Google, LLC

Tensor UFS Inline Storage Encryption Cryptographic Module

FIPS 140-3 Non-Proprietary Security Policy

Table of Contents

1 General 5
1.1 Overview 5
1.2 Security Levels 5
2 Cryptographic Module Specification 5
2.1 Description 5
2.2 Tested and Vendor Affirmed Module Version and Identification 7
2.3 Excluded Components 8
2.4 Modes of Operation 8
2.5 Algorithms 8
2.6 Security Function Implementations 9
2.7 Algorithm Specific Information 10
2.8 RBG and Entropy 10
2.9 Key Generation 10
2.10 Key Establishment 10
2.11 Industry Protocols 10
3 Cryptographic Module Interfaces 10
3.1 Ports and Interfaces 10
4 Roles, Services, and Authentication 11
4.1 Authentication Methods 11
4.2 Roles 11
4.3 Approved Services 11
4.4 Non-Approved Services 13
4.5 External Software/Firmware Loaded 13
5 Software/Firmware Security 13
5.1 Integrity Techniques 13
5.2 Initiate on Demand 13
6 Operational Environment 13
6.1 Operational Environment Type and Requirements 13
7 Physical Security 14
7.1 Mechanisms and Actions Required 14
8 Non-Invasive Security 14
9 Sensitive Security Parameters Management 14
9.1 Storage Areas 14
9.2 SSP Input-Output Methods 14
9.3 SSP Zeroization Methods 15

This document may be freely reproduced and distributed whole and intact including this Copyright Notice

9.4 SSPs 15
10 Self-Tests 16
10.1 Pre-Operational Self-Tests 16
10.2 Conditional Self-Tests 16
10.3 Periodic Self-Test Information 17
10.4 Error States 18
11 Life-Cycle Assurance 18
11.1 Installation, Initialization, and Startup Procedures 18
11.2 Administrator Guidance 18
11.3 Non-Administrator Guidance 18
12 Mitigation of Other Attacks 18

This document may be freely reproduced and distributed whole and intact including this Copyright Notice

List of Tables

Table 1: Security Levels 5
Table 2: Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets) 7
Table 3: Tested Module Identification – Hybrid Disjoint Hardware 7
Table 4: Tested Operational Environments - Software, Firmware, Hybrid 8
Table 5: Modes List and Description 8
Table 6: Approved Algorithms 9
Table 7: Security Function Implementations 9
Table 8: Ports and Interfaces 11
Table 9: Roles 11
Table 10: Approved Services 12
Table 11: Mechanisms and Actions Required 14
Table 12: Storage Areas 14
Table 13: SSP Input-Output Methods 15
Table 14: SSP Zeroization Methods 15
Table 15: SSP Table 1 15
Table 16: SSP Table 2 15
Table 17: Pre-Operational Self-Tests 16
Table 18: Conditional Self-Tests 17
Table 19: Pre-Operational Periodic Information 17
Table 20: Conditional Periodic Information 17
Table 21: Error States 18

List of Figures

Figure 1. Module’s Block Diagram 7

This document may be freely reproduced and distributed whole and intact including this Copyright Notice

1 General

1.1 Overview

This document is the non-proprietary FIPS 140-3 Security Policy for Tensor UFS Inline Storage Encryption Cryptographic Module. It contains a specification of the rules under which the module must operate and describes how this module meets the requirements as specified in FIPS PUB 140-3 (Federal Information Processing Standards Publication 140-3) for a Security Level 1 Software-Hybrid cryptographic module. This security policy is for the validation of the Tensor G2/G3/G4 UFS Inline Storage Encryption Cryptographic Module.

In this document, the terms “cryptographic module” or “module” are used interchangeably to refer to the Tensor UFS Inline Storage Encryption Cryptographic Module with Software version 2.0.0 and Hardware version 4.1.0 or 4.2.1.

1.2 Security Levels

Section	Title	Security Level
1	General	1
2	Cryptographic module specification	1
3	Cryptographic module interfaces	1
4	Roles, services, and authentication	1
5	Software/Firmware security	1
6	Operational environment	1
7	Physical security	1
8	Non-invasive security	N/A
9	Sensitive security parameter management	1
10	Self-tests	1
11	Life-cycle assurance	1
12	Mitigation of other attacks	N/A
	Overall Level	1

Table 1: Security Levels

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

The module is a multi-chip standalone Software-Hybrid module designed for on-the-fly hardware encryption for a flash storage device. The module’s ID/name can be correlated to the module’s software component and hardware component within Module’s boundary, as detailed below.

- UFS Pixel FIPS CMVP Module (Module’s Software Component, version 2.0.0)
- UFS ISE (Module’s Hardware Component, version 4.1.0 on Google Pixel 7 with Tensor G2 processor; version 4.2.1 on Google Pixel 8 with Tensor G3 processor; version 4.2.1 on Google Pixel 9 with Tensor G4 processor)

This document may be freely reproduced and distributed whole and intact including this Copyright Notice

The UFS Pixel FIPS CMVP Module (hereafter referred to as ISE Driver) is the software component of the cryptographic module running in the Linux Kernel, which calls the cryptographic algorithms for the module's pre-operational self-tests, and also sets the FIPS status of the entire cryptographic module once the pre-operational self-test is completed successfully.

The UFS ISE is the hardware component of the cryptographic module that supports AES-XTS encryption and decryption. This hardware component located between the device's DRAM and the Flash Storage Device, which can provide inline encryption/decryption while maintaining device performance (such as responsiveness and power consumption). Please see Figures 1 below for more information.

Module Type: Software-hybrid

Module Embodiment: Multi-Chip Standalone

Cryptographic Boundary:

Module is defined as a multi-chip standalone software-hybrid module, with the boundary of the Tested Operational Environment's Physical Perimeter (TOEPP) being defined as the physical perimeter of the tested platform enclosure around which everything runs. Figure 1 below illustrates a logical cryptographic boundary and physical boundary of the module. The module's cryptographic boundary contains module's executable file ufs-pixel-fips140.ko. The module is intended only for single-process execution.

Tested Operational Environment's Physical Perimeter (TOEPP):

The boundary of the TOEPP is defined as the entire chassis unit's physical perimeter encompassing the "top," "front," "left," "right," "rear" and "bottom" surfaces of the tested platform.

In the following figures, the bidirectional arrows depict the flow of the status, control and data within the device's Tested Operational Environment's Physical Perimeter (TOEPP). The TOEPP in the block diagram contains the module (the blue dotted region). The operations within the module's cryptographic boundary (the blue dotted region) use the cipher from the UFS ISE (Module's hardware component) that is included in the module's cryptographic boundary. The UFS Pixel FIPS CMVP Module (Module's software component) is only used at module's initialization to perform the pre-operational self-tests.

This document may be freely reproduced and distributed whole and intact including this Copyright Notice

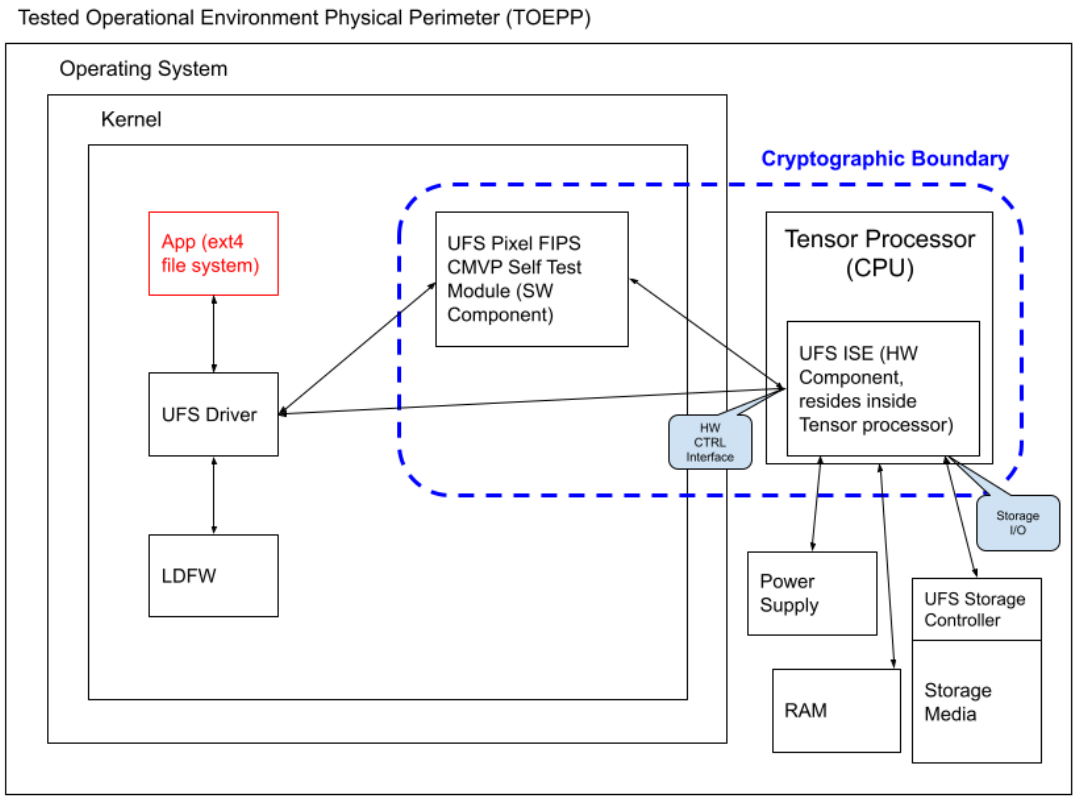


Figure 1. Module's Block Diagram

2.2 Tested and Vendor Affirmed Module Version and Identification

Tested Module Identification – Hardware:

N/A for this module.

Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets):

Package or File Name	Software/ Firmware Version	Features	Integrity Test
ufs-pixel-fips140.ko	2.0.0		HMAC-SHA-256

Table 2: Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets)

Tested Module Identification – Hybrid Disjoint Hardware:

Model and/or Part Number	Hardware Version	Firmware Version	Processors	Features
Google Pixel 7	4.1.0		Google Tensor G2	
Google Pixel 8	4.2.1		Google Tensor G3	
Google Pixel 9	4.2.1		Google Tensor G4	

Table 3: Tested Module Identification – Hybrid Disjoint Hardware

Tested Operational Environments - Software, Firmware, Hybrid:

This document may be freely reproduced and distributed whole and intact including this Copyright Notice

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Linux Kernel 6.12	Google Pixel 7	Google Tensor G2	Yes	N/A	2.0.0
Linux Kernel 6.12	Google Pixel 8	Google Tensor G3	Yes	N/A	2.0.0
Linux Kernel 6.12	Google Pixel 9	Google Tensor G4	Yes	N/A	2.0.0

Table 4: Tested Operational Environments - Software, Firmware, Hybrid

Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid:

N/A for this module.

CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when so ported if the specific operational environment is not listed on the validation certificate.

2.3 Excluded Components

There are no components within the cryptographic boundary excluded from the FIPS 140-3 Requirements.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved Mode of Operation	The module is designed with approved mode of operation only	Approved	Equivalent to the indicator of the requested service as defined in section 4.3

Table 5: Modes List and Description

The module always runs in the Approved Mode of Operation and does not implement any Non-Approved Security Functions. The module doesn't support degraded operational mode.

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A2937	Key Length - 256	SP 800-38A
AES-CBC	A4645	Key Length - 256	SP 800-38A
AES-XTS Testing Revision 2.0	A2937	Key Length - 256	SP 800-38E

This document may be freely reproduced and distributed whole and intact including this Copyright Notice

Algorithm	CAVP Cert	Properties	Reference
AES-XTS Testing Revision 2.0	A4645	Key Length - 256	SP 800-38E
HMAC-SHA2-256	A8225	Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
SHA2-256	A8225	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4

Table 6: Approved Algorithms

Vendor-Affirmed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms with No Security Claimed:

N/A for this module.

Non-Approved, Not Allowed Algorithms:

N/A for this module.

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
AES-XTS encrypt/decrypt	BC-UnAuthDecrypt BC-UnAuthEncrypt	Perform AES-XTS encryption or decryption		AES-CBC: (A2937, A4645) Key Length: 256 bits AES-XTS Testing Revision 2.0: (A2937, A4645) Key Length: 256 bits
Software integrity test	MAC	Software integrity test		SHA2-256: (A8225) HMAC-SHA2-256: (A8225)

Table 7: Security Function Implementations

This document may be freely reproduced and distributed whole and intact including this Copyright Notice

2.7 Algorithm Specific Information

The length of a single data unit encrypted with the AES-XTS shall not exceed 2^{20} AES blocks, that is, 16 MiB of data.

AES-XTS is only approved for storage purposes. It shall not be used for other purposes, such as the encryption of data in transit. To meet the requirement stated in IG C.I, the module implements a check that ensures, before performing any cryptographic operation, that the two AES keys used in AES-XTS mode are not identical. As the module does not generate symmetric keys, the check is performed when keys are input the service APIs. Key_1 and Key_2 shall be generated and/or established independently according to the rules for component symmetric keys from NIST SP 800-133rev2, Sec. 6.3.

2.8 RBG and Entropy

N/A for this module

2.9 Key Generation

The module does not provide any key generation service or perform any key generation for any of its Approved algorithms. Keys are instead provided by applications and stored in memory location outside of the cryptographic module boundary, but still within the Tested Operational Environment's Physical Perimeter (TOEPP) of the tested platform. The module does not support any key establishment methods or asymmetric algorithms and hence no key generation services for them.

2.10 Key Establishment

The module does not support manual key entry or key output. SSP (AES-XTS Key is the only SSP) can only be exchanged via the TOEPP of the device. SSP is entered to module per the request from the module's calling application running on the same tested platform. Keys/SSPs are electronically entered into the module via Module's API in plaintext form. The Module doesn't output the SSPs.

2.11 Industry Protocols

The module doesn't implement Industry Protocols.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

This document may be freely reproduced and distributed whole and intact including this Copyright Notice

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	Arguments used or processed by Module's software component
N/A	Data Output	Output data returned to calling function
N/A	Control Input	Arguments used to control Module's software component
N/A	Status Output	Return values from the Module's software component API
HW CTRL interface, Storage I/O interface	Data Input	Arguments input used or processed by Module's hardware component
HW CTRL interface, Storage I/O interface	Data Output	Data output returned to calling function
HW CTRL interface	Control Input	Control signal for Hardware cryptographic services
HW CTRL interface	Status Output	Return values from the Module's hardware component
N/A	Control Output	N/A
Power Interface	Power	Module's hardware component power supply

Table 8: Ports and Interfaces

The module provides its logical interfaces via Application Programming Interface (API) calls. The logical interfaces provided by the module are mapped onto the FIPS 140-3 interfaces (data input, data output, control input, control output and status output.

4 Roles, Services, and Authentication

4.1 Authentication Methods

N/A for this module. The module does not implement authentication.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	Crypto Officer	None

Table 9: Roles

The module supports Crypto Officer (CO). The cryptographic module does not provide any authentication methods. The module does not allow concurrent operators. The Crypto Officer is implicitly assumed based on the service requested.

4.3 Approved Services

This document may be freely reproduced and distributed whole and intact including this Copyright Notice

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Show status	Module's current status	N/A	API command to check the status	Module's current operational status	None	Crypto Officer
Show version	Module's version	N/A	API command to get module's version	Module's ID and component's versions (SW and HW)	None	Crypto Officer
Perform self-tests	Perform self-tests (pre-operational self-tests and Conditional Self-Tests)	Self-test Passed or Failed status	On-demand self-test (Power cycling)	Self-test Passed or Failed status. Failure self-test results in kernel panic	Software integrity test	Crypto Officer
Perform AES-XTS encryption or decryption	Perform AES-XTS encryption or decryption	Symmetric encryption or decryption successful completion status log (e.g., Encryption verification passed or Decryption verification passed)	Key and message (plaintext message for Encryption or cipher text for decryption)	Encrypted message or Decrypted message	AES-XTS encrypt/decrypt	Crypto Officer - AES-XTS key: W,E
Perform zeroization	Zeroize all SSPs stored within the module	N/A	Power down the tested platform	Zeroized all SSPs stored within the module	None	Crypto Officer - AES-XTS key: Z

Table 10: Approved Services

Please note that the module supports which approved services in an approved manner by default. Non-approved services are not implemented, Thus, an implicit indication via the successful completion of a service is sufficient, for example the following logs show the successful completion of AES-XTS with 256 bits key:

[T1] fscrypt: AES-256-XTS using blk-crypto (native);

This document may be freely reproduced and distributed whole and intact including this Copyright Notice

[T544] lwis lwis-g3aa: Base Probe: Success

To view more kernel logs, the end User shall follow up the steps below.

- adb bugreport my_bugreport.zip
- Unzip my_bugreport.zip.
- Inspect dumpstate_board.txt or the main bug report text file (bugreport-BUILD-DATE.txt).
- Search for sections labeled ------ KERNEL LOG (dmesg) ------ or check the FS/sys/fs/pstore/ directory inside the archive for crash/reboot logs (console-ramoops).

4.4 Non-Approved Services

N/A for this module. The module does not implement any non-approved, not allowed algorithms; therefore, it also does not provide any non-approved services.

4.5 External Software/Firmware Loaded

The module doesn’t support external software load service.

5 Software/Firmware Security

5.1 Integrity Techniques

To ensure software security, a software integrity test is performed on the runtime image of the module. The HMAC-SHA2-256 (HMAC Cert. #A8225) implemented in the module is used as an approved algorithm for the integrity test. If the test fails, the module enters an error state where no cryptographic services are provided, and data output is prohibited. The module is provided in the form of binary executable code.

5.2 Initiate on Demand

Software Integrity test is performed as part of the Pre-operational self-tests. It is automatically executed at power-on. Thus, it can be invoked by rebooting the tested platform.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

The module operates in a modifiable operational environment per FIPS 140-3 Security Level 1 Specifications. The operating system is restricted to a single operator mode of operation. The procurement, build and configuring procedure are controlled. The module is installed into a commercial production grade mobile device. The external application that makes calls to the cryptographic module is the single instance of the cryptographic module, even when the application is serving multiple clients.

This document may be freely reproduced and distributed whole and intact including this Copyright Notice

7 Physical Security

7.1 Mechanisms and Actions Required

Mechanism	Inspection Frequency	Inspection Guidance
Production grade components	N/A	N/A

Table 11: Mechanisms and Actions Required

The Module is a software-hybrid module that operates on a multi-chip standalone platform, which conforms to the Level 1 requirements for physical security. All disjoint components of the module are entirely contained within the production-grade enclosure of the host platform, which blocks physical access to the module. The tested platform (mobile device) shall comprise production grade components with standard passivation (a sealing coat applied over the chip circuitry to protect it against environmental and other physical damage) and a production grade enclosure that completely surrounds the cryptographic module.

8 Non-Invasive Security

The requirements of this section are not applicable to the module.

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
Hardware register	Registers used explicitly for key storage within UFS ISE which are inaccessible outside hardware	Dynamic
RAM	Temporary storage for SSPs used by the module as part of service execution.	Dynamic

Table 12: Storage Areas

The module does not provide persistent keys/SSPs storage. After the SSP (AES-XTS Key is the only SSP) is entered to the module via the Module's API, the module temporarily stores it in the Module's hardware register. The Module's hardware register is internal to the hardware module and is not shared with any external component (operating system or other hardware). No process other than the module itself can access the keys/SSPs in its memory.

9.2 SSP Input-Output Methods

This document may be freely reproduced and distributed whole and intact including this Copyright Notice

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
API Input via TOEPP path	Operator calling application (TOEPP)	Module	Plaintext	Manual	Electronic	

Table 13: SSP Input-Output Methods

The module does not support manual key entry or key output. SSP (AES-XTS Key is the only SSP) can only be exchanged via a DMA descriptor inside the TOEPP of the device. All SSPs are entered to module per the request from the module's calling application running on the same tested platform. Keys/SSPs are electronically entered into the module via Module's API in plaintext form. The Module doesn't output the SSPs.

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Power cycle	Hardware register and RAM quickly lose their stored values upon power-off	Power-off tested platform will erase all SSPs stored within the module	Power off tested platform

Table 14: SSP Zeroization Methods

All SSPs are zeroized when the system is powered down. Input and output interfaces are inhibited while zeroization is performed. The successful act of powering off the module serves as the implicit indicator of zeroization.

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES-XTS key	Used for Symmetric Encryption and Decryption	256-bits - 256-bits	XTS Symmetric Key - CSP			AES-XTS encrypt/decrypt

Table 15: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES-XTS key	API Input via TOEPP path	Hardware register:Plaintext RAM:Plaintext	While the module is active	Power cycle	

Table 16: SSP Table 2

This document may be freely reproduced and distributed whole and intact including this Copyright Notice

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A8225)	HAMC-SHA2-256	MAC verification	SW/FW Integrity	Module is in normal operation	Module performs HMAC-SHA2-256 self-test prior to firmware integrity test

Table 17: Pre-Operational Self-Tests

When the module is loaded or instantiated (after being powered off, rebooted, etc.), the module runs pre-operational self-tests. The operating system is responsible for the initialization process and loading of the library. The module is designed with a default entry point (DEP) which ensures that the self-tests are initiated automatically when the module is loaded. Prior to the module providing any data output via the data output interface, the module would perform and pass the pre-operational self-tests. Following the successful pre-operational self-tests, the module would execute the Conditional Cryptographic Algorithm Self-tests (CASTs).
The self-test success or failure is output as a return value of the library load API call, which is functioning as the self-test status indicator. If one of the self-tests fails, the module transitions into an error state and outputs the error message via the module's status output interface. While the module is in the error state, all data through the data output interface and all cryptographic operations are disabled. The error state can only be cleared by reloading the module. All self-tests must be completed successfully before the module transitions to the operational state.

The module software integrity test parameters are configured at build time and then executed at runtime. As the module supports KASLR and is not loaded as a contiguous block, a series if begin/end addresses (__fips140_text_start/__fips140_text_end and __fips140_rodata_start/__fips140_rodata_end) are used to determine the code that must be checked. The HMAC value (ufs_pixel_fips_hmac_key) of that code is then calculated when building the ELF, storing the calculated digest (ufs_pixel_fips_hmac_expected) in the module. The driver is a kernel loadable module included in the vendor image for the kernel and loaded once the ramdisk holding all the kernel modules has been unpacked into memory.
Once the module is loaded, the HMAC-SHA2-256 is calculated over the entire area specified as part of the build process. This is then compared to the stored value (ufs_pixel_fips_hmac_expected) from the build time calculation. A non-match will place the module into the error state.

The Module conducts HMAC-SHA2-256 KAT self-test before the integrity test is performed.

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA2-256 KAT (A8225)	HMAC-SHA2-256	KAT	CAST	Module is in normal state	HMAC-SHA2-256 KAT	Power Up

This document may be freely reproduced and distributed whole and intact including this Copyright Notice

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-XTS Encrypt KAT (A2937)	256-bits	KAT	CAST	Module is in normal state	AES-XTS Encrypt KAT	Power Up
AES-XTS Decrypt KAT (A2937)	256 bits	KAT	CAST	Module is in normal state	AES-XTS Decrypt KAT	Power Up
AES-XTS Encrypt KAT (A4645)	256-bits	KAT	CAST	Module is in normal state	AES-XTS Encrypt KAT	Power Up
AES-XTS Decrypt KAT (A4645)	256 bits	KAT	CAST	Module is in normal state	AES-XTS Decrypt KAT	Power Up

Table 18: Conditional Self-Tests

The module conducts the CASTs before the first operational use of the cryptographic algorithm.

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A8225)	MAC verification	SW/FW Integrity	Recommend 60 Days	Reboot the tested platform

Table 19: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 KAT (A8225)	KAT	CAST	Recommend every 60 Days	Reboot
AES-XTS Encrypt KAT (A2937)	KAT	CAST	Recommend every 60 Days	Reboot
AES-XTS Decrypt KAT (A2937)	KAT	CAST	Recommend every 60 Days	Reboot
AES-XTS Encrypt KAT (A4645)	KAT	CAST	Recommend every 60 Days	Reboot
AES-XTS Decrypt KAT (A4645)	KAT	CAST	Recommend every 60 Days	Reboot

Table 20: Conditional Periodic Information

The module performs on-demand self-tests initiated by the operator, by powering off and powering the module back on. The full suite of self-tests is then executed. The same procedure may be employed by the operator to perform periodic self-tests.

This document may be freely reproduced and distributed whole and intact including this Copyright Notice

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Error State	If the self-test tests fail, the module is put into an error state	Self-test failure	Reboot the module	System Halt (Kernel panic)

Table 21: Error States

In the Error State, the output interface is inhibited, and the module accepts no more inputs or requests (as the module is no longer running).

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

The module is enabled by default (and hence used automatically) as part of the device without any user configuration. The module always runs in the Approved Mode of Operation and does not implement any Non-Approved Security Functions. When the module is loaded or instantiated (after being powered off, rebooted, etc.), the module runs pre-operational self-tests without any operator intervention. The Module will be operated in an approved mode of operation when pre-operational self-tests have completed successfully.

The module is provided directly to solution developers and is not intended for direct download by the general public. The module is installed on an operating system (Linux kernel 6.12) specified in Section 2.

Additional Rules of Operation:
1. The module does not support concurrent operators.
2. The operating system is responsible for multitasking operations so that other processes cannot access the address space of the process containing the module.
3. The end user of the operating system is also responsible for zeroizing SSPs via wipe/secure delete procedures.

11.2 Administrator Guidance

N/A for this module.

11.3 Non-Administrator Guidance

N/A for this module.

12 Mitigation of Other Attacks

The module does not support Mitigation of Other Attacks. Thus, the security requirements from Section Mitigation of Other Attacks in FIPS 140-3 are not applicable.

This document may be freely reproduced and distributed whole and intact including this Copyright Notice

This document may be freely reproduced and distributed whole and intact including this Copyright Notice