SAMSUNG

Samsung Electronics Co., Ltd.

Samsung CryptoCore Cryptographic Module

FIPS 140-3 Non-Proprietary Security Policy

Document Version 0.1.17
Last Update: 17-07-2024

Table of Contents

List of Tables

List of Figures

1 General

1.1 Overview

This document is the non-proprietary FIPS 140-3 Security Policy for the Samsung CryptoCore 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 a 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 module provides cryptographic services to applications through an application program interface (API). The module also interacts with the operating system via system calls.

Module Type: Software

Module Embodiment: MultiChipStand

Module Characteristics:

Cryptographic Boundary:

The cryptographic boundary of the module is a single object file named cryptocore.0.2.9.FIPS.1.o, which is statically linked into the position-independent code

libcryptocore.so.0.1.0 shared library. There are no excluded components inside the cryptographic boundary. The module is intended only for single-threaded execution per process.

Tested Operational Environment's Physical Perimeter (TOEPP):

The TOEPP of the module is a Samsung Smart TV Q70B with Tizen 7.0.
The module is located in the volatile memory controlled by OS Tizen 7.0 installed on hardware platform Samsung Smart TV Q70B.

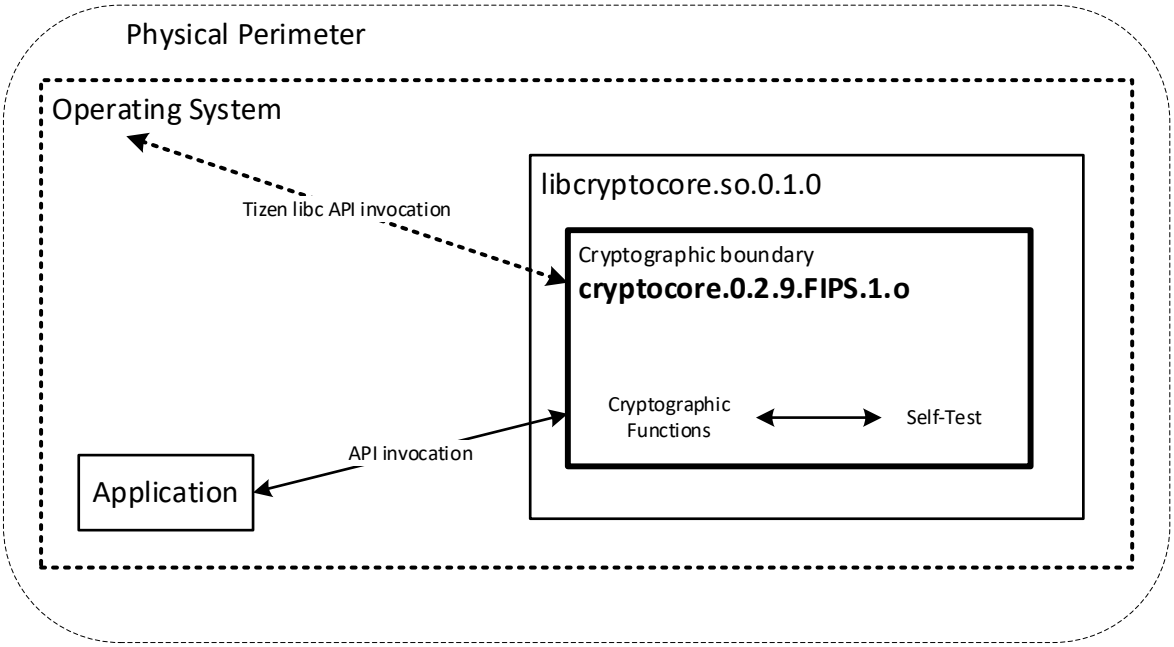

Figure 1: Block Diagram

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
cryptocore.0.2.9.FIPS.1.o	0.2.9.FIPS.1		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)
Tizen 7.0	Samsung Smart TV Q70B	Pontus-M	No		0.2.9.FIPS.1

Table 3: 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 excluded from the security requirements.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved Mode	The module's only mode of operation in which all approved services are available	Approved	

Table 4: Modes List and Description

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A4968	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB128	A4968	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A4968	Direction - Generation, Verification Key Length - 128	SP 800-38B
AES-CTR	A4968	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A4968	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-OFB	A4968	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A

Algorithm	CAVP Cert	Properties	Reference
ECDSA KeyGen (FIPS186-4)	A4968	Curve - P-224, P-256, P-384, P-521	FIPS 186-4
ECDSA KeyVer (FIPS186-4)	A4968	Curve - P-192, P-224, P-256, P-384, P-521	FIPS 186-4
ECDSA SigGen (FIPS186-4)	A4968	Curve - P-224, P-256, P-384, P-521	FIPS 186-4
ECDSA SigVer (FIPS186-4)	A4968	Curve - P-192, P-224, P-256, P-384, P-521	FIPS 186-4
HMAC DRBG	A4968	Prediction Resistance - No Mode - SHA2-256, SHA2-512	SP 800-90A Rev. 1
HMAC-SHA2-224	A4968	Key Length - Key Length: 112, 504, 512, 520, 2048	FIPS 198-1
HMAC-SHA2-256	A4968	Key Length - Key Length: 112, 504, 512, 520, 2048	FIPS 198-1
HMAC-SHA2-384	A4968	Key Length - Key Length: 112, 504, 1024, 1032, 2048	FIPS 198-1
HMAC-SHA2-512	A4968	Key Length - Key Length: 112, 504, 1024, 1032, 2048	FIPS 198-1
KAS-ECC CDH-Component SP800-56Ar3 (CVL)	A4968	-	SP 800-56A Rev. 3
RSA SigGen (FIPS186-4)	A4968	Signature Type - PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096	FIPS 186-4
RSA SigVer (FIPS186-4)	A4968	Signature Type - PKCS 1.5, PKCSPSS Modulo - 1024, 2048, 3072, 4096	FIPS 186-4
SHA2-224	A4968	-	FIPS 180-4
SHA2-256	A4968	-	FIPS 180-4
SHA2-384	A4968	-	FIPS 180-4
SHA2-512	A4968	-	FIPS 180-4
RSA KeyGen (FIPS186-4)	A4968	Key Generation Mode - B.3.3 Modulo - 2048, 3072, 4096 Primality Tests - Table C.2 Private Key Format - Standard	FIPS 186-4

Table 5: Approved Algorithms

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
CKG: RSA	Type:Asymmetric RSA Modulo:2048, 3072, 4096	Samsung CryptoCore Cryptographic Module	Section B.3.3 of FIPS 186-4 and Sections 4 / 5.1 of SP 800-133r2 (V is all zeroes)
CKG: ECC	Type:Asymmetric ECC / ECDSA	Samsung CryptoCore Cryptographic Module	Section B.4.2 of FIPS 186-4 respectively Section 5.6.1.2.2 of SP

Name	Properties	Implementation	Reference
	Curve:P-224, P-256, P-384, P-521		800-56Ar3 and Sections 4, 5.1, and 5.2 of SP 800-133r2 (V is all zeroes)

Table 6: Vendor-Affirmed Algorithms

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 encryption	BC-UnAuth	AES encryption		AES-CBC AES-CFB128 AES-CTR AES-ECB AES-OFB
AES decryption	BC-UnAuth	AES decryption		AES-CBC AES-CFB128 AES-CTR AES-ECB AES-OFB
HMAC generation	MAC	HMAC generation	Truncation:Not supported	HMAC-SHA2-224 SHA2-224 HMAC-SHA2-256 SHA2-256 HMAC-SHA2-384 SHA2-384 HMAC-SHA2-512 SHA2-512
CMAC generation	MAC	CMAC generation	Truncation:Not supported	AES-CMAC AES-ECB
Hash generation	SHA	Hash generation		SHA2-224 SHA2-256 SHA2-384 SHA2-512

Name	Type	Description	Properties	Algorithms
RSA key pair generation	AsymKeyPair-KeyGen	RSA key pair generation		RSA KeyGen (FIPS186-4) CKG: RSA
RSA signature generation	DigSig-SigGen	RSA signature generation		RSA SigGen (FIPS186-4) SHA2-224 SHA2-256 SHA2-384 SHA2-512
RSA signature verification	DigSig-SigVer	RSA signature verification		RSA SigVer (FIPS186-4) SHA2-224 SHA2-256 SHA2-384 SHA2-512
ECC key generation	AsymKeyPair-KeyGen	ECDSA and ECC key generation		ECDSA KeyGen (FIPS186-4) CKG: RSA
ECDSA signature generation	DigSig-SigGen	ECDSA signature generation		ECDSA SigGen (FIPS186-4) SHA2-224 SHA2-256 SHA2-384 SHA2-512
ECDSA signature verification	DigSig-SigVer	ECDSA signature verification		ECDSA SigVer (FIPS186-4) SHA2-224 SHA2-256 SHA2-384 SHA2-512
ECDSA public key validation	AsymKeyPair-PubKeyVal	ECDSA public key validation		ECDSA KeyVer (FIPS186-4)
DRBG	DRBG	Random number generation		HMAC DRBG HMAC-SHA2-256 SHA2-256 HMAC-SHA2-512 SHA2-512
Shared secret computation	KAS-SSC	ECC CDH primitive	Encryption strength:Between 112 and 256 bits	KAS-ECC CDH-Component SP800-56Ar3

Table 7: Security Function Implementations

2.7 Algorithm Specific Information

AES CTR: the externally loaded counters of AES CTR shall have the property defined in [SP800-38A], 6.5 The Counter Mode.

2.8 RBG and Entropy

N/A for this module.

N/A for this module.

The module passively receives the entropy for seeding/reseeding DRBG while exercising no control over the amount or the quality of the obtained entropy. The random string for seeding/reseeding must supply at least 112 bits of entropy to provide the minimum acceptable security strength that is 112 bits according to [SP800-57pt1r5].

The module employs a SP800-90Ar1 HMAC_DRBG as Random Number Generation service.
For DRBG Instantiation to create a seed the module requires a random string of 48 bytes (384 bits) size: 32 bytes (256 bits) for "entropy input" and 16 bytes (128 bits) for "nonce".
For periodic DRBG Reseeding to create a reseed the module requires a random string of 64 bytes (512 bits) size: 32 bytes (256 bits) for "entropy input" and 32 bytes (256 bits) for "additional input".
For explicit DRBG Reseeding to create a reseed the module requires a random string of 32 bytes (256 bits) size: 32 bytes (256 bits) for "entropy input".

The output of the module implemented DRBG is used to generate random bits for

- asymmetric key pair generation that compliant with FIPS 186-4 and SP800-90Ar1,
- signature generation that compliant with FIPS 186-4,
- generation of random number.

The calling application is responsible for storage of generated keys returned by the module. It is not possible for the module to output information during the key generating process.

2.9 Key Generation

For generating RSA, ECC key pairs, the module implements Asymmetric Key Generation services compliant with FIPS 186-4. The random value used in asymmetric key generation is obtained using Random Number Generation service of the module. In accordance with FIPS 140-3 IG D.H, the module performs Cryptographic Key Generation (CKG) for asymmetric keys as per sections 5.1, 5.2 SP800-133r2 (vendor affirmed) by obtaining a random bit string as per section 4 SP800-133r2 directly from a DRBG without any V, as described in Additional Comments 2 in FIPS 140-3 IG D.H.

2.10 Key Establishment

N/A

2.11 Industry Protocols

N/A

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
N/A	Status Output	API return codes, log messages

Table 8: Ports and Interfaces

As a software-only module, the module does not have physical ports. For the purpose of the FIPS 140-3 validation, the physical ports are interpreted to be the physical ports of the hardware platform on which it runs.
The logical interfaces are the application program interface (API) through which applications request services.
The control output interface is omitted on purpose because the module does not implement it.

4 Roles, Services, and Authentication

4.1 Authentication Methods

N/A for this module.

The module does not support user authentication.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None

Table 9: 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	The successful completion of a service is an implicit indicator for the use of an approved service	Key, plain text, mode, padding method, initialization vector	Cipher text	AES encryption	Crypto Officer - AES keys: R,E
Symmetric decryption	Decrypt a cipher text	The successful completion of a service is an implicit indicator for the use of an approved service	Key, cipher text, mode, padding, initialization vector	Plain text	AES decryption	Crypto Officer - AES keys: R,E
Asymmetric key generation	Generate asymmetric RSA and ECC key pair	The successful completion of a service is an implicit indicator for the use of an approved service	RSA: padding method, modulus length, optionally fixed private key, entropy input string; ECDSA: curve, entropy input string	Key pair	RSA key pair generation ECC key generation	Crypto Officer - RSA private key: G,R,W - RSA public key: G,W - ECDSA private key: G,W - ECDSA public key: G,W - ECC CDH private key: G,W - ECC CDH public key: G,W - Intermediate key generation

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						values: G,E,Z - Entropy input string: R - DRBG V: G,E - DRBG Key: G,E - Entropy buffer: E
Digital signature generation	Generate digital signature	The successful completion of a service is an implicit indicator for the use of an approved service	RSA: private key, message, modulus, padding method; ECDSA: public key, message, curve	Signature	RSA signature generation ECDSA signature generation	Crypto Officer - RSA private key: R,E - ECDSA private key: R,E
Digital signature verification	Verify digital signature	The successful completion of a service is an implicit indicator for the use of an approved service	RSA: public key, message, signature, modulus, padding method; ECDSA: public key, message, signature, curve	Verification result	RSA signature generation ECDSA signature verification ECDSA public key validation	Crypto Officer - RSA public key: R,E - ECDSA public key: R,E
Message digest generation	Generate message digest	The successful completion of a service is an implicit indicator for the use of an approved service	Message, algorithm	Message digest	Hash generation	Crypto Officer
MAC generation	Generate message	The successful completion of a service is an	Message, algorithm, key	MAC	HMAC generation	Crypto Officer - CMAC keys: R,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	authentication code	implicit indicator for the use of an approved service			CMAC generation	- HMAC keys: R,E
Random number generation	Generate random number	The successful completion of a service is an implicit indicator for the use of an approved service	Entropy input string, Personalization string, additional input	Random bits	DRBG	Crypto Officer - Entropy input string: R - DRBG V: G,E - DRBG Key: G,E - Entropy buffer: E
Shared secret computation	Compute shared secret	The successful completion of a service is an implicit indicator for the use of an approved service	Received public key, private key	Shared secret	Shared secret computation	Crypto Officer - ECC CDH received public key: R,E - ECC CDH private key: R,E - Shared secret: G,W
Show status	Show status of the module	N/A	None	Status information	None	Crypto Officer
Show module's versioning information	Show the versioning information of the module	N/A	None	Module name and version	None	Crypto Officer
Zeroization	Zeroize SSP	N/A	None	None	None	Crypto Officer - AES keys: Z - CMAC keys: Z - HMAC keys:

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Z - RSA private key: Z - RSA public key: Z - ECDSA private key: Z - ECDSA public key: Z - ECC CDH private key: Z - ECC CDH public key: Z - Entropy buffer: Z - DRBG V: Z - DRBG Key: Z
Cryptographic algorithm self-test and integrity test	Initiate cryptographic algorithm self-test and integrity test	The successful completion of a service is an implicit indicator for the use of an approved service	Entropy input string	Status information	AES decryption HMAC generation CMAC generation Hash generation RSA signature generation RSA signature verification ECDSA	Crypto Officer - Entropy input string: R - DRBG V: G,E - DRBG Key: G,E - Entropy buffer: E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					signature generation ECDSA signature verification ECDSA public key validation DRBG Shared secret computation	
Module start-up	Run cryptographic algorithm self-test and integrity test at the module start-up	The successful completion of a service is an implicit indicator for the use of an approved service	Entropy input string	Status information	AES decryption HMAC generation CMAC generation Hash generation RSA signature generation RSA signature verification ECDSA signature generation ECDSA signature verification ECDSA	Crypto Officer - Entropy input string: R - DRBG V: G,E - DRBG Key: G,E - Entropy buffer: E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					public key validation DRBG Shared secret computation	

Table 10: Approved Services

The approved security service indicator of the module is compliant to the example scenario 2) of [IG] 2.4.C.

4.4 Non-Approved Services

N/A for this module.

4.5 External Software/Firmware Loaded

The module does not support Software/Firmware loading.

5 Software/Firmware Security

5.1 Integrity Techniques

The integrity of the module is verified by comparing MAC calculated on the module at runtime and the value stored in the module, which was calculated at build-time.
An approved integrity technique used in the integrity test is HMAC-SHA256 algorithm implemented in the module itself. The integrity test uses a 256-bit key, which resides within the module code and is not considered a SSP. Before executing the integrity test, the module performs CAST of SHA-256 and HMAC-SHA512 algorithms.

5.2 Initiate on Demand

The integrity test can be initiated on demand by the operator by calling fips_post() API.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

How Requirements are Satisfied:

The module runs on a commercially available general-purpose operating system. The operating system is restricted to a single operator (concurrent operators are explicitly excluded). The operational environment is non-configurable for operator, thus the module operates securely by default. The application that requests cryptographic services is the single user of the module, even when the application is serving multiple clients. The operating system 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
Volatile memory	The OE's volatile memory (RAM) shared with the linked application, but under the module's control	Dynamic

Table 11: Storage Areas

The module does not provide persistent storage for keys or SSPs. The module stores SP800-90Ar1 DRBG state values and the Entropy buffer for at most the runtime of the module. The module uses pointers to plaintext keys/SSPs that are passed in by the calling application. The module does not store SSP beyond the lifetime of an API call or beyond the runtime of the module. Allocated memory in RAM for SSP is managed by the module.

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
API input	Linked application in the TOEPP	Volatile memory	Plaintext	Manual	Electronic	
API output	Volatile memory	Linked application in the TOEPP	Plaintext	Manual	Electronic	

Table 12: SSP Input-Output Methods

SSPs enter the module's cryptographic boundary as cryptographic algorithm API parameters in plaintext. They are associated with memory locations and do not persist across power cycles. The module does not output intermediate key generation values. The module provides the resulting keys as output parameters of key generation service API to the calling application, but they do not cross the physical perimeter. Import and export operations of SSP are plaintext manual electronic entry for the module and user application inside OE physical perimeter (TOEPP) in terms of IG 140-3 9.5.A.

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Destructor	CryptoCoreContainer destructor zeroises all SSPs stored in its context struct	SSPs are actively overwritten with zeroes and thus not recoverable	Using destroy_CryptoCoreContainer()
Intermediate	Intermediate and temporary SSPs are automatically zeroized by the module before a function returns	SSPs are actively overwritten with zeroes and thus not recoverable	No, done automatically by the module
Entropy buffer zeroization	The module's entropy buffer SSP storing the user-provided entropy is zeroized	SSPs are actively overwritten with zeroes and thus not recoverable	Using fips_cleanup_entropy_buffer() or by unloading the module

Table 13: SSP Zeroization Methods

Zeroisation of sensitive data is performed by calling destruction API function destroy_CryptoCoreContainer() by the operator. This functions overwrites the memory occupied by SSPs with “zeros” and deallocates the memory. The application that uses the module is responsible for calling the destruction function destroy_CryptoCoreContainer(). The calling application is responsible for parameters passed in and out of the module. The return of the destroy_CryptoCoreContainer() function indicates the successful completion of the zeroisation procedure.
Zeroisation of entropy buffer is performed by calling the fips_cleanup_entropy_buffer() API function. The API function is called either by the operator or automatically through the destructor of the module at the unloading stage. The return of the fips_cleanup_entropy_buffer() function or successful completion of the module indicates the successful completion of the zeroisation procedure, accordingly.

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES keys	Keys used for AES encryption / decryption	128, 192, 256 bits - 128 to 256 bits	Symmetric AES key - CSP			AES encryption AES decryption
CMAC keys	Keys used for CMAC generation	128 bits - 128 bits	Symmetric AES key - CSP			CMAC generation
HMAC keys	Keys used for HMAC computation	112 bits or longer - 112 bits or greater	Symmetric HMAC key - CSP			HMAC generation
RSA private key	RSA private key for digital signature computations	Up to 4096 bits - 112 bits or greater	Asymmetric RSA private key - CSP	RSA key pair generation		RSA signature generation RSA signature verification
RSA public key	RSA public key for digital signature computations	Up to 4096 bits - Less than 112 bits for module size 1024 bits, greater than 112 bits for modulus sizes of 2048 bits or longer	Asymmetric RSA public key - PSP	RSA key pair generation		RSA signature generation RSA signature verification
ECDSA private key	ECDSA private key for digital signature computations	Up to 521 bits - 112 bits or greater	Asymmetric ECDSA private key - CSP	ECC key generation		ECDSA signature generation ECDSA signature verification
ECDSA public key	ECDSA public key for digital signature computations	Up to 521 bits - 112 bits or greater	Asymmetric ECDSA public key - PSP	ECC key generation		ECDSA signature generation ECDSA

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
						signature verification ECDSA public key validation
ECC CDH received public key	ECC public key from other party for shared secret computation	Up to 521 bits - 112 to 256 bits	Asymmetric ECC public key - PSP			Shared secret computation
ECC CDH private key	Own ECC private key for shared secret computation	Up to 521 bits - 112 bits or greater	Asymmetric ECC private key - CSP	ECC key generation		Shared secret computation
ECC CDH public key	Own ECC public key for shared secret computation	Up to 521 bits - 112 bits or greater	Asymmetric ECC public key - PSP	ECC key generation		Shared secret computation
Shared secret	Result of the shared secret computation	Up to 521 bits - 112 bits or greater	Shared secret - CSP		Shared secret computation	
Entropy input string	User-provided entropy input	Up to 2048 bits - 112 bits or greater	Entropy input - CSP			
Entropy buffer	Module-internal entropy buffer and its inputs	Up to 2048 bits - 112 bits or greater	Entropy buffer - CSP			DRBG
DRBG V	Value V of the HMAC DRBG's state	256 or 512 bits - 112 bits or greater	Value V of HMAC DRBG - CSP	DRBG		DRBG
DRBG Key	Key of the HMAC DRBG	256 or 512 bits - 112 bits or greater	Key of HMAC DRBG - CSP	DRBG		DRBG
Intermediate key generation values	Intermediate RSA and ECC key generation values	Up to 4096 bits - 112 bits or greater	Intermediate key generation values - CSP	RSA key pair generation ECC key generation		

Table 14: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES keys	API input	Volatile memory:Plaintext	For the lifetime of the API call	Destructor	
CMAC keys	API input	Volatile memory:Plaintext	For the lifetime of the API call	Destructor	
HMAC keys	API input	Volatile memory:Plaintext	For the lifetime of the API call	Destructor	
RSA private key	API input API output	Volatile memory:Plaintext	For the lifetime of the API call	Destructor	RSA public key:Paired With
RSA public key	API input API output	Volatile memory:Plaintext	For the lifetime of the API call	Destructor	RSA private key:Paired With
ECDSA private key	API input API output	Volatile memory:Plaintext	For the lifetime of the API call	Destructor	ECDSA public key:Paired With
ECDSA public key	API input API output	Volatile memory:Plaintext	For the lifetime of the API call	Destructor	ECDSA private key:Paired With
ECC CDH received public key	API input	Volatile memory:Plaintext	For the lifetime of the API call	Intermediate	ECC CDH private key:Used With Shared secret:Used To Derive
ECC CDH private key	API input API output	Volatile memory:Plaintext	For the lifetime of the API call	Destructor	ECC CDH received public key:Used With ECC CDH public key:Paired With Shared secret:Used To Derive

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
ECC CDH public key	API input API output	Volatile memory:Plaintext	For the lifetime of the API call	Destructor	ECC CDH private key:Paired With
Shared secret	API output	Volatile memory:Plaintext	For the lifetime of the API call	Intermediate	ECC CDH received public key:Derived From ECC CDH private key:Derived From
Entropy input string	API input	Volatile memory:Plaintext	For the lifetime of the API call	Intermediate	Entropy buffer:Stored In
Entropy buffer		Volatile memory:Plaintext	For the runtime of the module or until zeroization	Entropy buffer zeroization	Entropy input string:Used To Store
DRBG V		Volatile memory:Plaintext	For the runtime of the module or until zeroization	Destructor Intermediate	Entropy buffer:Seeded From DRBG Key:Paired With
DRBG Key		Volatile memory:Plaintext	For the runtime of the module or until zeroization	Destructor Intermediate	Entropy buffer:Seeded From DRBG V:Paired With
Intermediate key generation values		Volatile memory:Plaintext	For the lifetime of the API call	Intermediate	RSA private key:Used to Generate RSA public key:Used to Generate ECDSA private key:Used to Generate ECDSA public key:Used to Generate ECC CDH private key:Used to Generate ECC CDH public key:Used to Generate

Table 15: SSP Table 2

9.5 Transitions

Transition from FIPS 186-4 to FIPS 186-5 and SP 800-186 as specified in FIPS 140-3 IG C.K.

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A4968)	Key length: 256 bits	See (*) below the table.	SW/FW Integrity	When the test passes, get_fips_status() returns FIPS_STATUS_SUCCESS, otherwise it returns FIPS_STATUS_FAIL and get_fips_status_reason() returns FIPS_STATUS_INTEGRITY_FAIL	This test is performed after the CASTs for SHA2-256 and HMAC-SHA2-512

Table 16: Pre-Operational Self-Tests

(*) A MAC is calculated over the module FIPS-relevant APIs at runtime and compared to the value stored in the module, which was calculated at build-time. This MAC is calculated between designated memory addresses of the .text and .rodata sections of the object file, excluding relocatable address parts. These sections have read-only and executable attributes in terms of the ELF file. The content of these sections is formed at the module build stage.

While the module is performing the Pre-Operational Self-Test, no other functions are available and all output is inhibited. Once Pre-Operational Self-Test is completed successfully, the module enters the Approved Mode of Operation and cryptographic services are available.

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-ECB (A4968)	Key length: 256 bits	KAT	CAST	When the test passes, get_fips_status_reason() returns	AES decryption	During module start-up or on-

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				FIPS_STATUS_SUCCESS, otherwise it returns FIPS_STATUS_SELFTEST_FAIL		demand using fips_post()
AES-CMAC (A4968)	Key length: 128 bits	KAT	CAST	When the test passes, get_fips_status_reason() returns FIPS_STATUS_SUCCESS, otherwise it returns FIPS_STATUS_SELFTEST_FAIL	CMAC generation	During module start-up or on-demand using fips_post()
SHA2-224 (A4968)	N/A	KAT	CAST	When the test passes, get_fips_status_reason() returns FIPS_STATUS_SUCCESS, otherwise it returns FIPS_STATUS_SELFTEST_FAIL	Hash generation	During module start-up or on-demand using fips_post()
SHA2-256 (A4968)	N/A	KAT	CAST	When the test passes, get_fips_status_reason() returns FIPS_STATUS_SUCCESS, otherwise it returns FIPS_STATUS_SELFTEST_FAIL	Hash generation	During module start-up or on-demand using fips_post()
SHA2-384 (A4968)	N/A	KAT	CAST	When the test passes, get_fips_status_reason() returns FIPS_STATUS_SUCCESS, otherwise it returns FIPS_STATUS_SELFTEST_FAIL	Hash generation	During module start-up or on-demand using fips_post()
SHA2-512 (A4968)	N/A	KAT	CAST	When the test passes, get_fips_status_reason() returns FIPS_STATUS_SUCCESS, otherwise it returns FIPS_STATUS_SELFTEST_FAIL	Hash generation	During module start-up or on-demand using fips_post()

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA2-512 (A4968)	Key length: 1024 bits	KAT	CAST	When the test passes, get_fips_status_reason() returns FIPS_STATUS_SUCCESS, otherwise it returns FIPS_STATUS_SELFTEST_FAIL	HMAC generation	During module start-up or on-demand using fips_post()
RSA SigGen (FIPS186-4) (A4968)	Key length: 2048 bits, Padding: PKCS-v1.5	KAT	CAST	When the test passes, get_fips_status_reason() returns FIPS_STATUS_SUCCESS, otherwise it returns FIPS_STATUS_SELFTEST_FAIL	RSA digital signature generation	During module start-up or on-demand using fips_post()
RSA SigVer (FIPS186-4) (A4968)	Key length: 2048 bits, Padding: PKCS-v1.5	KAT	CAST	When the test passes, get_fips_status_reason() returns FIPS_STATUS_SUCCESS, otherwise it returns FIPS_STATUS_SELFTEST_FAIL	RSA digital signature verification	During module start-up or on-demand using fips_post()
ECDSA SigGen (FIPS186-4) (A4968)	Curve: P-224, Hash function: SHA2-512	KAT	CAST	When the test passes, get_fips_status_reason() returns FIPS_STATUS_SUCCESS, otherwise it returns FIPS_STATUS_SELFTEST_FAIL	ECDSA digital signature generation	During module start-up or on-demand using fips_post()
ECDSA SigVer (FIPS186-4) (A4968)	Curve: P-224, Hash function: SHA2-512	KAT	CAST	When the test passes, get_fips_status_reason() returns FIPS_STATUS_SUCCESS, otherwise it returns FIPS_STATUS_SELFTEST_FAIL	ECDSA digital signature verification	During module start-up or on-demand using fips_post()
KAS-ECC CDH-Component	Curve: P-224	KAT	CAST	When the test passes, get_fips_status_reason() returns	Shared secret computation	During module start-up or on-

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
SP800-56Ar3 (A4968)				FIPS_STATUS_SUCCESS, otherwise it returns FIPS_STATUS_SELFTEST_FAIL		demand using fips_post()
HMAC DRBG (A4968)	MAC: HMAC-SHA2-256	KAT	CAST	When the test passes, get_fips_status_reason() returns FIPS_STATUS_SUCCESS, otherwise it returns FIPS_STATUS_SELFTEST_FAIL	KAT of instantiation, reseeding, generate, and generate calls in one sweep according to SP 800-90Ar1, Sec. 11.3 and 7. of IG 10.3.A	During module start-up or on-demand using fips_post()
HMAC DRBG (A4968)	MAC: HMAC-SHA2-256 or HMAC-SHA2-512	KAT	CAST	When the test passes, get_fips_status_reason() returns FIPS_STATUS_SUCCESS, otherwise it returns FIPS_STATUS_DRBG_HEALTH_FAIL	Same as test above, but the tested MAC variant is chosen based on the one used by the current HMAC instance. The instance state is untouched	Every 400 generate calls of an DRBG instance
RSA (FIPS186-4) (A4968)	Padding: none	PCT	PCT	When the test passes, get_fips_status_reason() returns FIPS_STATUS_SUCCESS, otherwise it returns FIPS_STATUS_RSA_PCT_FAIL	RSA signature generation and verification of a fixed 32-byte message	After each RSA key pair generation and key pair import
ECDSA (FIPS186-4) (A4968)	N/A	PCT	PCT	When the test passes, get_fips_status_reason() returns FIPS_STATUS_SUCCESS, otherwise it returns FIPS_STATUS_ECDSA_PCT_FAIL	ECDSA signature generation and verification of a fixed 32-byte message	After each ECDSA key pair generation

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
						and key pair import

Table 17: Conditional Self-Tests

None of the keys used for the KAT are considered as SSP.

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A4968)	See (*) below the table.	SW/FW Integrity	On Demand	Reloading the module or using fips_post()

Table 18: Pre-Operational Periodic Information

(*) A MAC is calculated over the module FIPS-relevant APIs at runtime and compared to the value stored in the module, which was calculated at build-time. This MAC is calculated between designated memory addresses of the .text and .rodata sections of the object file, excluding relocatable address parts. These sections have read-only and executable attributes in terms of the ELF file. The content of these sections is formed at the module build stage.

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-ECB (A4968)	KAT	CAST	On demand	Reloading the module or using fips_post()
AES-CMAC (A4968)	KAT	CAST	On demand	Reloading the module or using fips_post()
SHA2-224 (A4968)	KAT	CAST	On demand	Reloading the module or using fips_post()
SHA2-256 (A4968)	KAT	CAST	On demand	Reloading the module or using the fips_post() API
SHA2-384 (A4968)	KAT	CAST	On demand	Reloading the module or using fips_post()
SHA2-512 (A4968)	KAT	CAST	On demand	Reloading the module or using fips_post()
HMAC-SHA2-512 (A4968)	KAT	CAST	On demand	Reloading the module or using fips_post()
RSA SigGen (FIPS186-4) (A4968)	KAT	CAST	On demand	Reloading the module or using fips_post()
RSA SigVer (FIPS186-4) (A4968)	KAT	CAST	On demand	Reloading the module or using fips_post()
ECDSA SigGen (FIPS186-4) (A4968)	KAT	CAST	On demand	Reloading the module or using fips_post()

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
ECDSA SigVer (FIPS186-4) (A4968)	KAT	CAST	On demand	Reloading the module or using fips_post()
KAS-ECC CDH-Component SP800-56Ar3 (A4968)	KAT	CAST	On demand	Reloading the module or using fips_post()
HMAC DRBG (A4968)	KAT	CAST	On demand	Reloading the module or using fips_post()
HMAC DRBG (A4968)	KAT	CAST	On demand	Repeated calls of the DRBG generate function
RSA (FIPS186-4) (A4968)	PCT	PCT	On demand	After RSA key pair generation and key pair import
ECDSA (FIPS186-4) (A4968)	PCT	PCT	On demand	After ECDSA key pair generation and key pair import

Table 19: Conditional Periodic Information

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	Reloading the module or using fips_post()	get_fips_status() returns FIPS_STATUS_FAIL

Table 20: Error States

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

The module is built into the operational environment and delivered with a device. There is no standalone delivery of the module as a software library.
The module is initialized during the loading of the module before any cryptographic functionality is available. The Tizen operating system is responsible for the initialization and loading processes of the module. The module is designed with constructor (default entry point of the

module) which ensures that CAST and POST are initiated automatically when the module is loaded.

11.2 Administrator Guidance

The guidance is provided in the document “Samsung CryptoCore Cryptographic Module, Software Version: 0.2.9.FIPS.1, Functional Design, Document Version 0.1.23, Last Update: 12-07-2024”.

11.3 Non-Administrator Guidance

The guidance is provided in the document “Samsung CryptoCore Cryptographic Module, Software Version: 0.2.9.FIPS.1, Functional Design, Document Version 0.1.23, Last Update: 12-07-2024”.

11.4 Design and Rules

The usual sequence of secure operations:

- create crypto-container:

```
CryptoCoreContainer *sha = create_CryptoCoreContainer(ID_SHA256);
```

- perform the operation:

```
sha->MD_getHASH(sha, message, message_len_bytes, digest);
```

- destroy crypto container:

```
destroy_CryptoCoreContainer(sha);
```

12 Mitigation of Other Attacks

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