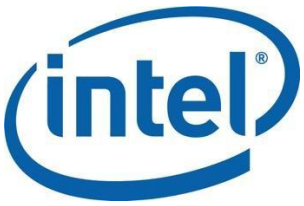

Intel Corporation

Intel FNIC Control Plane Cryptographic Module

FIPS 140-3 Non-Proprietary Security Policy

Table of Contents

List of Tables

List of Figures

1 General

1.1 Overview

The Intel FNIC Control Plane Cryptographic Module provides cryptographic services to the Intel E610 and E830 SOCs. E610 is a two port, low bandwidth Ethernet controller supporting speeds of 1Gb/s and 10Gb/s on each port. E830 is a 200G/s Ethernet controller. The cryptographic module is a sub-chip cryptographic subsystem hardware 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	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:

As specified in Section 1.1. above, the Intel FNIC Control Plane Cryptographic Module provides cryptographic services to the Intel E610 and E830 SOCs.

Module Type: Hardware

Module Embodiment: Single Chip

Module Characteristics : SubChip

Cryptographic Boundary:

The physical perimeter of the module is the Intel SoC (E610 and E830). The cryptographic boundary of the module comprises of the sub-chip cryptographic subsystem (hardware versions 5.0.11 and 5.0.19 respectively) within each SoC.

Tested Operational Environment's Physical Perimeter (TOEPP):

The TOEPP of the module is the Intel E610 or E830 SOC.

Figure 1: Intel SoC E610

Figure 2: Intel SoC E830

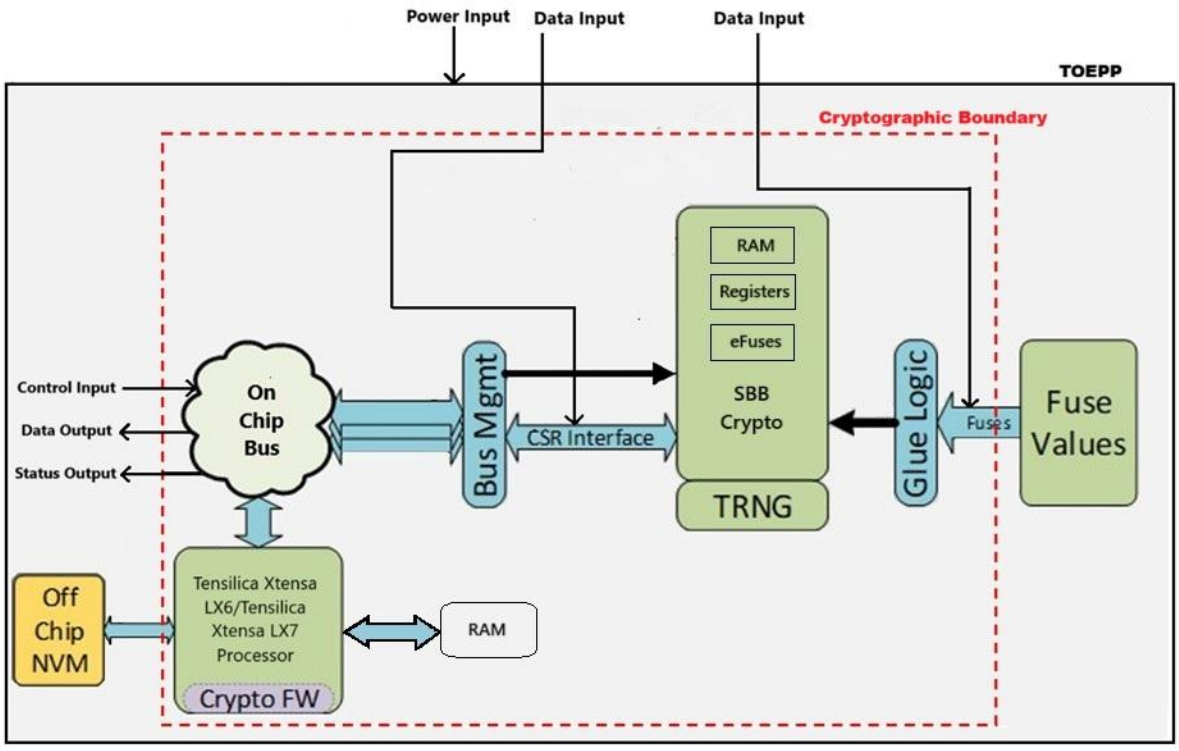

Figure 3: Block Diagram

2.2 Tested and Vendor Affirmed Module Version and Identification

Tested Module Identification – Hardware:

Model and/or Part Number	Hardware Version	Firmware Version	Processors	Features
Intel E610 SoC	5.0.19	1.00.01	Tensilica Xtensa LX6	
Intel E830 SoC	5.0.11	1.00.01	Tensilica Xtensa LX7	

Table 2: Tested Module Identification – Hardware

Please Note: The module identifier "SBB Hybrid Cryptographic Module" correlates to the module name i.e., Intel FNIC Control Plane Cryptographic Module.

2.3 Excluded Components

There are no components excluded within the module boundary.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved Mode	The module supports an Approved mode of operation by default. No configuration of the module or installation steps are required from the operator. When the module is powered on, its pre-operational self-tests are executed without any operator intervention. The module's cryptographic functions will only be available after all self-tests at boot (pre-operational and cryptographic algorithm self-tests) have passed successfully. If any of the self-tests fail, the module will transition to the Hard Error state.	Approved	fips_approved flag set to True
Non-Approved Mode	The module supports a Non-Approved mode of operation, depending on the service. No separate configuration of the module or installation steps are required to be done. If a service using non-approved algorithm is called, the module executes the service and returns status indicator different from the one returned in the Approved Mode. and enters the non-Approved mode of operation.	Non-Approved	None (fips_approved flag not set)

Table 3: Modes List and Description

Mode Change Instructions and Status:

The module supports the Approved mode of operation by default and transitions implicitly to the Non-Approved mode of operation, when a service from the Non-Approved, Not Allowed Algorithms table from Section 2.5 below is used.

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A4995	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38A
AES-CMAC	A4995	Direction - Generation Key Length - 128, 256	SP 800-38B
AES-CTR	A4995	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38A

Algorithm	CAVP Cert	Properties	Reference
AES-ECB	A4995	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38A
AES-GCM	A4995	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D
AES-KW	A4996	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38F
Counter DRBG	A2721	Prediction Resistance - No, Yes Mode - AES-128, AES-256 Derivation Function Enabled - No	SP 800-90A Rev. 1
ECDSA KeyGen (FIPS186-5)	A5861	Curve - P-384 Secret Generation Mode - extra bits	FIPS 186-5
ECDSA SigGen (FIPS186-4)	A4997	Component - Yes Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-256, SHA2-384, SHA2-512, SHA2-512/256	FIPS 186-4
ECDSA SigVer (FIPS186-4)	A4998	Component - Yes Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-256, SHA2-384, SHA2-512, SHA2-512/256	FIPS 186-4
HMAC-SHA2-224	A4999	Key Length - Key Length: 8-512 Increment 8	FIPS 198-1
HMAC-SHA2-256	A4999	Key Length - Key Length: 8-512 Increment 8	FIPS 198-1
HMAC-SHA2-384	A4999	Key Length - Key Length: 8-1024 Increment 8	FIPS 198-1
HMAC-SHA2-512	A4999	Key Length - Key Length: 8-1024 Increment 8	FIPS 198-1
HMAC-SHA2-512/224	A4999	Key Length - Key Length: 8-512 Increment 8	FIPS 198-1
HMAC-SHA2-512/256	A4999	Key Length - Key Length: 8-1024 Increment 8	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A5861	Domain Parameter Generation Methods - P-384 Scheme - ephemeralUnified - KAS Role - responder	SP 800-56A Rev. 3
KDA HKDF SP800-56Cr2	A5861	Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-1024 Increment 8 HMAC Algorithm - SHA2-384, SHA2-512	SP 800-56C Rev. 2
KDF SP800-108	A4995	KDF Mode - Counter Supported Lengths - Supported Lengths: 128-1024 Increment 128	SP 800-108 Rev. 1

Algorithm	CAVP Cert	Properties	Reference
RSA SigVer (FIPS186-4)	A5000	Signature Type - PKCS 1.5, PKCSPSS Modulo - 3072	FIPS 186-4
SHA2-224	A5001	Message Length - Message Length: 8-65536 Increment 8	FIPS 180-4
SHA2-256	A5001	Message Length - Message Length: 8-65536 Increment 8	FIPS 180-4
SHA2-384	A5001	Message Length - Message Length: 8-65536 Increment 8	FIPS 180-4
SHA2-512	A5001	Message Length - Message Length: 8-65536 Increment 8	FIPS 180-4
SHA2-512/224	A5001	Message Length - Message Length: 8-65536 Increment 8	FIPS 180-4
SHA2-512/256	A5001	Message Length - Message Length: 8-65536 Increment 8	FIPS 180-4

Table 4: Approved Algorithms

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
CKG - Section 4	Key Type:Symmetric and Asymmetric	N/A	NIST SP800-133r2 Section 4: Using the Output of a Random Bit Generator (example 1); The module supports the following per NIST SP 800-133r2: 1. Section 5.2: Key Pairs for Key Establishment 2. Section 6.2.1: Symmetric Keys Generated Using Key-Agreement Schemes 3. Section 6.2.2: Symmetric Keys Derived from a Pre-existing Key

Table 5: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

There are no non-approved, allowed algorithms implemented in the module.

Non-Approved, Allowed Algorithms with No Security Claimed:

There are no non-approved, allowed algorithms with no security claimed, implemented in the module.

Non-Approved, Not Allowed Algorithms:

Name	Use and Function
ECDSA SigVer (186-5) Component	Pre-hashed signature verification function
Brainpool P384r1, P512r1	ECC curve for scalar point multiplication, signing and verification
SECG secp256k1	ECC curve for scalar point multiplication, signing and verification
EdDSA curve Ed25519ph, Ed448	ECC curve for scalar point multiplication, signing and verification

Table 6: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
FW Integrity Check	DigSig-SigVer	Used in integrity check value calculation		RSA SigVer (FIPS186-4): (A5000) Signature Type: PKCSPSS Modulo: 3072 Hash Algorithm: SHA2-512 Public Exponent Mode: Fixed Fixed Public Exponent: 010001
Encryption	BC-UnAuth	Encryption of data as part of Secure Data Object Storage (SDOS) flow		AES-CBC: (A4995) Direction: Encrypt Key Length: 128, 256 AES-ECB: (A4995) Direction: Encrypt Key Length: 128, 256 AES-CTR: (A4995) Direction: Encrypt Key Length: 128, 256 Payload Length: 8, 128

Name	Type	Description	Properties	Algorithms
Key Derivation	CKG KBKDF	Key derivation function for SPDM, attestation based on Device ID. AES-CMAC used as the pseudorandom function (PRF)		KDF SP800-108: (A4995) KDF Mode: Counter MAC Mode: CMAC-AES128, CMAC-AES256 Supported Lengths: 128-1024 Increment 128 Fixed Data Order: Before Fixed Data Counter Length: 32 Custom Key In Length: 0 AES-CMAC: (A4995) Direction: Generation Key Length: 128, 256 MAC Length: 128 Message Length: 0-65536 Increment 8 CKG - Section 4 : () Key Type: Symmetric and Asymmetric
Signature Generation	DigSig-SigGen	Used for digital signatures of Image Signing Keys		ECDSA SigGen (FIPS186-4): (A4997) Component: Yes Curve: P-256, P-384, P-521 Hash Algorithm: SHA2-256, SHA2-384, SHA2-512, SHA2-512/256 SHA2-256: (A5001) Message Length: 8-65536

Name	Type	Description	Properties	Algorithms
				Increment 8 SHA2-384: (A5001) Message Length: 8-65536 Increment 8 SHA2-512: (A5001) Message Length: 8-65536 Increment 8 SHA2-512/256: (A5001) Message Length: 8-65536 Increment 8
Hashing	SHA	Hash for creating digital signatures of image signing keys. Creation of image digests for the various images which require authentication. Measurement and signing functions for SPDM messages and attestation based on Device ID.		SHA2-224: (A5001) Message Length: 8-65536 Increment 8 SHA2-256: (A5001) Message Length: 8-65536 Increment 8 SHA2-384: (A5001) Message Length: 8-65536 Increment 8 SHA2-512: (A5001) Message Length: 8-65536 Increment 8 SHA2-512/224: (A5001) Message Length: 8-65536 Increment 8 SHA2-512/256: (A5001) Message Length: 8-65536 Increment 8
Random Number Generation	DRBG ENT-ESV	Generation of random bits		Counter DRBG: (A2721)

Name	Type	Description	Properties	Algorithms
				Prediction Resistance: Yes, No Reseed: Yes Mode: AES-128, AES-256 Derivation Function Enabled: No Additional Input: 0-256 Increment 8 for AES-128, 0-384 Increment 8 for AES-256 Entropy Input: 256 for AES-128, 384 for AES-256 Nonce: 0 Personalization String Length: 0-256 Increment 8 for AES-128 and 0-384 Increment 8 for AES-256 Returned Bits: 128 for AES-128, 512 for AES-256
Key Wrapping/Unwrapping	BC-AuthDecrypt BC-AuthEncrypt	Key Wrapping in compliance with [SP800-38F] when approved using AES KW	SP 800-38F key wrapping per IG D.G:128 and 256-bit keys providing 128 or 256 bits of encryption strength	AES-KW: (A4996) Direction: Decrypt, Encrypt Cipher: Cipher Key Length: 128, 256 Payload Length: 128-256 Increment 64
ECDSA Key Generation	AsymKeyPair-KeyGen CKG	Used to generate an ECDSA key pair for SPDM, attestation based on Device ID		ECDSA KeyGen (FIPS186-5): (A5861) CKG - Section 4 : () Key Type: Symmetric and Asymmetric
Signature Verification	DigSig-SigVer	Used to authenticate		ECDSA SigVer (FIPS186-4): (A4998)

Name	Type	Description	Properties	Algorithms
		Image Signing Key objects		Curve: P-256, P-384, P-521 Hash Algorithm: SHA2-256, SHA2-384, SHA2-512, SHA2-512/256 SHA2-256: (A5001) SHA2-384: (A5001) SHA2-512: (A5001) SHA2-512/256: (A5001) RSA SigVer (FIPS186-4): (A5000) Padding/Mode: PKCS1.5 Hash: SHA2-384, SHA2-512/256, SHA2-512, SHA2-256 Padding/Mode : PKCSPSS Hash : SHA2-512
Authentication	MAC	Authentication of data as part of the Secure Data Object Storage (SDOS) flow		HMAC-SHA2-224: (A4999) HMAC-SHA2-256: (A4999) HMAC-SHA2-384: (A4999) HMAC-SHA2-512: (A4999) HMAC-SHA2-512/224: (A4999) HMAC-SHA2-512/256: (A4999)
Authenticated Encryption	BC-Auth	Secure manageability traffic, secured messages as per SPDM protocol.		AES-GCM: (A4995) Direction: Encrypt Key Length: 128, 256
Decryption	BC-UnAuth	Decryption of data as part of Secure Data		AES-CBC: (A4995) Direction:

Name	Type	Description	Properties	Algorithms
		Object Storage (SDOS) flow		Decrypt Key Length: 128, 256 AES-ECB: (A4995) Direction: Decrypt Key Length: 128, 256 AES-CTR: (A4995) Direction: Decrypt Key Length: 128, 256 Payload Length: 8, 128
Authenticated Decryption	BC-Auth	Secure manageability traffic, secured messages as per SPDM protocol.		AES-GCM: (A4995) Direction: Decrypt Key Length: 128, 256
HKDF	BC-Auth CKG	Derivation using the hash-based KDF per NIST SP 800-56Cr2		KDA HKDF SP800-56Cr2: (A5861) CKG - Section 4 : () Key Type: Symmetric and Asymmetric
ECC point multiplication	UNK	ECC Point Multiplication (P-384) operation followed by range checking for selected curve		
Entropy Source	ENT-P	Entropy Source (ESV Cert. E136) used to seed the approved NIST SP 800-90Ar1 DRBG		

Name	Type	Description	Properties	Algorithms
KAS-SSC	KAS-SSC	Shared Secret Computation (SSC) per NIST SP 800-56Ar3	IG:IG D.F Scenario 2 path (1)	KAS-ECC-SSC Sp800-56Ar3: (A5861) Scheme: ephemeralUnified KAS Role: responder Domain Parameter Generation Methods: P-384

Table 7: Security Function Implementations

2.7 Algorithm Specific Information

a. AES-GCM Usage
AES GCM IV generation must be compliant to IG C.H Key/IV Pair Uniqueness Requirements from SP 800-38D Scenario 5 per the Security Protocol and Data Model (SPDM) Specification (versions 1.1.1b and 1.1.2) Section 12.6 (derived using the HKDF expand function). The IV is constructed within the module's cryptographic boundary in compliance with the SPDM protocol and shall only be used in the context of the AES-GCM mode encryptions within the protocol.

The Module does not implement the SPDM protocol itself, however, it provides the cryptographic functions required for implementing the protocol. AES GCM encryption is used in the context of the SPDM protocol. The module provides the primitives to support the required AES GCM cipher suites. The module's implementation of AES-GCM is used together with an application that runs outside the module's cryptographic boundary. The application negotiates the protocol session's keys.

The (key, IV) pair collision probability is less than 2^-32 and this is ensured by the IV construction in the context of the SPDM protocol as follows:

IV = HKDF-Expand(major-secret, bin_str6, iv_length);
bin_str6 = BinConcat(iv_length, Version, "iv", NULL)

- Each SPDM session uses a unique major-secret, derived from: The ECDHE shared secret and session-specific nonces. Even if the same bin_str6 inputs are reused across sessions, the major-secret is different every time. Thus, the output of the HKDF-Expand for both key and IV is unique per session i.e., each session produces a unique (key, IV) pair.
- The IV derived from HKDF is not used directly in AES-GCM. Instead, this "IV" serves as a 96-bit salt, and SPDM combines it with a 64-bit monotonically increasing sequence number to build the actual AES-GCM IV. This guarantees, no reuse of an IV within a session.
- SPDM prevents reuse across sessions by:

- Using HKDF with session-unique secrets and labels
- Deriving the IV from HKDF-Expand(major-secret, bin_str6, iv_length), ensuring that even the base salt is different between sessions

When the IV exhausts the maximum number of possible values for a given session key, this results in a failure in encryption and a handshake to establish a new encryption key will be required. It is the responsibility of the user of the module, i.e., the first party, client or server, to encounter this condition, to trigger this handshake in accordance with the SPDM protocol.

In the event that the Module power is lost and restored the user must ensure that the AES GCM encryption/decryption keys are re-distributed in accordance with IG C.H Scenario 3.

b. Component Validation List (CVL)
In accordance with IG 2.4.B, the ECDSA SigGen tested component that may be called during the operation of the module and shown in the module's CVL certificate has been listed individually in the Approved Algorithms table in Section 2.5. All vendor affirmed components that may be called during the operation of the module have also been listed individually in the Vendor Affirmed Algorithms table in Section 2.5 per IG 2.4.B.

c. RSA Usage
Per IG C.F, the RSA SigVer implementation has been tested for the implemented RSA modulus length i.e. 3072 bits since CAVP testing is available. The module does not support generation of RSA keys with any untested moduli/sizes.

d. DRBG Usage
Per IG D.L Additional Comment 2, the CTR_DRBG is seeded with full entropy by the entropy source (ESV Cert. #E136) given that a derivation function is not used.

e. NIST SP 800-108 KDF Usage
The SP 800-108 KDF is not used to generate asymmetric keys. The module implements CKG per NIST SP 800-133r2 Section 6.2.2.

f. FIPS 186-4 Usage
In accordance with IG C.M and IG C.K Additional Comment #3, the FIPS 186-4 claims for ECDSA/RSA algorithms per the following algorithm implementations have been made and are considered to be equivalent to FIPS 186-5: CAVP Certs. #A5000, #A4998 and #A4997.

g. KTS
The module does not establish SSPs using an approved key transport scheme (KTS). However, it does offer approved authenticated algorithms that can be used by an external operator/application as part of an approved KTS.

h. KAS-SSC
The module does not establish SSPs using an approved key agreement scheme (KAS). However, it does offer some or all of the underlying KAS cryptographic functionality to be used by an external operator/application as part of an approved KAS.

2.8 RBG and Entropy

Cert Number	Vendor Name
E136	Intel Corporation

Table 8: Entropy Certificates

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
NIST SP800-90B TRNG Entropy Source	Physical	FNIC CM TRNG E610 Step A0, FNIC CM TRNG E830 Step A0	128 bits	Full Entropy (128 bits)	BlockCipher_DF (A2932)

Table 9: Entropy Sources

The Public Use Document (PUD) for the entropy source validation (ESV Cert. #E136) can be found at: https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/entropy/E136_PublicUse.pdf.

2.9 Key Generation

The module implements a NIST SP 800-90Ar1 CTR_DRBG (Counter DRBG) and supports the following sections per NIST SP 800-133r2 (CKG): Sections 4, 5.2, 6.2.1 and 6.2.2.

2.10 Key Establishment

Key Agreement

Per IG D.F:

The module supports a Key Agreement Scheme per NIST SP800-56Ar3 and IG D.F Scenario 2 (path 1). The KAS-SSC entry in the Security Functions Implementation table in Section 2.6 has been documented accordingly. The Approved Algorithm list includes the tested KAS-ECC-SSC as an individual entry.

Per IG D.F Additional Comment 1.e:
The elliptic curve used in the key agreement scheme (P-384) and the associated domain parameter provides more than 112 bits of security (128 bits supported).

Per IG D.F Additional Comment 2:
The KAS-ECC-SSC implementation supports a scheme of the Diffie-Helllman variety (EphemeralUnified).

Key Wrapping:
The module does not establish SSPs using an approved key transport scheme (KTS). However, it does offer approved authenticated algorithms that can be used by an external operator/application as part of an approved KTS.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
CSR Control Interface, Fuse inputs	Data Input	CSR control interfaces to transaction initiators including the FW engine which implements the FW portion of the module. Fuse inputs from One-Time-Programmable Non-Volatile fuse macros, input data to SBB for encryption, decryption, hashing, signature operations and key derivation
AMBA (on-chip system bus)	Data Output	Output data resulting from cryptographic operations and execution of services
AMBA (on-chip system bus)	Control Input	Reset signal
AMBA (on-chip system bus)	Status Output	Module status, Interrupt Status Register output after completion of a service
Physical power connector	Power	Power input to the module

Table 10: Ports and Interfaces

The module does not implement a control output interface.

4 Roles, Services, and Authentication

4.1 Authentication Methods

The module does not implement authentication.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
CO	Role	Crypto Officer	None

Table 11: Roles

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Show Status	Status returned by the module	Module status	None	Module Status	None	CO

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		can be obtained via its current operational state (approved mode which is identified by approved service indicator i.e. fips_approved flag set to True, non-approved mode which is identified by non-approved service indicator i.e. fips_approved flag not set and error state which is identified by looking at the self-test failure message)				
Initialization (INIT) (Perfor	Self-tests can be executed by rebooting the module,	Approved mode indicator (1) or	Reset signal	Self-Test status	FW Integrity Check	CO - Storage Root Keys (SRK): E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
m Self-Tests)	triggers the INIT function which results in the SRK and SBKEK key derivations	error code returned from the module, in case of failure				- Secure Boot Key Encryption Keys (SBKEK): E Unauthenticated
Zeroisation	Zeroization can be executed by rebooting the module	Approved mode indicator (1)	Reset signal	None	None	CO - Storage Root Keys (SRK): Z - Secure Boot Key Encryption Keys (SBKEK): Z - Unique Device Secret (UDS): Z - UDS Hash (UDS_SHA): Z - Signature Public Key Authentication Key (SigKAKPub): Z - OEM Public Key Authentication Key (OEMKAKPub): Z - OEM Public Key Authentication Key Object Hash (OEMKAKPub_Hash): Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- Image Signing Public Keys (ISKPub): Z - Image Encryption Key (CR_IEK) : Z - Authentication Signing Public Keys : Z - SDO encryption / decryption key (SDOencrypt): Z - SRK Hash (SRK_SHA): Z - SDO authentication key (SDOauth) : Z - Device ID Private Key (DevIDpriv): Z - Device ID Public Key (DevIDPub): Z - Attestation Manifest Metal Key: Z -

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Attestation Manifest Private Key (AttManpriv): Z - Attestation Manifest Public Key (AttMan pub): Z - Secure Manageability Keys - Symmetric data-encryption key and Symmetric authentication key: Z - DRBG Entropy Input : Z - DRBG Seed: Z - DRBG State: Z - HKDF KDK: Z - HKDF Derived Keying Material: Z - ECDH Private Key: Z - ECDH Public Key: Z - Shared Secret (Z): Z - Image Signing Public

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Keys (ISKPub) Hash : Z
Image Verification (Code: 1)	SBB function to perform image integrity and authentication check. This function reads the image one time from memory and performs verification and decryption at the same time	Interrupt Status Register (ISR) bit0 value 1 for success, 0 for failure	Hash function, signature algorithm, signature key size, image pointer	Image hash, Status in Interrupt Status Register	FW Integrity Check Signature Verification	CO - Signature Public Key Authentication Key (SigKAKPub): W,E
Image Signing (Code: 2)	Performs verification of Signed Key Object Chain, image integrity hashing and signature creation, creation of secure boot headers and trailers, and optional image encryption	ISR bit0 value 1 for success, 0 for failure	Hashing algorithm, Signing Algorithm, Boot image source address, SBB image destination address, encrypt image flag, version information, SBKEK index, ROT bitmap, IV, image encryption key, key object source address, image signing private key, random value	secure boot image, status in Interrupt Status Register	Encryption Signature Generation	CO - Secure Boot Key Encryption Keys (SBKEK): W,E - Image Encryption Key (CR_IEK) : W,E
Create SDO (Code: 3)	Encrypt and integrity protect any data object for storage anywhere in the	ISR bit0 value 1 for success,	data source address, SDO destination address, data	encrypted and integrity protected SDO data	Encryption Hashing	CO - Storage Root Keys (SRK): E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	system (for example, SPI flash, DRAM, and so on) by using a key derived from the Storage Root Key and the Owner_Info field	0 for failure	length, owner information, object information, SRK index, IV	at provided destination address		
Get Data Object from SDO (Code: 4)	Provides ability to recover plaintext from an SDO by decryption and integrity checking of an SDO from anywhere in the system (for example, SPI flash, DRAM, and so on) by using a key derived from the Storage Root Key and the Owner_Info field	ISR bit0 value 1 for success, 0 for failure	SDO source address, data destination address, owner information	Object information	Hashing Decryption	CO - Storage Root Keys (SRK): E
Write Values from Entropy Source (Code: 10)	The Write Values from Entropy Source function outputs values from the ring oscillator entropy source. This function is supported only for entropy assessment and is only enabled when the SBB is in TRNG entropy collection mode.	ISR bit0 value 1 for success, 0 for failure	Data destination address, data length	Interrupt Status Register Done bit, Write entropy bits to destination address	Random Number Generation Entropy Source	CO

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Encrypt Data (Code: 15)	The Encrypt Data function can encrypt any data object for storage anywhere in the system (for example, SPI flash, DRAM, and so on) by using a key loaded with the function or taken from one of the SRK derivated keys. This function provides only encryption and no integrity function. If you need an integrity function for the data, use this function with the Calculate Hash or SDO functions.	ISR bit0 value 1 for success, 0 for failure	Plaintext data, Encryption key, IV	Ciphertext	Encryption	CO - Storage Root Keys (SRK): E
Decrypt Data (Code: 16)	The Decrypt Data function can decrypt any data object by using a key loaded with the function or taken from one of the SRK derivated keys. This function provides only decryption and no integrity function. If you need an integrity function for the	ISR bit0 value 1 for success, 0 for failure	Ciphertext, Decryption Key, IV	Plaintext data	Decryption	CO - Storage Root Keys (SRK): E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	data, use this function with the Calculate Hash or SDO functions.					
Calculate Hash (Code 17)	The Calculate Hash function can calculate a SHA hash on any data stored anywhere in the system (for example, DRAM, and so on). If a keyed hash function is requested, this function uses a key loaded with the function. This function provides only an integrity function. If you need an encryption or decryption function for the data, use this function with the Encrypt Data, Decrypt Data, or SDO functions	ISR bit0 value 1 for success, 0 for failure	data to hash, hash key	hashed data	Hashing Authentication	CO - SDO authentication key (SDOauth): E
Signature Generation (Code: 18)	Run a signature generation function by using parameters provided in the communication registers. The resultant signature is written to the communication registers and is	ISR bit0 value 1 for success, 0 for failure	Signing algorithm, hashing algorithm, hash, private key	Signature	Key Derivation Signature Generation	CO - Device ID Private Key (DevIDpriv): G,E - Attestation Manifest Private Key (AttManpriv): G,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	available to the function caller.					
Authenticated Encryption With Additional Data (Code: 21)	function sbb_drv_calc_aead_aes_gcm() performs authenticated encryption and decryption using AES-GCM (128/256). This function encrypts or decrypts any data object by using a key loaded with the function or taken from one of the SRK derived keys. This function also generates an integrity check value (authentication tag) for authenticating the encrypted data and additional data	ISR bit0 value 1 for success, 0 for failure	source data address, destination data address, data length, AEAD flags, additional authenticated data length, IV, encryption key	Plaintext/ Ciphertext, ICV hash	Authenticated Encryption Authenticated Decryption HKDF	CO - GCM Session IV: G,E - Secure Manageability Keys - Symmetric data-encryption key and Symmetric authentication key: G,E
ECC Point Multiplication (Code: 24)	Elliptic curve multiplication is the operation of successively adding a point along an elliptic curve to itself repeatedly. It is used in elliptic curve cryptography (ECC) as a means of producing a	ISR bit0 value 1 for success, 0 for failure	ECC curve, scalar, ECC point X coordinate, ECC point Y coordinate	ECC point X coordinate, ECC point Y coordinate	ECC point multiplication	CO

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	one-way function					
Wrap Image Encryption Key (Code: 26)	This function takes an Image Encryption Key (IEK) received as an input parameter and wraps it with the selected secret Secure Boot Key Encryption Key (SBKEK) without exposing the SBKEK value.	ISR bit0 value 1 for success, 0 for failure	SBKEK index, image encryption key	wrapped image encryption key	Key Wrapping/ Unwrapping	CO - Secure Boot Key Encryption Keys (SBKEK): E - Image Encryption Key (CR_IEK) : W,E
SPDM Initialization (Code: 29)	This function allows software to configure the SBB for future SPDM operations	ISR bit0 value 1 for success, 0 for failure	Signing algorithm, hashing algorithm	DeviceID public key	None	CO - Device ID Public Key (DevIDPub): R
SPDM Get Device ID Attestation Manifest (Code: 30)	This function returns the signed attestation manifest. The attestation manifest contains the DeviceID public key and metadata about the device. It is signed with the Attestation Manifest Private Key that is derived by this function	ISR bit0 value 1 for success, 0 for failure	template source address, signing key algorithm, measurement hash algorithm, key ID, signature offset	DeviceID public key	Key Derivation Signature Generation	CO - Attestation Manifest Private Key (AttManpriv): G,E - Attestation Manifest Public Key (AttMan pub): G,E
SPDM Challenge Response	The SPDM Challenge Response function returns the signed response to the	ISR bit0 value 1 for success, 0 for failure	response source address, response destination address,	signed SPDM Challenge Response	Key Derivation Signature Generation Hashing	CO - Device ID Private Key (DevIDpriv): G,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
(Code: 31)	SPDM CHALLENGE request.		challenge request parameter, opaque length			
SPDM Get Measurements Response (Code: 32)	The SPDM Get Measurements Response function returns the signed response to the SPDM GET_MEASUREMENTS request	ISR bit0 value 1 for success, 0 for failure	measurement source address, measurement destination address, measurement request parameter, opaque length	signed measurement response	Key Derivation Signature Generation Hashing	CO - Device ID Private Key (DevIDpriv): G,E
SPDM Key Exchange (KEX) Response Message (Code: 33)	The SPDM Key Exchange Response function returns the signed response to the SPDM KEY_EXCHANGE request	ISR bit0 value 1 for success, 0 for failure	KEX source address, KEX destination address, request parameter, opaque length	signed KEX response	Signature Generation	CO - Device ID Private Key (DevIDpriv): G,E
SPDM Measure (Code: 34)	The SPDM Measure function calculates a measurement hash of the input data and stores the measurement in the SBB measurement registers corresponding to the selected measurement index	ISR bit0 value 1 for success, 0 for failure	data source address, measurement index, measurement type	Measurement hash	Hashing	CO
SPDM Add Measurement	The SPDM Add Measurement function stores input data from	ISR bit0 value 1 for success,	measurement index, measurement	None	None	CO

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
(Code: 35)	software as a measurement in the SBB measurement registers corresponding to the specified measurement index	0 for failure				
SPDM Clear Measurements (Code: 36)	The SPDM Clear Measurements function clears the SBB measurement registers corresponding to the measurement indices specified in the input bitmap	ISR bit0 value 1 for success, 0 for failure	measurements bitmap	None	None	CO
Wrap Key (Code: 37)	The Wrap Key function performs an SP 800-38F key wrap using AES-ECB on a key which is an input to this function	ISR bit0 value 1 for success, 0 for failure	Key encryption key (KEK), key to be wrapped	wrapped key	Key Wrapping/ Unwrapping	CO - Secure Boot Key Encryption Keys (SBKEK): E
Unwrap key (Code: 38)	The Unwrap Key function performs an SP 800-38F key unwrap using AES-ECB on a wrapped key which is an input to this function	ISR bit0 value 1 for success, 0 for failure	Key encryption key (KEK), wrapped key	Unwrapped key	Key Wrapping/ Unwrapping	CO - Secure Boot Key Encryption Keys (SBKEK): E
Get Version Numbers	The Get Version Numbers function writes version information	ISR bit0 value 1 for success, 0 for failure	None	SBB identifier, SBB version, crypto version,	None	CO

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
(Code: 39)	about the SBB to output registers			ROM version, ROM creation date, ROM creation time		
Get SBB State (Code: 49)	The Get SBB State function returns various SBB state information	ISR bit0 value 1 for success, 0 for failure	None	Various state data	None	CO
Get Module Information (Code: 0xF105) (Show module's versioning information)	Provides module information	"FW version: ID = 'SBB Hybrid Cryptographic Module' version = 1.00.01, sbb version = 5.0.13 (hex) = 5.0.19 (dec)"	Function call GetModuleInfo	"FW version: ID = 'SBB Hybrid Cryptographic Module' version = 1.00.01, sbb version = 5.0.13 (hex) = 5.0.19 (dec)"	None	CO
Hash based Key Derivation (Code: 0xF100)	This firmware service performs Hash based Key Derivation using SHA-384, SHA-512	fips_approved_flag value 0 for success, non-0 for failure	Key material	Derived Key	HKDF	CO - HKDF KDK: G,E - HKDF Derived Keying Material: G
Key Exchange (Code: 0xF103)	This service generates an ECDHE P-384 keypair and uses it during key exchange to create the DHE secret	fips_approved_flag value 0 for success, non-0 for failure	Function call ExchangeECDSAP384Key	Shared secret	Random Number Generation ECDSA Key Generation Entropy Source KAS-SSC	CO - ECDH Private Key: G,E - ECDH Public Key: G,E - Shared

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Secret (Z): G,E
Get TRNG Alarm (Code: 0xF109)	This service is used to identify if a SBB TRNG alarm was detected and that the SBB memory was clean	fips_approved_flag value 0 for success, non-0 for failure	Get TRNG Alarm function called	A return code associated with the alarm raised	None	CO
Get Sequence Number (Code: 0xF108)	Should be sent after session was created. It returns the SPDM sequence number for the selected session. The number refers to the sequence number for the encryption of SPDM RESPONSE messages	fips_approved_flag value 0 for success, non-0 for failure	origin (SPDM over DoE or SPDM over MCTP)	64 bit sequence number	None	CO
Authenticated Encryption (Code: 0xF101)	Encrypt message using AES GCM and IV internally generated	fips_approved_flag value 0 for success, non-0 for failure	Plaintext, Encryption key, session origin id	Ciphertext	Encryption Key Derivation	CO - SDO encryption / decryption key (SDOencrypt): G,E
Authenticated Decryption (Code: 0xF101)	Decrypt message using AES GCM	fips_approved_flag value 0 for success, non-0 for failure	Ciphertext, Decryption key	Plaintext	Key Derivation Decryption	CO - SDO encryption / decryption key (SDOencrypt): G,E
AES GCM Clear Session IV	AES GCM IV zeroization	fips_approved_flag value 0 for success,	session origin id	None	None	CO - GCM Session IV: Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
(Code: 0xF107)		non-0 for failure				
Create Session IV (Code: 0xF106)	96 bits IV generation	fips_approved_flag value 0 for success, non-0 for failure	secret (upto 48 bytes), session origin id, SPDM version	Initialization Vector	HKDF	CO - GCM Session IV: G
Get SBB Lock	Function SBB_drb_get_lock() used to get exclusive SBB access	fips_approved_flag value 0 for success, non-0 for failure	Function call SBB_drb_get_lock()	completion status code	None	CO
Release SBB Lock	Function SBB_drv_release_lock() to release SBB lock	fips_approved_flag value 0 for success, non-0 for failure	Function call SBB_drv_release_lock()	completion status code	None	CO
eFuses Reload and Key Validation	Requests that the eFuses be reloaded, checks all keys to see if they are valid (non-zero and not all ones), runs private key validation routine to compare keys to programmed SHA values, performs SRK and SBKEK key derivation	fips_approved_flag value 0 for success, non-0 for failure	N/A	N/A	Key Derivation	CO - Storage Root Keys (SRK): E - Secure Boot Key Encryption Keys (SBKEK): E
Remove (Overwrite) SDO	The Remove Secure Data Object (SDO) function deletes an SDO from the memory	fips_approved_flag value 0 for success,	Owner_Info, Valid SRK	N/A	Key Derivation Authentication	CO - Storage Root Keys (SRK): E - SDO authenticat

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	location indicated if all the checks and the integrity check pass. It also derives keys from the SRK and checks the SDO header (using an HMAC-SHA2-256).	non-0 for failure				ion key (SDOauth) : E
Signature Verification	The Signature Verification function verifies a signature by using a hash result that is included as a parameter	fips_approved_flag value 0 for success, non-0 for failure	hash function, signature, signature and hashing algorithms and public key	result of the verification	Signature Verification	CO - Signature Public Key Authentication Key (SigKAKPub): W,E
Generate 16 Byte Random Number	The Generate 16 Byte Random Number function runs a NIST SP800-90 DRBG generate function and returns a 16 byte (128 bit) random number	fips_approved_flag value 0 for success, non-0 for failure	Function call i.e. function code 22 (decimal)	128-bit random number	Random Number Generation Entropy Source	CO - DRBG Entropy Input : E - DRBG Seed: E - DRBG State: E
In-place Image Verification	The function checks all needed parameters and keys, reads the image data from memory, checks the image header, performs selected functions to create integrity SHA value, run signature	fips_approved_flag value 0 for success, non-0 for failure	Function call i.e. function code 23 (decimal)	N/A	Hashing Signature Verification	CO - Signature Public Key Authentication Key (SigKAKPub): W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	checking, check AES key, and check firmware revision					
Clear Active Key	Removes the last verified public key, key ID and key flags from internal SBB registers.	fips_approved_flag value 0 for success, non-0 for failure	Function call i.e. function code 25 (decimal)	N/A	None	CO - Signature Public Key Authentication Key (SigKAKPub): Z - OEM Public Key Authentication Key (OEMKAKPub): Z - Image Signing Public Keys (ISKPub): Z - Authentication Signing Public Keys : Z - SDO authentication key (SDOauth) : Z - Device ID Public Key (DevIDPub): Z - Attestation Manifest Public Key (AttManpub): Z
Verify LT-CSS	The Verify LT-CSS Signed ISK function verifies the KAK	fips_approved_flag value 0 for	CSS Signed ISK Source Address, ROT Bitmap	KAKpub Hash (regardless of	Hashing Signature Verification	CO - Signature Public Key Authenticat

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Signed ISK	public key inside the ISK object against the selected ROT in fuses, performs key revocation checking, and verifies the ISK object signature	success, non-0 for failure	(bit 0 - Signature, bit 1 - OEM)	verification status)		ion Key Object Hash (SigKAKPub_Hash): W,E
Verify LT-CSS Signed Image	The Verify LT-CSS Signed Image function verifies the ISK public key inside the boot image against the hash that was saved from the ISK object by the Verify LT-CSS Signed ISK function. It also performs security version checking, and verifies the boot image signature.	fips_approved_flag value 0 for success, non-0 for failure	CSS Signed ISK Source Address, Data length, Measurement Index	Version Information, Image Hash, ISKpub Hash	Hashing Signature Verification	CO - Image Signing Public Keys (ISKPub): E

Table 12: Approved Services

4.4 Non-Approved Services

Name	Description	Algorithms	Role
Signature Verification	ECDSA signature verification with an externally provided hash	ECDSA SigVer (186-5) Component	CO
Point Multiplication	ECC Curve for scalar point multiplication	Brainpool P384r1, P512r1 SECG secp256k1 EdDSA curve Ed25519ph, Ed448	CO
Signature Generation and Verification	ECC Curve for signing and verification	Brainpool P384r1, P512r1 SECG secp256k1	CO

Name	Description	Algorithms	Role
		EdDSA curve Ed25519ph, Ed448	

Table 13: Non-Approved Services

4.5 External Software/Firmware Loaded

The module supports loading firmware from an external source. The module firmware is loaded afresh on each boot, and this is a complete image replacement. In accordance with IG 10.3.F, a firmware load test does not apply to the module. The CO role is responsible for loading the firmware on the module as by booting up the underlying SoC.

5 Software/Firmware Security

5.1 Integrity Techniques

The Module uses an RSA mod 3072 SHA2-512 signature verification (CAVP Cert. #A5000) as the approved integrity technique. The RSA public key (Image Signing Public Keys (ISKPub)) used for the integrity test is considered a non-SSP. The RSA 3072 SHA2-512 signature verification CAST is performed prior to the firmware integrity test. The module firmware is in an executable form.

5.2 Initiate on Demand

The pre-operational firmware integrity test can be initiated on demand by power-cycling the Intel SoC i.e. resetting the module.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Limited

7 Physical Security

7.1 Mechanisms and Actions Required

Mechanism	Inspection Frequency	Inspection Guidance
Standard passivation applied to the Intel SoC	N/A	N/A

Table 14: Mechanisms and Actions Required

The Intel SOC chip is a single chip with production grade IC packaging and hence conforms to the Level 1 requirement for physical security. The chip is a standard integrated circuit with uniform exterior material and standard connectors. It is a commercial/industrial grade chip in regard to power and voltage ranges, temperature, reliability, shock and vibration.

8 Non-Invasive Security

8.1 Mitigation Techniques

The module does not implement any non-invasive attack mitigation techniques.

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
SRAM	Non-Volatile Memory accessible by SBB	Static
Registers	Each module contains configuration and status registers (CSRs). SBB has registers which are exposed to firmware and others which are internal only	Dynamic
eFuses	eFuses are write-once non-volatile storage elements. SBB accesses eFuse values via the EFuse Controller Module (ECM)	Static
RAM	Volatile Memory	Dynamic
In the context of the Image Signing service	Provided as an input to the service and zeroised thereafter	Dynamic

Table 15: Storage Areas

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
Registers (CSRs) to External Endpoint	Registers	External Endpoint	Plaintext	Manual	Electronic	
Input at manufacturing	Input by the manufacturer	eFuses	Plaintext	N/A	N/A	

Table 16: SSP Input-Output Methods

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Reset SBB memory block	SBB zeroisation service triggers a reset of the entire SBB block which causes the contents of all the SBB registers to revert to their reset values and all internal memory to be erased. This also resets the SBB TRNG.	Zeroisation during the module reset prevents the reuse of the SSP parameter	Crypto Officer resets the module
Post cryptographic operation zeroisation	Within the cryptographic cores, all storage spaces containing sensitive parameters are cleaned up by writing zeros after each cryptographic operation	Zeroisation after every cryptographic operation prevents the reuse of the SSP parameters	N/A
SoC reset	SSPs stored in volatile memory (RAM) zeroised on power-cycling/resetting the SoC	Zeroisation on SoC power-cycle/reset	Crypto Officer power-cycles/resets the SoC

Table 17: SSP Zeroization Methods

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Storage Root Keys (SRK)	These root keys are used to derive the symmetric encryption, decryption and authentication keys for Secure Data Object Storage. Three root keys are populated during manufacturing.	256 bits - 256 bits	Symmetric Key - CSP			Encryption Authentication Decryption

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Secure Boot Key Encryption Keys (SBKEK)	SBKEK keys are received in SBB fuses. They are used by SBB for integrity check, image encryption	256 bits - 256 bits	Symmetric Key - CSP			Encryption
Unique Device Secret (UDS)	This root key is used to derive the Device ID for SPDM. This root key is populated during manufacturing using value provided by a TRNG. nitialized during manufacture by manufacturing FW. To validate that the fuse provisioning has taken effect, the value will be hashed by SBB, and the result compared to the expected hash stored in additional fuses.	256 bits - 256 bits	Secret - CSP			Key Derivation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
UDS Hash (UDS_SHA)	Used for UDS integrity checking. Upper 128 bits of a SHA2-512 hash of UDS.	128 bits - 256 bits	Hash - CSP	Hashing		Hashing
Signature Public Key Authentication Key (SigKAKPub)	Public signature verification key used to authenticate image signing objects. Initialized during manufacture by KAK owner.	3072 bits - 128 bits	Public Key - PSP			FW Integrity Check Signature Verification
Signature Public Key Authentication Key Object Hash (SigKAKPub_Hash)	These hashes are used to verify the KAK public key objects stored on SPI Flash	SHA2-512 - 256 bits	Hash - CSP	Hashing		Hashing
OEM Public Key Authentication Key (OEMKAKPub)	These keys are used to authenticate key objects associated with the Option ROM.	3072 bits - 128 bits	Public Key - PSP			Signature Verification
OEM Public Key Authentication Key Object Hash (OEMKAKPub_Hash)	These hashes are used to verify the KAK public key objects	SHA2-512 - 256 bits	Hash - CSP	Hashing		Hashing

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Image Signing Public Keys (ISKPub)	Authentication of the miniloader firmware, the full firmware with configuration and, optionally, the topology netlist and the Option ROM	3072 bits - 128 bits	Public Key - Neither			FW Integrity Check Signature Verification
Image Encryption Key (CR_IEK)	Key used for optional encryption after signing	256 bits - 256 bits	Symmetric Key - CSP			Encryption Authenticated Encryption Decryption Authenticated Decryption
Authentication Signing Public Keys	These keys are used to authenticate the Option ROM if signed by the OEM	P-256, P-384, P-521 - 128, 192, 256 bits	Public Key - PSP			Signature Verification
SDO encryption / decryption key (SDOencrypt)	Used for encryption and decryption of Secure Data Objects. The IV is generated by the SBB using the TRNG. If the encrypted data is not a multiple of the block	256 bits - 256 bits	Symmetric Key - CSP	Key Derivation		Encryption Decryption

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	size, firmware appends pad bytes up to the block size. Firmware will not add more than 16 bytes of padding. Pad bytes will consist of monotonically increasing numbers, starting with the number 0x01					
SRK Hash (SRK_SHA)	Upper 128 bits of a SHA-512 hash of each SRK. Used for SRK integrity checking	128 bits - 256 bits	Hash - CSP	Hashing		Hashing
SDO authentication key (SDOauth)	Symmetric authorization key. Used for authentication of Secure Data Objects	HMAC SHA2-256, 256 bits - 128 bits	Symmetric Key - CSP	Key Derivation		Authentication
Device ID Private Key (DevIDpriv)	Private Key used to sign messages during attestation flows	P-256, P-384, P-521 - 128, 192, 256 bits	Private Key - CSP	Key Derivation		Signature Generation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	including SPDM					
Device ID Public Key (DevIDPub)	Public Key used by attestation requestor to verify the signature of messages signed using the Device ID private key	P-256, P-384, P-521 - 128, 192, 256 bits	Public Key - PSP	ECC point multiplication		Signature Verification
Attestation Manifest Metal Key	This root key is used to derive the Attestation manifest key pairs for export of DeviceID public key for external certificate generation	256 bits - 256 bits	Symmetric Key - CSP			Key Derivation
Attestation Manifest Private Key (AttManpriv)	Private Key used to sign Attestation Manifest data structure which is used to export out the DeviceID public key in manufacturing	P-256, P-384, P-521 - 128, 192, 256 bits	Private Key - CSP	Key Derivation		Signature Generation
Attestation Manifest Public Key (AttMan pub)	Public Key used to verify the Attestation Manifest data structure	P-256, P-384, P-521 - 128, 192, 256 bits	Public Key - PSP	ECC point multiplication		Signature Verification

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	which is exported in manufacturing					
Secure Manageability Keys - Symmetric data-encryption key and Symmetric authentication key	AEAD for secure manageability traffic	256 bits - 256 bits	Symmetric Key - CSP	HKDF		Authenticated Encryption Authenticated Decryption
GCM Session IV	Initialization Vector used in GCM encryption and decryption	96 bits - 96 bits	IV - CSP	HKDF		Authenticated Encryption Authenticated Decryption
DRBG Entropy Input	Entropy input from the entropy source used for DRBG seeding	128 - 256 bits - 128 - 256 bits	Entropy input - CSP	Entropy Source		Random Number Generation
DRBG Seed	Seed generated from the entropy input for the DRBG	128 - 256 bits - 128 - 256 bits	DRBG Seed - CSP	Random Number Generation		Random Number Generation
DRBG State	In accordance with the IG D.L Resolution 3., the values of V and Key are the "secret values" of the internal state of the CTR_DRBG	128, 192, 256 - 128, 192, 256	DRBG State - CSP	Random Number Generation		Random Number Generation
HKDF KDK	Key derivation key used	256 or 384 bits -	Symmetric Key - CSP	HKDF		HKDF

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	with the NIST SP 800-56Cr2 HKDF	256 or 384 bits				
HKDF Derived Keying Material	Keying material derived using the NIST SP 800-56Cr2 HKDF	2048 bits - 112 bits	DKM - CSP	HKDF		
ECDH Private Key	Private key of keypair used in EC Diffie-Hellman Key Exchange	P-384 - 128 bits	Private Key - CSP	Random Number Generation ECDSA Key Generation		KAS-SSC
ECDH Public Key	Public key of keypair used in EC Diffie-Hellman Key Exchange	P-384 - 128 bits	Public Key - PSP	Random Number Generation ECDSA Key Generation		KAS-SSC
Shared Secret (Z)	Shared secret (Z) value computed during the EC Diffie-Hellman Key Exchange	P-384 - 128 bits	Shared secret - CSP		KAS-SSC	
Image Signing Public Keys (ISKPub) Hash	These hashes are used to verify the ISK public key	SHA2-384 - 384 bits	Hash - CSP	Hashing		Hashing

Table 18: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Storage Root Keys (SRK)	Input at manufacturing	eFuses:Plaintext	Based on SRK Policy	Reset SBB memory block	
Secure Boot Key Encryption Keys (SBKEK)	Input at manufacturing	eFuses:Plaintext		Reset SBB memory block Post cryptographic operation zeroisation	
Unique Device Secret (UDS)	Input at manufacturing	eFuses:Plaintext	Based on Device ID Key Policy	Reset SBB memory block Post cryptographic operation zeroisation	Device ID Private Key (DevIDpriv):Derives Device ID Public Key (DevIDPub):Derives
UDS Hash (UDS_SHA)	Registers (CSRs) to External Endpoint	Registers:Plaintext eFuses:Plaintext	Based on Device ID Key Policy	Reset SBB memory block	Unique Device Secret (UDS):Derived From
Signature Public Key Authentication Key (SigKAKPub)	Input at manufacturing	Registers:Plaintext	Based on KAK Policy	Reset SBB memory block	Image Signing Public Keys (ISKPub):Signed with
Signature Public Key Authentication Key Object Hash (SigKAKPub_Hash)	Registers (CSRs) to External Endpoint	eFuses:Plaintext Registers:Plaintext	Based on KAK Policy	Reset SBB memory block	Signature Public Key Authentication Key (SigKAKPub):Hashes
OEM Public Key Authentication Key (OEMKAKPub)	Input at manufacturing	Registers:Plaintext	Based on KAK Policy	Reset SBB memory block	

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
OEM Public Key Authentication Key Object Hash (OEMKAKPub_Hash)	Registers (CSRs) to External Endpoint	eFuses:Plaintext Registers:Plaintext	Based on KAK Policy	Reset SBB memory block	OEM Public Key Authentication Key (OEMKAKPub):Hashes
Image Signing Public Keys (ISKPub)	Input at manufacturing	Registers:Plaintext	Based on ISK and Signed Key Object Policy	Reset SBB memory block	Signature Public Key Authentication Key (SigKAKPub):Encrypted with
Image Encryption Key (CR_IEK)	Input at manufacturing	Registers:Plaintext	Based on ISK and Signed Key Object Policy	Post cryptographic operation zeroisation	
Authentication Signing Public Keys	Input at manufacturing	Registers:Plaintext	Duration based on ISK and Signed Key Object Policy	Reset SBB memory block	
SDO encryption / decryption key (SDOencrypt)		Registers:Plaintext	Based on SDO Policy	Reset SBB memory block Post cryptographic operation zeroisation	Storage Root Keys (SRK):Derived From
SRK Hash (SRK_SHA)	Registers (CSRs) to External Endpoint	eFuses:Plaintext Registers:Plaintext	Based on SRK Policy	Reset SBB memory block	Storage Root Keys (SRK):Hashes
SDO authentication key (SDOauth)		Registers:Plaintext	Based on SDO Policy	Reset SBB memory block Post cryptographic operation	Storage Root Keys (SRK):Derived From

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
				zeroisation	
Device ID Private Key (DevIDpriv)		Registers:Plaintext	Based on Device ID Key Policy	Reset SBB memory block	Device ID Public Key (DevIDPub):Paired With
Device ID Public Key (DevIDPub)	Registers (CSRs) to External Endpoint	Registers:Plaintext	Based on Device ID Key Policy	Reset SBB memory block	Device ID Private Key (DevIDpriv):Paired With
Attestation Manifest Metal Key	Input at manufacturing	SRAM:Plaintext	Based on Attestation Manifest Key Policy	Reset SBB memory block	Attestation Manifest Private Key (AttManpriv):Derives Attestation Manifest Public Key (AttMan pub):Derives
Attestation Manifest Private Key (AttManpriv)		SRAM:Plaintext	Based on Attestation Manifest Key Policy	Reset SBB memory block	Attestation Manifest Public Key (AttMan pub):Paired With Attestation Manifest Metal Key:Derived From
Attestation Manifest Public Key (AttMan pub)	Registers (CSRs) to External Endpoint	SRAM:Plaintext Registers:Plaintext	Based on Attestation Manifest Key Policy	Reset SBB memory block	Attestation Manifest Private Key (AttManpriv):Paired With Attestation Manifest Metal Key:Derived From
Secure Manageability Keys - Symmetric data-encryption key and Symmetric authentication key		RAM:Plaintext	Based on Secure Manageability Policy	SoC reset	GCM Session IV:Used With
GCM Session IV		RAM:Plaintext	Based on Secure Manageability Policy	SoC reset	Secure Manageability Keys - Symmetric data-encryption key and

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
					Symmetric authentication key:Used With
DRBG Entropy Input		RAM:Plaintext	Until module reset	SoC reset	DRBG Seed:Used to derive
DRBG Seed		RAM:Plaintext	Until module reset	SoC reset	DRBG Entropy Input :Derived From
DRBG State		RAM:Plaintext	Until module reset	SoC reset	DRBG Seed:Derived From
HKDF KDK		RAM:Plaintext	Until module reset	SoC reset	HKDF Derived Keying Material:Derives
HKDF Derived Keying Material		RAM:Plaintext	Until module reset	SoC reset	HKDF KDK:Derived From
ECDH Private Key		RAM:Plaintext	Until module reset	SoC reset	ECDH Public Key:Paired With Shared Secret (Z):Derives
ECDH Public Key		RAM:Plaintext	Until module reset	SoC reset	ECDH Private Key:Paired With Shared Secret (Z):Derives
Shared Secret (Z)		RAM:Plaintext	Until module reset	SoC reset	ECDH Private Key:Derived From ECDH Public Key:Derived From
Image Signing Public Keys (ISKPub) Hash	Registers (CSRs) to External Endpoint	Registers:Plaintext	Based on ISK and Signed Key Object Policy	Reset SBB memory block	Image Signing Public Keys (ISKPub):Hashes

Table 19: SSP Table 2

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
RSA SigVer (FIPS186-4) (A5000)	Mode: PKCSPSS; Modulus 3072 bits, Hash: SHA2-512	KAT	SW/FW Integrity	Success: return code: 0; Failure: SBB_STATUS_AUTH_FAIL(33554442) and INTEGRITY_SELF_TEST_ERROR (16777225)	RSA Signature Verification of the firmware

Table 20: Pre-Operational Self-Tests

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
KDA HKDF SP800-56Cr2 (A5861)	SHA2-384, SHA2-512	KAT	CAST	Return code zero (0) for success, non-zero for failure	Key Derivation	Module boot
KAS-ECC-SSC Sp800-56Ar3 (A5861)	P-384	KAT	CAST	Return code zero (0) for success, non-zero for failure	Shared Secret Computation	Module boot
ECDSA KeyGen (FIPS186-5) (A5861)	P-384	PCT	PCT	Return code zero (0) for success, non-zero for failure	Key Generation	On ECDSA keypair generation
AES-CBC (A4995) - Encrypt	128 bits	KAT	CAST	Return code zero (0) for success, non-zero for failure	Encrypt	Module boot
AES-CBC (A4995) - Decrypt	128 bits	KAT	CAST	Return code zero (0) for success, non-zero for failure	Decrypt	Module boot

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-GCM (A4995)	256 bits	KAT	CAST	Return code zero (0) for success, non-zero for failure	Encrypt	Module boot
KDF SP800-108 (A4995)	256 bits	KAT	CAST	Return code zero (0) for success, non-zero for failure	Key Derivation	Module boot
AES-KW (A4996) - Encrypt	256 bits	KAT	CAST	Return code zero (0) for success, non-zero for failure	Encrypt	Module boot
AES-KW (A4996) - Decrypt	256 bits	KAT	CAST	Return code zero (0) for success, non-zero for failure	Decrypt	Module boot
HMAC-SHA2-256 (A4999)	256 bits hash	KAT	CAST	Return code zero (0) for success, non-zero for failure	Message Digest	Module boot
HMAC-SHA2-512 (A4999)	512 bits hash	KAT	CAST	Return code zero (0) for success, non-zero for failure	Message Digest	Module boot
ECDSA SigGen (FIPS186-4) (A4997)	P-384 curve, SHA2-384 hash	KAT	CAST	Return code zero (0) for success, non-zero for failure	Signature Generation	Module boot
ECDSA SigVer (FIPS186-4) (A4998)	P-384 curve, SHA2-384 hash	KAT	CAST	Return code zero (0) for success,	Signature Verification	Module boot

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				non-zero for failure		
RSA SigVer (FIPS186-4) (A5000)	Signature Type: PSS, Modulo: 3072 bits, Hash: SHA2-512	KAT	CAST	Return code zero (0) for success, non-zero for failure	Signature Verification	Module boot
NIST SP800-90B Entropy Source Health Test - Repetition Count Test (RCT)	cutoff value = 41; false positive error rate = 2^-20	Repetition Count Test (RCT)	CAST	A non-maskable alarm raised, the cause of failure can be read from ALARMS register	N/A	Entropy Source Start Up and Continuously
NIST SP800-90B Entropy Source Health Test - Adaptive Proportion Test (APT)	cutoff value = 793; window size = 1024; false positive error rate = 2^-20	Adaptive Proportion Test (APT)	CAST	A non-maskable alarm raised, the cause of failure can be read from ALARMS register	N/A	Entropy Source Start Up and Continuously

Table 21: Conditional Self-Tests

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
RSA SigVer (FIPS186-4) (A5000)	KAT	SW/FW Integrity	On Demand	Automatically on boot

Table 22: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
KDA HKDF SP800-56Cr2 (A5861)	KAT	CAST	On demand	Manually, via reboot

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
KAS-ECC-SSC Sp800-56Ar3 (A5861)	KAT	CAST	On demand	Manually, via reboot
ECDSA KeyGen (FIPS186-5) (A5861)	PCT	PCT	On keypair generation	Programmatically
AES-CBC (A4995) - Encrypt	KAT	CAST	On demand	Manually, via reboot
AES-CBC (A4995) - Decrypt	KAT	CAST	On demand	Manually, via reboot
AES-GCM (A4995)	KAT	CAST	On demand	Manually, via reboot
KDF SP800-108 (A4995)	KAT	CAST	On demand	Manually, via reboot
AES-KW (A4996) - Encrypt	KAT	CAST	On demand	Manually, via reboot
AES-KW (A4996) - Decrypt	KAT	CAST	On demand	Manually, via reboot
HMAC-SHA2-256 (A4999)	KAT	CAST	On demand	Manually, via reboot
HMAC-SHA2-512 (A4999)	KAT	CAST	On demand	Manually, via reboot
ECDSA SigGen (FIPS186-4) (A4997)	KAT	CAST	On demand	Manually, via reboot
ECDSA SigVer (FIPS186-4) (A4998)	KAT	CAST	On demand	Manually, via reboot
RSA SigVer (FIPS186-4) (A5000)	KAT	CAST	On demand	Manually, via reboot
NIST SP800-90B Entropy Source Health Test - Repetition Count Test (RCT)	Repetition Count Test (RCT)	CAST	On demand	Manually, via reboot
NIST SP800-90B Entropy Source Health Test - Adaptive	Adaptive Proportion Test (APT)	CAST	On demand	Manually, via reboot

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
Proportion Test (APT)				

Table 23: Conditional Periodic Information

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Hard Error	The module enters the Hard Error state when the pre-operational self-test or any conditional self-test fails	Pre-operational self-test failure Conditional self-test failure	Module reload	Error indicator for HW self-test failure: 0x00020001 read from interrupt register Error indicator for FW self-test failure: CM_RC_HKDF512_SELF_TEST_ERROR, CM_RC_HKDF384_SELF_TEST_ERROR, non-zero error value for failure in the KAS-ECC-SSC self-test, Error indicator for firmware integrity test: SBB_STATUS_AUTH_FAIL(33554442) and INTEGRITY_SELF_TEST_ERROR (16777225)
Soft Error	The module enters the Soft Error state when the pairwise consistency test fails	Pairwise consistency test failure	The module discards the keypair and continues to be operational	A non-zero error code returned

Table 24: Error States

If any of the tests fail, the module will return an error code and transition to the corresponding error state. Upon reaching the Hard Error state, no cryptographic functions can be executed, and all data output is inhibited. An operator can attempt to reset the state by cycling the power. However, the failure of a self-test may require the module to be replaced. In the event of a transition to a Soft Error state, the module will return an error and resume operation.

10.5 Operator Initiation of Self-Tests

An operator can initiate self-tests on demand by power-cycling the Intel SoC i.e. by rebooting/resetting the module.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

No specific instructions are required for initialization of the module in the Approved mode apart from applying power to the Intel SoC.

11.2 Administrator Guidance

No additional guidance applies for the operation of the module apart from that specified in Sections 2, 3 of this document and other subsections under this section.

11.3 Non-Administrator Guidance

No additional guidance applies for the operation of the module apart from that specified in Sections 2, 3 of this document and other subsections under this section.

11.4 End of Life

The module can be zeroised to perform secure sanitization.

12 Mitigation of Other Attacks

12.1 Attack List

The module does not implement any attack mitigation techniques.