generation or Establishment	Import / Export	Storage	Use and Related Keys
User AES Key 128, 192 or 256-bit AES Key.	AES (#A1779, #A1778, #A2634, #A2635). HKDF (#A1779).	**Option 1:** [SP800-90Ar1] HMAC_DRBG with HMAC-SHA2-512. **Option 2:** Can be derived from another key using HKDF from [SP800-56Cr2] and user selected HMAC variant. **Option 3:** N/A. When key imported.	**Option 1:** Imported/exported encrypted under the Account KEK for user keys using AES in GCM mode with 256-bit key and 16-byte random IV generated from the approved DRBG. **Option 2:** Imported/exported on request of user using one of the methods for User Keys covered in section 6.3, 'Key Import and Export Methods'.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	User keys for use with AES, created on request of the end-user and subsequently used by them with the user consumable cryptographic services. Key can be used to encrypt/decrypt data or can be used to derive keys subsequently used to wrap/unwrap other user key objects for import/export when used with HKDF.
User EC Key ECC public/private key on curve P-224, P-256, P-384 or P-521.	ECDSA (#A2634). KAS-ECC-SSC (#A1779, #A2634).	**Option 1:** [FIPS 186-4], Appendix B.4.2. **Option 2:** N/A. When key imported.	<As per 'User AES Key' above.>	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	User keys for use with ECDSA for signature creation of verification. Keys can separately also be used to support key derivation of keys used as an option during user key export operations. Keys are created on request of the end-user and subsequently used by them with the user consumable cryptographic services or for key export purposes.

Key / CSP name and Type	Security Function and Cert Number	Generation or Establishment	Import / Export	Storage	Use and Related Keys
User HMAC Key 128, 160 and 256-bit HMAC Key	HMAC (#A2634). HKDF (#A1779).	**Option 1:** [SP800-90Ar1] HMAC_DRBG with HMAC-SHA2-512. **Option 2:** Can be derived from another key using HKDF from [SP800-56Cr2] and user selected HMAC variant. **Option 3:** N/A. When key imported.	<As per 'User AES Key' above.>	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	User keys for use with HMAC, created on request of the end-user and subsequently used by them with the user consumable cryptographic services. Key can be used to generate or validate MAC or can be used to derive keys subsequently used to wrap/unwrap other user key objects for import/export when used with HKDF.

Key / CSP name and Type	Security Function and Cert Number	Generation or Establishment	Import / Export	Storage	Use and Related Keys
User RSA Key Public/private key pair 1024, 2048, 3072 or 4096 bits	RSA (#A1779). KTS-RSA (#A1779).	**Option 1:** [FIPS 186-4], Appendix B.3.3. **Option 2:** N/A. When key imported.	<As per 'User AES Key' above except only Symmetric Encryptions options are available with JWE export format.>	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	User keys with the option to be used with RSA for sign and verify and separately for wrapping of user AES, Triple-DES and HMAC keys for import/export. The exact role of a given key is controlled based on a series of configurable key attributes that can be controlled by the key owner. Created on request of the end-user and subsequently used by them with the user consumable cryptographic services. Keys with modulus length of 1024-bit can exclusively be used for signature verification operations and where these are exclusively imported to the module. This key type can only be used with RSA using PKCA#1 v1.5 encryption operations until Dec 31st 2023 after which point it is disallowed for encrypt operations using this algorithm based on planned algorithm transitions outlined in [SP800-131Ar2]. Continued use for both encrypt and decrypt operations is permitted beyond Dec 31st 2021 when using this key with KTS-OAEP-Basic for encrypt and decrypt.

Key / CSP name and Type	Security Function and Cert Number	Generation or Establishment	Import / Export	Storage	Use and Related Keys
User Triple-DES Key 168 bit key	Triple-DES (#A1779, #A2634). HKDF (#A1779).	**Option 1:** [SP800-90Ar1] HMAC_DRBG with HMAC-SHA2-512. **Option 2:** Can be derived from another key using HKDF from [SP800-56Cr2] and user selected HMAC variant. **Option 3:** N/A. When key imported.	<As per 'User AES Key' above.>	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	User keys for use with Triple-DES, created on request of the end-user and subsequently used by them with the user consumable cryptographic services. Key can be used to encrypt/decrypt data or can be used to derive keys subsequently used to wrap/unwrap other user key objects for import/export when used with HKDF. This key type can only be used for new encryption operations until Dec 31st 2023 after which point it is disallowed for encrypt operations based on planned algorithm transitions outlined in [SP800-131Ar2].
User Key Wrapping PWD 8 – 128 bytes	PBKDF (#A1779).	N/A.	**Option 1:** Imported over the TLS trusted channel when used with KMIP or NAE-XML API. **Option 2:** Transferred in plaintext when using the REST API from other applications internal to the physical boundary of the module.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	User supplied password used as one option to derive the User Key Wrapping Key. Derived key is permitted for use in approved mode exclusively when used to wrap / unwrap user keys for/from storage. Additional guidance is provided in section 10.6, 'Security Rules'.

Key / CSP name and Type	Security Function and Cert Number	Generation or Establishment	Import / Export	Storage	Use and Related Keys
User Key Wrapping Key 128, 192 or 256 bit AES key	AES (#A1779, #A1778).	**Option 1:** Derived from another key using HKDF from [SP800-56Cr2] and user selected HMAC variant. **Option 2:** Derived using ECDH and One-Pass Diffie-Hellman [SP800-56Ar3] with X9.63 KDF [SP800-135r1]. **Option 3:** Derived using [SP800-132] from User Key Wrapping PWD.	N/A.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Ephemeral key encryption / decryption key either: > derived from other keys using HKDF to AES wrap (KW) another user key for export; or > derived using ECDH and X9.63 KDF. > derived from a user supplied password using PBKDF. The key is erased following the export. When derived using PBKDF, the derived key is permitted for use in approved mode exclusively when used to wrap / unwrap user keys to/from storage. Additional guidance is provided in section 10.6, 'Security Rules'.
Ephemeral EC Key for User Key Export ECC Key-pair on curve P-224, P-256, P-384 or P-512.	KAS-ECC-SSC (#A1779, #A2634).	[FIPS 186-4], Appendix B.4.2.	N/A.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Ephemeral keys generated in scope and used with ECDH and X9.63 KDF to generate a symmetric transport key used to encrypt user keys for export. Used for encrypting user specified data. The key is erased following use to derive the transport key.

Key / CSP name and Type	Security Function and Cert Number	Generation or Establishment	Import / Export	Storage	Use and Related Keys
JWE Content Encryption Key for User Key Export 128, 192 or 256-bit AES Key	AES (#A1779, #A1778).	[SP800-90Ar1] HMAC_DRBG with HMAC-SHA2-512.	Export under a user key using either Symmetric or Asymmetric encryption option. For a full list of encryption options, see Section 6.3, 'Key Import and Export Methods'.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Ephemeral key used in the JWE encryption scheme to encrypt the target user key being exported. Encryption uses AES (128, 192 or 256-bit) with GCM mode or CBC mode with a separate HMAC. This key is included in the JWE package encrypted under a key and using an algorithm selected by the end-user.
JWE Content MAC Key for User Key Export 128, 192 or 256-bit MAC Key	HMAC (#A1779).	[SP800-90Ar1] HMAC_DRBG with HMAC-SHA2-512	Export under a user key using either Symmetric or Asymmetric encryption option. For a full list of encryption options, see Section 6.3, 'Key Import and Export Methods'.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Ephemeral key used in the JWE encryption scheme to MAC the target user key being exported when CBC mode encryption is used. This key is included in the JWE package encrypted under a key and using an algorithm selected by the end-user.
TLS Cert Private Keys RSA (2048, 3072 or 4096-bit) or ECDSA (Curve: P-224, P-256, P-384, P-521).	RSA (#A1779). ECDSA (#A1779).	N/A.	**Option 1:** Transferred in plaintext from other applications internal to the physical boundary of the module using its Rest API. **Option 2:** N/A **–** passed encrypted under the Local Storage MKEK for user keys using AES in GCM mode with 256-bit key and 12-byte random IV generated from the approved DRBG to the host operating system for storage in the HDD.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Used with the TLS based trusted channel covered in section 3.2, Trusted Channel. Key is used to support authentication of the module by the client.

Key / CSP name and Type	Security Function and Cert Number	Generation or Establishment	Import / Export	Storage	Use and Related Keys
TLS ECDHE Private Components	KAS-ECC-SSC (#A1779).	[FIPS 186-4], Appendix B.4.2.	N/A	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Used with the TLS based trusted channel covered in section 3.2, Trusted Channel. Keys are used as the ephemeral keys used during the ECDH derivation of the TLS Pre-Master Secret (TLS 1.2) or TLS Master Secret (TLS 1.3).
TLS Pre-master Secret (TLS 1.2 only) 112 to 260 bits	CVL (#A1779) – TLS 1.2 KDF.	[SP800-56Ar3] using C(2e,0s,ECC CDH) (#A1779).	N/A	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Used with the TLS based trusted channel covered in section 3.2, Trusted Channel. Intermediate secret used as part of the TLS 1.2 protocol.
TLS Master Secret 384 bits	CVL (#A1779) – TLS 1.2 or TLS 1.3 KDF.	[SP800-135r1] TLS 1.2 or TLS 1.3 KDF.	N/A	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Used with the TLS based trusted channel covered in section 3.2, Trusted Channel. Intermediate secret used as part of the TLS protocol.

Key / CSP name and Type	Security Function and Cert Number	Generation or Establishment	Import / Export	Storage	Use and Related Keys
TLS Encryption Keys 128 or 256-bit AES Key.	AES (#A1779, #A1778).	[SP800-135r1] TLS 1.2 or TLS 1.3 KDF.	N/A.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Used with the TLS based trusted channel covered in section 3.2, Trusted Channel. Data encryption key used to encapsulate traffic transiting the TLS tunnel once key establishment is complete. The key is used with AES in GCM mode to additionally provide authentication in all TLS 1.3 cipher suites and select suites from TLS 1.2. Used in certain supported TLS suites for AES in CBC mode where in addition a separate HMAC is included with messages.
TLS HMAC Keys (TLS 1.2 only)	HMAC (#A1779).	<As per TLS Encryption Keys>	N/A.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Used with the TLS based trusted channel covered in section 3.2, Trusted Channel. MAC key used with HMAC-SHA2-256 to detect modification of traffic transiting a TLS 1.2 tunnel once key establishment is complete.
System Key RSA (2048, 3072 or 4096-bit) or ECC (Curve: P-224, P-256, P-384, P-521), 128, 192 or 256-bit MAC Key, 128, 192 or 256-bit AES Key.	RSA (#A1779). ECDSA (#A1779). KAS-ECC-SSC (#A1779). HMAC (#A1779). AES (#A1779, #A1778).	**RSA:** [FIPS 186-4], Appendix B.3.3. **ECC:** [FIPS 186-4], Appendix B.4.1. **AES and HMAC:** [SP800-90Ar1] HMAC_DRBG with HMAC-SHA2-512.	**Option 1:** Import/Export encrypted under the MKEK using AES in GCM mode with 256-bit key and 16-byte random IV generated from the approved DRBG. **Option 2:** Transferred in plaintext to other applications internal to the physical boundary of the module using its Rest API.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Keys used by applications running internal to the physical boundary of the module. Keys are used to support internal security functions and services supported by the CipherTrust manager product and where these keys are available in a plaintext form for transfer to other applications.

Key / CSP name and Type	Security Function and Cert Number	Generation or Establishment	Import / Export	Storage	Use and Related Keys
DRBG Seed 111-bytes.	DRBG (#A1779).	ENT (P) (No Cert).	N/A.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Seed used as input to the platform DRBG (HMAC_DRBG) function. Seed is either drawn from the configured entropy source (Intel® SecureKey® RDSeed) or supplied via callback to the module during power-on where an alternate noise source has been configured.
DRBG_Key	DRBG (#A1779).	Generated as internal state of HMAC_DRBG as per [SP800-90Ar1].	N/A.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Part of the internal state of the platform DRBG (HMAC_DRBG) following instantiation.
DRBG_V	DRBG (#A1779).	<As per DRBG_V>	N/A.	Stored in DRAM as managed by the host OS. DRAM is automatically zeroized by the OS following restart of the module or power-cycle.	Part of the internal state of the platform DRBG (HMAC_DRBG) following instantiation.

6.2 Non-Deterministic Random Number Generation Specification

6.2.1 Summary

The module supports the option to configure the use of one of two different configurations to support the seeding of the modules software DRBG:

> Intel® Secure Key RDSEED; or

> use of an independent source to seed the module DRBG.

Each option is covered in more detail in the following sections.

Selection of the entropy source during secure initialization is covered in section 10.3, 'Approved Mode of Operation'.

NOTE The module does not support a configuration that allows concurrent input from both RDSEED alongside an independent seed.

Changing the source configuration requires a module reset where all key material previously generated by the Thales CipherTrust Manager Core Security Module will be re-generated on next power-on.

The root-of-trust is automatically selected as the noise source when configured and the noise source is set to `AUTO` as covered in more detail in section 10.3, 'Approved Mode of Operation'.

Further guidance on configuring the entropy source and checking which source is being used can be found in section, 10.3, 'Approved Mode of Operation' and section 10.4.1, 'Checking the configured entropy source'.

6.2.2 Use of Intel® Secure Key RDSEED

When the module is configured to use the Intel® Secure Key RDSEED, this has been certified as part of the module validation to meet the requirements of [SP800-90B] and has been certified using [FIPS 140-2 IG], 7.18 and 7.19.

When using this source, all keys produced by the module have the ability to claim their full security strength.

When using RDSEED, the non-deterministic RNG is used exclusively to feed an approved conditioning function, where in-turn the output of the conditioning function is used to seed the DRBG.

The following table provides summary details for the hardware noise source used by RDSEED:

Table 6-2: RDSEED Specification

Entropy sources	Minimum number of bits of entropy	Details
Feedback stabilized metastable latch.	Full-entropy output	[SP800-90B] compliant Non-Deterministic RNG using a hardware based noise internal to the module boundary. Digitized output from the noise source is fed through an approved conditioning function based on CBC-MAC using AES with a 128-bit key. Raw noise is generated based on feedback stabilized metastable latch which is used to generate entropic binary data by measuring the resolution state of the latch after exiting metastability. The feedback is used to keep the metastable latch balanced so that thermal noise will drive the resolution process. The module achieves full entropy for the output of the conditioning function where every 111-bytes used to seed the DRBG includes 111-bytes of entropy. All outputs from the noise source are subjected to statistical testing ahead of being fed to the conditioning function. The platform DRBG is seeded with 111-bytes every 2^{16} read operations.

6.2.3 Receiving an external entropy seed.

As a second option, the module can be configured to receive seed entropy from a source outside its boundary.

When operating with seed entropy material provided from a source outside the module boundary (i.e. anything other than when Intel ® Secure Key® RDSEED is configured for use) the modules certificate includes the caveat: '*No assurance of the minimum strength of generated keys'*.

In this configuration, assurance for the strength of keys will be dependent on the assurance claims of the HSM used to seed the DRBG.

When in this configuration, the module is operating under scenario 2.b. from [FIPS 140-2 IG], 7.14, 'Entropy Caveats'. During startup an entropy seed is supplied to the module by means of call-back functions and where it is provided with 111-bytes of entropy per seed or re-seed operation from the configured source by an application running internal to the physical boundary of the module but outside its logical boundary. The DRBG will enter an error state and fail to start should the required entropy load on startup not occur.

When using entropy loading, entropy sources used to generate the seed must meet the minimum entropy requirements for seeding the DRBG where the supplied 111-byte seed is expected by the module to contain full entropy in order to maintain a security strength of 256-bit for the DRBG.

6.3 Key Import and Export Methods

Depending on the key and configuration of the module, the following methods of key import and export are available as a service:

> **MKEK - secure transport between module instances**

Different instances of the module exchange the MKEK encrypted using AES in CTR mode with 128-bit MKEK Transport Key and with authenticity provided using HMAC-SHA2-256 and using the MKEK Transport MAC Key. Transport keys are derived from the output of an ECDH [SP800-56Ar3] key derivation using the X9.63 KDF [SP800-135r1] with SHA2-256.

> **User Keys – Export as JWE object**

User keys can be exported as a JWE object [RFC7516] where the user key is encrypted using either:

- AES in either GCM mode; or
- AES in CBC mode with the addition of an HMAC,

using the JWE Content Encryption Key for User Key Export and JWE Content MAC Key for User Key Export.

When using CBC with HMAC the following key length and MAC options are supported:

- AES in CBC mode (128-bit key) with HMAC-SHA2-256;
- AES in CBC mode (192-bit key) with HMAC-SHA2-384; or
- AES in CBC mode (256-bit key) with HMAC-SHA2-512.

The JWE Content Encryption Key for User Key Export is then separately encrypted using either:

- a User RSA Key and RSA and either:
  - KTS-OAEP-basic from [SP800-56Br2]; or
  - RSA with PKCS#1 v1.5 padding from [PKCS #1] as permitted until Dec 31st 2023 under [FIPS 140-2 IG], D.9. After this date, use of this algorithm is disallowed.
- AES in Key Wrap mode (with 128, 192 or 256-bit key) and using a key derived using X9.63 KDF [SP800-135r1] with SHA2-256 and using the shared secret derived using (Cofactor) One-Pass Diffie-Hellman, C(1e, 1s, ECC CDH) and a User EC Key.

The encrypted JWE Content Encryption Key for User Key Export is then packaged alongside the encrypted user key for export.

> **User Key - Import/Export using AES in KW or KWP mode**

Both user symmetric and asymmetric keys can be imported/exported using a User AES Key and AES in KW or KWP mode.

> **User Key - Import/Export using RSA**

User symmetric keys can be imported/exported using a specified User RSA Key and either:

- KTS-OAEP-basic from [SP800-56Br2]; or
- RSA with PKCS#1 v1.5 padding from [PKCS #1] as permitted until Dec 31st 2023 under [FIPS 140-2 IG], D.9. After this date, use of this algorithm is disallowed.

> **User Key - Import/Export over TLS**

The module supports export of user keys over a configured TLS channel. When using this method, confidentiality for the exported key is provided exclusively by the channel.

> **User Key/System Key - Import/Export plaintext using the REST API.**

The module supports the option to directly read and export plaintext CSP to internal applications running within the physical boundary of the module but outside its logical boundary. When using this export method,

no module-enforced protection for the CSP is provided by Thales CipherTrust Manager Core Security Module.

> **NOTE** For the purposes of this section – 'User symmetric and asymmetric keys' are considered to be:
>
> > Symmetric Keys - User AES Key, User Triple-DES Key and User HMAC Key; and
>
> > Asymmetric Keys - User RSA Key and User EC Key.

> **General – Export for module protected keys**

Where keys are protected by Thales CipherTrust Manager Core Security Module and exported for persistent storage by other applications running internal to the physical boundary of the module, keys are either:

- **Option 1 (all keys excluding Account KEK for user keys):** encrypted using AES-256 in GCM mode and either the MKEK or Account KEK for user keys ; or
- **Option 2 (Account KEK for user keys):** encrypted using AES-256 in CBC mode and the Master KEK for user keys.

Keys encrypted using CBC mode without an independent MAC are considered to be plaintext within the scope of this certification.

6.4 Key Zeroization

Key Zeroization is performed using procedural zeroization methods as covered in [FIPS 140-2 IG], 7.9, 'Procedural CSP Zeroization'.

The 'kscfg system reset' command can be used to erase all stored CSPs, including any backup keys. It does this by deleting the files that hold the CSP information. This command also restarts the module which releases any CSPs in RAM.

7 Self-Tests

7.1 Summary

Thales CipherTrust Manager Core Security Module provides both power-on and continuous tests in order to ensure correct operation of the module.

Failure of either power-on or continuous test will trigger the module, and all dependent CipherTrust Manager services, to enter their error state. When in the error state, no cryptographic or key management operations supported by the module are accessible to the user or system.

Recovery from self-test failure is achieved by restarting all services running on the CipherTrust Manager appliance. This triggers a repeat of all power-on tests and where if the detected error persists, the module will re-enter the error state. If the error persists, through multiple restarts, the module must be un-installed and re-installed.

Indicator for self-test failure is based on:

- **Option 1:** Error message output directly to an active session with the module on the platform serial interface;
- **Option 2:** Error message added to the system log.

Error messages transferred to the system audit log maintained by the operating environment for the module can be read by parsing the log located in /**opt/keysecure/logs/keysecure.system.log**. Example messages related to self-test failure are shown in the table below:

```
AES-KAT: CRITICAL SYSTEM ALERT
1 FIPS known answer tests(KAT) failed.
aes-gcm failed for golangCrypto: Failed decrypting cipher text. <additional error details if
exists>
SERVICES ARE GOING DOWN IMMEDIATELY.

- Binary Integrity Test:
CRITICAL SYSTEM ALERT
Checksum mismatch, binary file darkstar, checksum file darkstar.sha512.sum
SERVICES ARE GOING DOWN IMMEDIATELY.

- DRBG:
CRITICAL SYSTEM ALERT
FATAL error: Identical blocks detected at RNG output
SERVICES ARE GOING DOWN IMMEDIATELY.

- PairwiseConsistencyCheck:
CRITICAL SYSTEM ALERT
RSAPairwiseConsistencyCheck: signature invalid
SERVICES ARE GOING DOWN IMMEDIATELY.
```

Figure 7-1: Example log messages relating to self-test failure.

In addition to self-test failures, successful tests are separately recorded in the same log.

All self-test are confirmed as having successfully passed when the nae-kmip platform service is listed as started. This can be identified using **`HTTP GET`** on the **`/api/v1/system/services/status`** endpoint that will output the following:

```
{
    "status": "started",
    "services": [
        {
            "name": "nae-kmip",
            "status": "started"
        },
        {
            "name": "web",
            "status": "started"
        }
    ]
}
```

Figure 7-2: Returned output following successful startup of services including completion of POST.

Should POST fail – the service will enter an error state and be halted before being shut-down. Once shut-down, the service is not available as a valid end-point and will not be listed in the output of above.

Self-tests for cryptography supported by the module are run on all active implementations of a given algorithm during power-up tests (i.e. depending on whether PAA is active or not for AES, the self-test during power-up will test the corresponding active algorithm implementation).

7.2 Power-On Self-Tests

The module performs Power-On Self-Tests (POST) upon power-up to confirm the firmware integrity, and to check the continued correct operation of the random number generator and each of the implemented cryptographic algorithms.

Each of the six executable binaries that make up the module contain their own power-on self-tests that execute on startup. The module integrity test for each binary is run first followed by Known Answer Tests on each of the independent copies of the CipherTrust Manager Core Library Main and CipherTrust Manager Core Library Alt used by the Thales CipherTrust Manager Core Security Module.

While the module is running these tests, all interfaces to it are disabled until the tests successfully complete. If any test fails an error message is output, the module halts, and data output is inhibited.

POST supported by the module are covered in the following tables:

Table 7-1: Power-On Self-Tests (General)

Test Name	Description
Module integrity test	RSA signature check using PKCS#1 PSS with SHA2-512 and using the Binary Validation Key. This check is run on each binary ahead of it subsequently performing further POST.

Table 7-2: Power-On Self-Tests (KAT)

Test Name	CipherTrust Manager Core Library Main	CipherTrust Manager Core Library Alt	Description
KAT – RSA	x	-	Sign / verify PKCS#1 v1.5, sign/verify PKCS#1 PSS.
KAT – PBKDF	x	-	Key derivation KAT using HMAC-SHA2-384 and 480-bit output key.
KAT – DRBG	x	-	HMAC_DRBG – instantiate, generate and reseed tests.
KAT – HMAC	x	x	Message digest KAT for HMAC-SHA1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512.
KAT – SHA2	x	x	Message digest KAT for SHA1, SHA2-256, SHA2-384, SHA2-512.
KAT – SHA3	x	-	Message digest KAT for SHA3-256, SHA3-384 and SHA3-512.
KAT – AES	x	x	The following modes and key sizes are tested: > ECB (256-bit encrypt and decrypt test); > CBC (256-bit encrypt and decrypt test); > CTR (128, 192 and 256-bit encrypt and decrypt test); > GCM (128, 192 and 256-bit encrypt and decrypt tests); > KW (128, 192 and 256-bit encrypt and decrypt tests); and > KWP (128, 192 and 256-bit encrypt and decrypt tests).
KAT – KDA	x	-	The following KDF are tested: > OneStep (key derivation KAT using SHA2-256); and > HKDF (key derivation KAT using HMAC-SHA2-224 as PRF).
KAT – KAS-ECC-SSC	x	x	Key derivation KAT using P-224 with SHA2-224, P-256 with SHA2-256, P-384 with SHA2-384, P-521 with SHA2-512.

Test Name	CipherTrust Manager Core Library Main	CipherTrust Manager Core Library Alt	Description
KAT – KTS-RSA (OAEP)	x	-	The following modulus and hash options are tested: > modulus 2048-bit with SHA2-256 (encrypt and decrypt); > modulus 3072-bit with SHA2-384 (encrypt) and SHA2-512 (encrypt); and > modulus 4096-bit with SHA2-384 (decrypt) and SHA2-512 (decrypt).
KAT – CVL (TLS 1.2 KDF, TLS 1.3 KDF, X9.63 KDF)	x	-	> TLS 1.2 KDF (key derivation test using SHA2-384); > TLS 1.3 KDF (key derivation test using SHA2-256 in PSK and PSK-DHE modes and SHA2-384 in PSK-DHE mode); and > X9.63 KDF (key derivation test using SHA2-256 and SHA2-512).
KAT – Triple-DES	x	x	Variable plaintext/ciphertext KAT for ECB, inverse permutation KAT for CBC.
KAT – ECDSA	x	x	Pairwise consistency test using P-384.

7.3 Conditional Self Tests

The module automatically performs conditional self-tests based on the module operation. These self-tests do not require operator input to initiate.

Table 7-3: Conditional Self-Tests

Test	When Performed
Pairwise consistency test for new RSA keys.	Performed following new RSA key generation. Test is performed for keys generated by the 'CipherTrust Manager Core Library Main'.
Pairwise consistency test for new ECDSA keys.	Performed following new ECDSA key generation. Test is performed for keys generated by both the 'CipherTrust Manager Core Library Main' or 'CipherTrust Manager Core Library Alt'.

Test	When Performed
Noise Source (RDSEED) Continuous Tests	Performed on a continuous basis in hardware on the digitized output from the noise source.
Continuous Random Number Generator Test (CRNGT)	Performed on a continuous basis on the outputs of the DRBG. Test is performed by the 'CipherTrust Manager Core Library Main' hosting the DRBG.

8 Physical Security

The module makes no physical security claims owing to it being a **Software Module**.

9 Mitigation of Other Attacks

No assurance claims are made under this section.

10 Guidance

10.1 Verifying module integrity following delivery

10.1.1 Cipher Trust Manager Appliance

When the module is shipped as an internal component of CipherTrust Manger K470 or K570 appliance, the appliance is shipped pre-loaded with a disk image including the module. Tamper evident features of the product should by checked for evidence of tampering during transit on receipt of the unit.

The primary tamper evident mechanism on the appliance is two individually serialized tamper labels sited on the top and side of the chassis. Any attempts to access the internals of the chassis require removal of the chassis lid which in turn will damage both tamper evident labels. In addition the unit is shipped in an individually serialized tamper evident bag

Serial numbers on labels and the bag are unique and included in the Advanced Shipping Notification (ASN) that is provided to end-user ahead of a unit arriving by courier.

The following figure shows the expected location of tamper labels on the appliance:

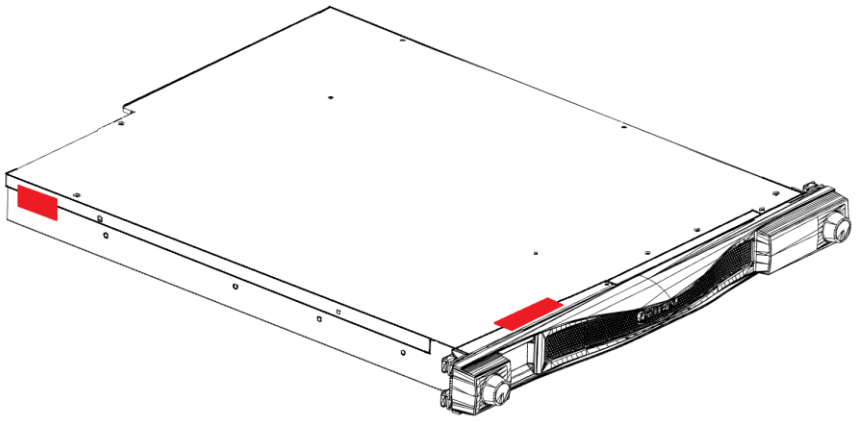


Figure 10-1: CipherTrust Manager tamper label locations (identified in red).

The label is of a multi-layer construction with attempts to remove the label damaging an internal film internal to the label leaving a visible ‘Tamper’ indicator visible. Attempts to re-adhere the label will leave the tamper indicator visible through the surface of the label and will change the visible appearance of the label.

Should any difference be identified between the serial numbers on tamper evidence labels on the CipherTrust Manager received and those recorded on the ASN, the appliance should be considered compromised and Thales contacted immediately.

10.1.2 Virtual CipherTrust Manager Downloads

Where the module is included as a component in the virtual CipherTrust Manage k170v, the required image is downloaded from the Thales customer support portal (https://supportportal.thalesgroup.com).

The image should only be downloaded direct from Thales ensuring the website correctly authenticates when setting up the required https tunnel as being **supportportal.thalesgroup.com**.

10.2 Identifying the Module Software Version

Ahead of putting the module into its approved mode of operation, it is important to identify the CPU and software versions of the target module, and to check these correspond to one of the tested configuration of the modules listed in section 2.3, 'Tested Configurations'.

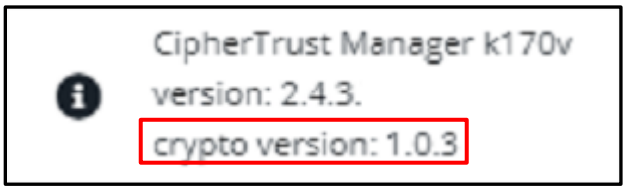
In order to display and verify the hardware CPU model on the CipherTrust Manager appliance, administrator shall open a serial console or SSH session and run `cat /proc/cpuinfo`. The reported CPU model shall be one of the tested modules listed in section 2.3, 'Tested Configuration'.

> **NOTE** This command is supported by CipherTrust Manager as part of the operating environment for Thales CipherTrust Manager Core Security Module.

In addition to checking the CPU number, the software version for the module must be verified to match one of the tested versions in the section 2.3, 'Tested Configuration'. The version can be checked using one of three different paths:

> **CipherTrust Manager GUI**

- Browse to the CipherTrust Manager URL and authenticate with existing user credentials.
- Click the ℹ button on the top of the interface to display the product model and version information.
- Locate and validate the **"crypto version"** value which is the software version for the Thales CipherTrust Manager Core Security Module:



CipherTrust Manager `ksctl` Command Line Utility

- Download the `ksctl` utility from target CipherTrust Manager appliance to a client machine.

Execute `ksctl --url=https://<target IP> --user=admin --nosslverify system info get`. The response JSON document contains `crypto_version` attribute and its value:

> **REST API**

- Perform **HTTP POST** to **/vt/auth/tokens endpoint**, to authenticate and issue an external JWT.

```
curl -k 'https://<CM IP>/api/v1/auth/tokens/' -H 'Content-Type:
application/json' --data-binary $'{"grant_type": "password","username":
"admin","password": "<pwd>"}'
```

- Use the issued JWT token in HTTP Authorization header, to perform a **HTTP GET** on **/v1/system/info** endpoint
- Use the issued JWT token in HTTP Authorization header, to perform a **HTTP GET** on **/v1/system/info** endpoint

```
curl -k 'https://<IP>/api/v1/system/info' -H 'Authorization: Bearer <JWT>'
```

Validate the value of **crypto_version** attribute in the received JSON document.

NOTE The standalone 'version' (e.g. 2.4.3) in the above examples is for the overall software build of the complete CipherTrust Manager product and is not relevant to establishing the version of the Thales CipherTrust Manager Core Security Module which is an internal component of the overall product.

Thales CipherTrust Manager Core Security Module is versioned independent of the overall software for CipherTrust Manager product. Multiple software versions of CipherTrust Manager may share the same version of the Thales CipherTrust Manager Core Security Module which is versioned independently under 'crypto_version'.

Thales CipherTrust Manager Core Security Module is always deployed as part of an overarching software version for CipherTrust manager and cannot be independently loaded or replaced for a given CipherTrust Manager software version.

10.3 Approved Mode of Operation

The module is only in the approved mode of operation when configured as described in Section 10 of this document. The module is in a non-approved mode of operation when not configured as described, or when a non-approved algorithm from Section 2.5 is in use.

To place the cryptographic module into its approved mode of operation the Administrator must configure the entropy source to be used.

This can be done by either:

> **Option 1:** setting the source to be use the Intel® secure key RDSEED; or

> **Option 2:** configuring a separate HSM as the root-of-trust and allowing it to be used as source of entropy for the Thales CipherTrust Manager Core Security Module.

The following steps cover this guidance:

> When selecting to use Intel® secure key RDSEED as the noise source, change the system entropy source to RDSEED using the command **`kscfg system entropy-source -s RDSEED`** followed by **`kscfg system reset -y`** to implement the change. This fixes RDSEED as the only noise source the module can use. Output from running the two commands is shown below:

```
$ kscfg system entropy-source -s RDSEED

Note: please run "sudo systemctl restart keysecure" to have new entropy
source effective in CipherTrust Manager

$ kscfg system reset -y
```

Figure 10-2: Configuring Entropy source as Intel® RDSEED using `kscfg`.

NOTE Configuring Intel® secure key RDSEED as the noise source using the above method will ensure the source is used by the platform including when the a separate root-of-trust is configured for the module.

> If a separate root-of-trust is present, and it is intended for this to act as the noise source, the Administrator shall:

- check the configured noise source has been left at its default of **`AUTO`**. If it needs changed this can be be achieved using **`kscfg system entropy-source -s AUTO`**
- make sure an HSM partition is created, initialized, and ready to be used as the root-of-trust; and
- configure the module to use the root-of-trust via CipherTrust Manager GUI, CLI, or by executing a POST to **`/v1/system/hsm/setup`** endpoint providing the partition information and credentials for the target HSM partition.

Following the configuration of the root-of-trust, it is automatically used as the noise source for the Thales CipherTrust Manager Core Security Module when the entropy source is set to **`AUTO`**.

After the configuration of the root-of-trust, the end-user can check the configured entropy source as covered in section 10.4.1, ‘Checking the configured entropy source’.

NOTE Full guidance on setting up the root-of-trust is covered in the online product deployment guide available at https://www.thalesdocs.com/ctp/cm/2.4/.

NOTE Following delegation of the root-of-trust, the configured device must be present at subsequent power-on of the Thales CipherTrust Manager Core Security Module in order for the module to successfully power-on and initialize including seeding of the platform DRBG.

During power-on, should the DRBG fail to seed from the configured root-of-trust it will enter its

error state consistent with power-up self-test failure.

NOTE Changing the module entropy source or configuring a root-of-trust will trigger Thales CipherTrust Manager Core Security Module to regenerate all module CSP.

NOTE The module does not support an atomic FIPS approved mode indicator.

Following configuration for an approved mode of operation, confirmation that settings required to achieve a FIPS approved mode of operation can be checked using guidance in Section 10.4.1, 'Checking the configured entropy source'.

Independent of configuring the entropy source, to be in an approved mode of operation, security rules as covered in Section 10.6, 'Security Rules' must separately be followed.

NOTE In the FIPS approved mode of operation, controls over the use of non-approved algorithms (as covered in Section 2.5) are procedural.

10.4 Module Status

10.4.1 Checking the configured entropy source

On every startup of the service responsible for Thales CipherTrust Manager Core Security Module RNG, the configured entropy source is output to CipherTrust Manager system logs.

Following initial secure configuration of the module, the configuration can be checked using tools available in the operational environment by:

- **Option 1:** downloading the CipherTrust Manager System Logs using the CipherTrust Manager GUI. Once downloaded and unpackaged, the file **keysecure.system.log** can be reviewed for the entries including the phrase `msg:Entropy source`; or
- **Option 2:** reviewing the logs directly over SSH using the **`ksadmin`** CipherTrust Manager role. From the terminal, the command `grep "darkstar |.*Entropy"` can be used to find the relevant log entry.

Below are a sample outputs using Option 2 when the Intel® secure key RDSEED (Figure 10-3) or Root-of-Trust (Figure 10-4) are used:

```
ksadmin@ciphertrust:~$ grep "darkstar |.*Entropy"
/opt/keysecure/logs/keysecure.system.log
2021-05-12 09:55:38 | darkstar | level:INF time:2021-05-12T09:55:38.299Z
msg:Entropy source - Intel RDSEED name:entropy
```

Figure 10-3: keysecure.system.log output showing Entropy source as Intel® Secure Key RDSEED.

```
ksadmin@ciphertrust:~$ grep "darkstar |.*Entropy"
/opt/keysecure/logs/keysecure.system.log
2021-05-12 09:55:38 | darkstar | level:INF time:2021-05-12T09:55:38.299Z
msg:Entropy source - HSM name:entropy
```

Figure 10-4: keysecure.system.log output showing Entropy source as HSM (configured root-of-trust).

10.5 Key Import and Export

Key import/export is performed using one of the methods covered in section 6.3, ‘Key Import and Export Methods’.

For detailed guidance on using each method, review the user guidance for CipherTrust Manager, which can be found online at: https://www.thalesdocs.com/ctp/cm/2.4/.

When importing a key using the NAE-XML or KMIP API, keys are assigned to the user performing the import unless explicitly configured to be available to a group. The module uses attribute based access control of objects with enforcement and management of policies performed by the module Groups are configured and managed outside the logical boundary of Thales CipherTrust Manager Core Security Module.

When importing keys using the NAE-XML interface, if exclusively using client certificates for authentication to TLS but separate authentication against a specific user role hasn’t been performed, imported keys will be available to all users.

10.6 Security Rules

The following security rules are recorded for all roles supported by the module:

- When using TLS to provide a client to module trusted channel, authentication keys for the client must not use a security strength of less than 112-bits for certificate signing.
- Use is permitted for TLS when configured to use TLS 1.2 and TLS 1.3 exclusively. By default the module has its latent capability to support TLS 1.0 and 1.1 disabled. The module must not be configured to allow the use TLS 1.0 or TLS 1.1 in the approved mode of operation.
- The GCM mode of AES for encrypt operations is not permitted for use in the approved mode of operation for user services when supplied with an externally provided IV.

  User services using GCM to encrypt or wrap data must exclusively use IV generated by the modules approved DRBG and where these must have a minimum length of 96-bits.

  Use of external IV with GCM is exclusively permitted for decryption operations or where the GCM is used to support the modules TLS implementation used to protect connections to the modules KMIP and NAE-XML API.
- In conformance with limitations on the maximum number of blocks encrypted for a given 168-bit Triple-DES key outlined in SP800-67r1, and further tightened in [FIPS 140-2 IG], A.13, 'SP 800-67rev1 transition', users are responsible for enforcing that any given Triple-DES key stored in the cryptographic module cannot be used for more than 2^{16} 64-bit data block encryption operations.

  Thales CipherTrust Manager Core Security Module does not technically enforce a limit on the number of times a Triple-DES key can be used.
- Use of keys derived using PBKDF is exclusively permitted for wrapping of CSP for storage. This is based on explicit scope defined for PBKDF as covered in [SP800-132]. When using PBKDF to derive storage keys, it is critically important to ensure that the User Key Wrapping PWD supplied to module contains sufficient entropy for the intended purpose of the derived key. Specific guidance on security considerations when using PBKDF is provided in Appendix A, 'Security Consideration' to [SP800-132].

Users must not use any of the algorithms listed in section 2.5, 'Non-Approved Algorithms' when in the approved mode of operation. Listed algorithms will be available when operating in the approved configuration where enforcement against use is procedural only.

- RSA based encryption, as available with the Encrypt Data and Decrypt Data services, must exclusively be used for encrypting externally supplied key objects for transport. Use of RSA for data encryption is not permitted in an approved mode of operation for any FIPS 140-2 approved cryptographic module.