SAMSUNG

Samsung Electronics Co., Ltd.

Samsung Kernel 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 6
2.1 Description 6
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 11
3.1 Ports and Interfaces 11
4 Roles, Services, and Authentication 12
4.1 Authentication Methods 12
4.2 Roles 12
4.3 Approved Services 12
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 14
6 Operational Environment 15
6.1 Operational Environment Type and Requirements 15
7 Physical Security 16
8 Non-Invasive Security 17
9 Sensitive Security Parameters Management 18
9.1 Storage Areas 18
9.2 SSP Input-Output Methods 18

9.3 SSP Zeroization Methods 18
9.4 SSPs 18
9.5 Transitions 19
10 Self-Tests 19
10.1 Pre-Operational Self-Tests 19
10.2 Conditional Self-Tests 20
10.3 Periodic Self-Test Information 21
10.4 Error States 22
11 Life-Cycle Assurance 24
11.1 Installation, Initialization, and Startup Procedures 24
11.2 Administrator Guidance 24
11.3 Non-Administrator Guidance 24
12 Mitigation of Other Attacks 25

List of Tables

Table 1: Security Levels 5
Table 2: Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets) 7
Table 3: Tested Operational Environments - Software, Firmware, Hybrid 7
Table 4: Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid 8
Table 5: Modes List and Description 8
Table 6: Approved Algorithms 9
Table 7: Non-Approved, Not Allowed Algorithms 9
Table 8: Security Function Implementations 10
Table 9: Ports and Interfaces 11
Table 10: Roles 12
Table 11: Approved Services 13
Table 12: Non-Approved Services 13
Table 13: Storage Areas 18
Table 14: SSP Input-Output Methods 18
Table 15: SSP Zeroization Methods 18
Table 16: SSP Table 1 19
Table 17: SSP Table 2 19
Table 18: Pre-Operational Self-Tests 19
Table 19: Conditional Self-Tests 21
Table 20: Pre-Operational Periodic Information 21
Table 21: Conditional Periodic Information 22
Table 22: Error States 22

List of Figures

Figure 1: Block Diagram 7

1 General

1.1 Overview

This document is a non-proprietary FIPS 140-3 Security Policy for the Samsung Kernel Cryptographic Module. It contains the security 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 an overall Security Level 1 module.

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	N/A
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 Samsung Kernel Cryptographic Module (hereafter referred to as “the module” or SKC) is a software module running on a multi-chip standalone general-purpose computing platform. The module provides cryptographic services to Kernel through an application program interface (API). The binary image that contains the SKC for the appropriate platform is **boot.img**.

Module Type: Software

Module Embodiment: Multi-Chip Standalone

Cryptographic Boundary:

The module is defined as a multi-chip standalone software module. Figure 1 below illustrates a logical cryptographic boundary and physical boundary of the module. The module’s cryptographic boundary contains **.rodata**, **.text** and **.init.text** sections of the object files: **fips140_integrity.o**, **fips140_post.o**, **fips140_test.o**, **fips140_out.o**, **fips140_3_services.o**, **api.o**, **cipher.o**, **algapi.o**, **scatterwalk.o**, **skcipher.o**, **ahash.o**, **shash.o**, **hmac.o**, **sha1_generic.o**, **sha256_generic.o**, **sha512_generic.o**, **ecb.o**, **cbc.o**, **aes_generic.o**, **aes.o**, **sha256.o**, **sha1.o**, **aes-ce-core.o**, **aes-ce-glue.o**, **aes-ce.o**, **aes-glue-ce.o**, **sha256-core.o**, **sha256-glue.o**, **sha2-ce-core.o**, **sha2-ce-glue.o**, **sha1-ce-core.o**, **sha1-ce-glue.o**.

The object files are generated from the sources through the kernel build process.

The module is intended only for single-process execution.

Tested Operational Environment’s Physical Perimeter (TOEPP):

The module’s Tested Operational Environment’s Physical Perimeter (TOEPP) is defined as the physical perimeter of the tested platform enclosure around which everything runs.

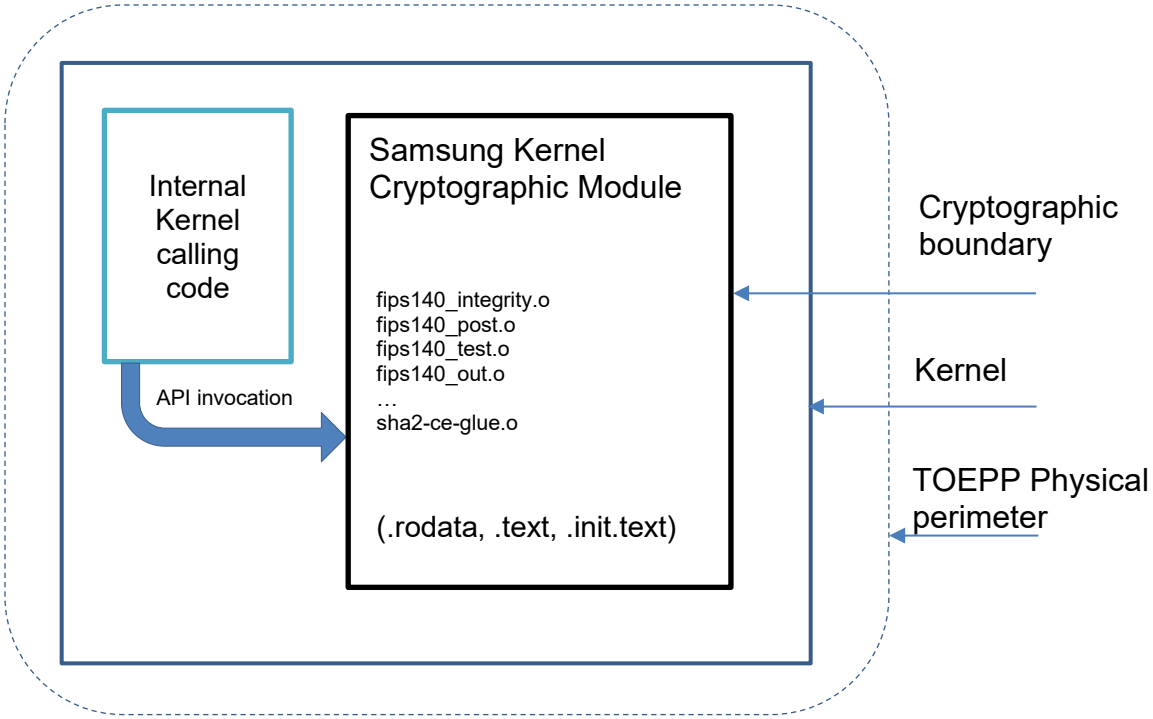

Figure 1: Block Diagram

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
boot.img	2.5		HMAC-SHA2-256

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

Tested Module Identification – Hybrid Disjoint Hardware:

N/A for this module.

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Linux Kernel 6.6	Samsung Galaxy S25	Qualcomm Snapdragon 8 Elite	Yes		2.5
Linux Kernel 6.6	Samsung Galaxy S25	Qualcomm Snapdragon 8 Elite	No		2.5

Table 3: Tested Operational Environments - Software, Firmware, Hybrid

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

Operating System	Hardware Platform
Linux Kernel 6.6	Samsung Electronics Exynos 1580 running on Galaxy Tab S10 FE 5G
Linux Kernel 6.6	Qualcomm Snapdragon 8 Elite running on Galaxy Z Fold7

Table 4: Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid

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 excluded from the security requirements.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved Mode	Automatically entered whenever an approved service is requested	Approved	Equivalent to the indicator of the requested service
Non-Approved Mode	Automatically entered whenever a non-approved service is requested	Non-Approved	Equivalent to the indicator of the requested service

Table 5: Modes List and Description

When the module starts up, the Self-tests are executed automatically and the module enters the operational state if the self-tests pass. In case of failure of the self-test the Module gets unrecoverable error state and OE panic is initiated. The only way to recover the state is the module restart.

The *Security Function Implementations* Table lists all Approved security functions of the module, including specific key size(s) employed for approved services, and implemented modes of operation. The module is in the Approved mode of operation when the module utilizes the services that use the security functions listed in the *Approved Services* Table. The Approved mode of operation is configured in the system by default and can only be transitioned into the non-Approved mode by calling one of the non-Approved services listed in the *Non-Approved Services* Table.

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A6893	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A

Algorithm	CAVP Cert	Properties	Reference
AES-ECB	A6893	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
HMAC-SHA-1	A6893	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A6893	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A6893	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A6893	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A6893	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
SHA-1	A6893	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-224	A6893	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-256	A6893	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A6893	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A6893	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:

Name	Use and Function
ESSIV-CBC-AES	Authenticated encryption/decryption
CMAC	Message authentication
AES-CTR	Symmetric Encryption and Decryption
AES-XTS	Symmetric Encryption and Decryption with ciphertext stealing
AES-CBC-CTS	Symmetric Encryption and Decryption with ciphertext stealing
AES-XCTR	Symmetric Encryption and Decryption with ciphertext stealing

Table 7: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
AES encryption	BC-UnAuth	AES encryption		AES-CBC: (A6893) AES-ECB: (A6893)

Name	Type	Description	Properties	Algorithms
AES decryption	BC-UnAuth	AES decryption		AES-CBC: (A6893) AES-ECB: (A6893)
Hash generation	SHA	Hash generation		SHA-1: (A6893) SHA2-224: (A6893) SHA2-256: (A6893) SHA2-384: (A6893) SHA2-512: (A6893)
HMAC generation	MAC	HMAC generation		HMAC-SHA-1: (A6893) HMAC-SHA2-224: (A6893) HMAC-SHA2-256: (A6893) HMAC-SHA2-384: (A6893) HMAC-SHA2-512: (A6893)

Table 8: Security Function Implementations

2.7 Algorithm Specific Information

N/A for this module.

2.8 RBG and Entropy

N/A for this module.

N/A for this module.

2.9 Key Generation

N/A for this module.

2.10 Key Establishment

N/A for this module.

2.11 Industry Protocols

N/A for this module.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	API input parameters
N/A	Data Output	API output parameters
N/A	Control Input	API function calls, API control input parameters
N/A	Control Output	N/A
N/A	Status Output	API return code, log messages

Table 9: Ports and Interfaces

As a software-only module, the module does not have physical ports. The module supports Crypto Officer (CO) role. 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. The module does not implement a bypass capability. The module does not implement a self-initiated cryptographic output capability. The module does not support Software/Firmware loading. The module provides the following services to the Crypto Officer.

4 Roles, Services, and Authentication

4.1 Authentication Methods

N/A for this module.

No authentication is required at security level 1 and the assumption of the role is implicit by the service being performed.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None

Table 10: Roles

The Crypto Officer role is implicitly assumed by the entity accessing the module services.

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Symmetric encryption	Encrypt a plaintext	get_service_status() returns 1	Key, ciphertext, initialization vector	Ciphertext	AES encryption	Crypto Officer - AES Key: W,E
Symmetric decryption	Decrypt a ciphertext	get_service_status() returns 1	Key, ciphertext, initialization vector	Plain text	AES decryption	Crypto Officer - AES Key: W,E
Message digest generation	Generate message digest	get_service_status() returns 1	Message	Message digest	Hash generation	Crypto Officer
MAC generation	Generate message authentication code	get_service_status() returns 1	Message, key	MAC	HMAC generation	Crypto Officer - HMAC Key: W,E
Show status	Show Module's status	N/A	None	Status information	None	Crypto Officer
Show version	Show Module's versioning information of the module	N/A	None	Module name and version	None	Crypto Officer
Zeroization	Zeroize all SSPs	N/A	None	None	None	Crypto Officer
Self-tests	Perform cryptographic algorithm self-test and integrity test	The successful completion of a service is an implicit indicator for the use of an approved service	N/A	Status information	AES encryption AES decryption Hash	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					generation HMAC generation	

Table 11: Approved Services

G = Generate: The module generates or derives the SSP.
R = Read: The SSP is read from the module (e.g. the SSP is output).
W = Write: The SSP is updated, imported, or written to the module.
E = Execute: The module uses the SSP in performing a cryptographic operation.
Z = Zeroise: The module zeroises the SSP.

4.4 Non-Approved Services

Name	Description	Algorithms	Role
Symmetric encryption	Encrypt a plaintext	ESSIV-CBC-AES AES-CTR	Crypto Officer
Symmetric decryption	Decrypt a ciphertext	ESSIV-CBC-AES AES-CTR	Crypto Officer
MAC generation	Generate message authentication code	CMAC	Crypto Officer
Symmetric encryption with ciphertext stealing	Encrypt ciphertext with ciphertext stealing	AES-XTS AES-CBC-CTS AES-XCTR	Crypto Officer
Symmetric decryption with ciphertext stealing	Decrypt ciphertext with ciphertext stealing	AES-XTS AES-CBC-CTS AES-XCTR	Crypto Officer

Table 12: Non-Approved Services

<Text>

4.5 External Software/Firmware Loaded

The module does not support Software/Firmware loading.

5 Software/Firmware Security

5.1 Integrity Techniques

The module is provided in the form of binary executable code. To ensure the software security, the module is protected by HMAC-SHA2-256 (HMAC Certs. #A6893) algorithm. The software integrity test key (non-SSP) was preloaded to the module's binary in the factory and used for software integrity test only at the pre-operational self-test. At Module's initialization, the integrity of the runtime executable is verified using a HMAC-SHA2-256 which is compared to a MAC value computed at build time. If at load time the MAC does not match the stored, known MAC value, the module would enter to an Error state with all crypto functionality inhibited.

5.2 Initiate on Demand

Integrity test is performed as part of the Pre-Operational Self-Tests. It is automatically executed at power-on. The operator can power-cycle or reboot the tested platform to initiate the software integrity test on-demand.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

How Requirements are Satisfied:

The module operates in a modifiable operational environment per FIPS 140-3 level 1 specifications. The module runs within a commercially available kernel of the general-purpose operating system. The module is executing on the hardware specified in the *Tested Operational Environments* Table.

The operating is restricted to a single operator. Only a single instance of the module is allowed in the Operational environment. The operating environment is non-configurable for the operator. The operational environment provides the capability to separate the module during operation from other functions in the operational environment. Those functions do not obtain information from the module related to the CSPs and do not modify CSPs, PSPs, or the execution flow of the module other than via the interfaces provided by the module itself.

The module does not spawn any processes.

7 Physical Security

The module is comprised of software only and thus does not claim any physical security.

8 Non-Invasive Security

The module does not implement non-invasive attack mitigation techniques to protect the module's unprotected SSPs from non-invasive attacks referenced in Annex F of FIPS 140-3.

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
RAM (Volatile memory)	Temporary storage within TOEPP for SSPs used by the module as part of service execution	Dynamic

Table 13: Storage Areas

The module does not provide persistent storage for keys or SSPs. The module uses pointers to plaintext keys/SSPs that are passed in by the calling application. The module does not store any SSP beyond the lifetime of an API call.

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
SSPs API input	Calling application (TOEPP)	Module	Plaintext	Manual	Electronic	

Table 14: SSP Input-Output Methods

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Zeroization API call	Zeroization command calling context destructor to zeroize all SSPs stored in its context struct	SSPs are actively overwritten with zeroes and thus not recoverable	Using crypto_free_<transformation type>()
Power down	Power down the tested platform to zeroize all SSPs	SSPs stored in the tested platform's volatile memory will be zeroized after power is lost	Power down the tested platform

Table 15: SSP Zeroization Methods

The zeroization mechanism for all of the CSPs is to replace 0s in the memory which originally store the CSPs. Zeroization of the sensitive data within the module and is performed automatically in the frame of release of early allocated crypto handler, by calling function crypto_free_<transformation type>() e.g. crypto_free_skcipher() for symmetric cyphers or crypto_free_shash() for hashes. In addition, powering down the tested platform also has all SSPs zeroized.

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES Key	Keys used for AES encryption / decryption	128, 192, 256 bits - 128 to 256 bits	Symmetric AES key - CSP			AES encryption

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
						AES decryption
HMAC Key	Keyed Hash	at least 112 bits - at least 112 bits	MAC key - CSP			HMAC generation

Table 16: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES Key	SSPs API input	RAM (Volatile memory):Plaintext	For the lifetime of the API call	Zeroization API call Power down	
HMAC Key	SSPs API input	RAM (Volatile memory):Plaintext			

Table 17: SSP Table 2

The table summarizes the keys and Sensitive Security Parameters (SSPs) that are used by the cryptographic services implemented in the module.

9.5 Transitions

- SHA-1

The module includes an implementation of SHA-1 for hashing. This implementation will be non-Approved for all uses starting January 1, 2031. User should move to SHA2, which is available in this module.

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
Software Integrity Test	HMAC-SHA2-256 (A6893)	KAT	SW/FW Integrity	When the test passes, do_integrity_check() returns 0, otherwise it returns -1 and device gets into error state	This test is performed after the CASTs for SHA2-256 and HMAC-SHA2-256

Table 18: Pre-Operational Self-Tests

The module performs Pre-operational Self-tests automatically when the module is loaded into memory (i.e. at power on). The Pre-operational Self-tests contain pre-operational software integrity test to ensure that the module is not corrupted. The integrity test is performed on the runtime image of the module using HMAC-SHA2-256. Prior to software integrity test, a CAST for HMAC-SHA2-256 is performed. If the CAST on the HMAC-SHA-256 is successful, the HMAC

value of the runtime image is recalculated and compared with the stored HMAC value pre-computed at compilation time (for details, see *Integrity Techniques* section). While the module is performing the Pre-operational Self-tests no other functions are available and all output is inhibited. Once Pre-operational Self-tests are completed successfully, the module enters operational mode and cryptographic services are available.

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-ECB Encrypt KAT (A6893)	Key lengths: 128, 192, 256 bits	KAT	CAST	When the test passes, fips140_kat() returns 0, otherwise it returns -1 and device gets into error state	Encrypt	During the module start-up or using fips140_kat()
AES-ECB Decrypt KAT (A6893)	Key lengths: 128, 192, 256 bits	KAT	CAST	When the test passes, fips140_kat() returns 0, otherwise it returns -1 and device gets into error state	Decrypt	During the module start-up or using fips140_kat()
AES-CBC Encrypt KAT (A6893)	Key lengths: 128, 192, 256 bits	KAT	CAST	When the test passes, fips140_kat() returns 0, otherwise it returns -1 and device gets into error state	Encrypt	During the module start-up or using fips140_kat()
AES-CBC Decrypt KAT (A6893)	Key lengths: 128, 192, 256 bits	KAT	CAST	When the test passes, fips140_kat() returns 0, otherwise it returns -1 and device gets into error state	Decrypt	During the module start-up or using fips140_kat()
SHA1 KAT (A6893)	N/A	KAT	CAST	When the test passes, fips140_kat() returns 0, otherwise it returns -1 and device gets into error state	Hash generation	During module start-up or on-demand using fips140_kat()
SHA2-224 KAT (A6893)	N/A	KAT	CAST	When the test passes, fips140_kat() returns 0, otherwise it returns -1 and device gets into error state	Hash generation	During module start-up or on-demand using fips140_kat()
SHA2-256 KAT (A6893)	N/A	KAT	CAST	When the test passes, fips140_kat() returns 0, otherwise it returns -1 and device gets into error state	Hash generation	During module start-up or on-demand using fips140_kat()
SHA2-384 KAT (A6893)	N/A	KAT	CAST	When the test passes, fips140_kat() returns 0, otherwise it returns -1 and device gets into error state	Hash generation	During module start-up or on-demand using fips140_kat()
SHA2-512 KAT (A6893)	N/A	KAT	CAST	When the test passes, fips140_kat() returns 0, otherwise it returns -1 and device gets into error state	Hash generation	During module start-up or on-demand using fips140_kat()
HMAC-SHA1 KAT (A6893)	Key length: 160 bits	KAT	CAST	When the test passes, fips140_kat() returns 0,	HMAC generation	During module start-up or on-

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				otherwise it returns -1 and device gets into error state		demand using fips140_kat()
HMAC-SHA2-224 KAT (A6893)	Key length: 160 bits	KAT	CAST	When the test passes, fips140_kat() returns 0, otherwise it returns -1 and device gets into error state	HMAC generation	During module start-up or on-demand using fips140_kat()
HMAC-SHA2-256 KAT (A6893)	Key length: 160 bits	KAT	CAST	When the test passes, fips140_kat() returns 0, otherwise it returns -1 and device gets into error state	HMAC generation	During module start-up or on-demand using fips140_kat()
HMAC-SHA2-384 KAT (A6893)	Key length: 160 bits	KAT	CAST	When the test passes, fips140_kat() returns 0, otherwise it returns -1 and device gets into error state	HMAC generation	During module start-up or on-demand using fips140_kat()
HMAC-SHA2-512 KAT (A6893)	Key length: 160 bits	KAT	CAST	When the test passes, fips140_kat() returns 0, otherwise it returns -1 and device gets into error state	HMAC generation	During module start-up or on-demand using fips140_kat()

Table 19: Conditional Self-Tests

The module performs Cryptographic Algorithm Self-Tests at module initialization to ensure that the algorithms work as expected, before any security function or process is invoked via module interface, as detailed below.

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
Software Integrity Test	KAT	SW/FW Integrity	On demand	Reload the module or use fips140_kat() API call

Table 20: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-ECB Encrypt KAT (A6893)	KAT	CAST	On demand	Reload the module or use fips140_kat() API call
AES-ECB Decrypt KAT (A6893)	KAT	CAST	On demand	Reload the module or use fips140_kat() API call
AES-CBC Encrypt KAT (A6893)	KAT	CAST	On demand	Reload the module or use fips140_kat() API call
AES-CBC Decrypt KAT (A6893)	KAT	CAST	On demand	Reload the module or use fips140_kat() API call

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
SHA1 KAT (A6893)	KAT	CAST	On demand	Reload the module or use fips140_kat() API call
SHA2-224 KAT (A6893)	KAT	CAST	On demand	Reload the module or use fips140_kat() API call
SHA2-256 KAT (A6893)	KAT	CAST	On demand	Reload the module or use fips140_kat() API call
SHA2-384 KAT (A6893)	KAT	CAST	On demand	Reload the module or use fips140_kat() API call
SHA2-512 KAT (A6893)	KAT	CAST	On demand	Reload the module or use fips140_kat() API call
HMAC-SHA1 KAT (A6893)	KAT	CAST	On demand	Reload the module or use fips140_kat() API call
HMAC-SHA2-224 KAT (A6893)	KAT	CAST	On demand	Reload the module or use fips140_kat() API call
HMAC-SHA2-256 KAT (A6893)	KAT	CAST	On demand	Reload the module or use fips140_kat() API call
HMAC-SHA2-384 KAT (A6893)	KAT	CAST	On demand	Reload the module or use fips140_kat() API call
HMAC-SHA2-512 KAT (A6893)	KAT	CAST	On demand	Reload the module or use fips140_kat() API call

Table 21: Conditional Periodic Information

The module performs on-demand self-tests initiated by the operator (via calling ***fips140_post()*** service), by power-cycling, or rebooting the tested platform. The pre-operational software integrity test and the full suite of self-tests listed in the *Conditional Self-Tests* Table are executed. The same procedure may be employed by the operator to perform periodic self-tests. If any of the tests fail, the module will enter error state.

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Error	The module's only error state	Any failure in context of the execution of the implemented self-tests during module start-up or the self-test service	Reload the module	The OE`s OS log contains message 'FIPS : POST - one or more self-tests failed'.

Table 22: Error States

The module has a variable (***skc_fips_enabled***) indicating the status of the Self-test. It contains ***0*** if the Self-test was failed and it contains ***1*** if the Self-test was successful.

The kernel logs contains message:

`FIPS : POST - integrity test failed`

in case of integrity test is failed, or

`FIPS : <alg name>, test failed, err=<test number>`

in case of CAST is failed.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

This cryptographic module is built-in along with the Linux Kernel. The module is initialized during the Kernel boot-up before any cryptographic functionality is available. The Kernel is responsible for the initialization and loading processes of the module. The module is designed with module init entry point which ensures that the Pre-operational self-tests and CASTs are initiated automatically when the module is loaded.

11.2 Administrator Guidance

N/A

11.3 Non-Administrator Guidance

N/A

12 Mitigation of Other Attacks

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