Crypto Module for Intel® Alder Point PCH Converged Security and Manageability Engine (CSME)
FIPS 140-3 Non-Proprietary Security Policy

Hardware Version 4.6.0.0

Firmware Version 5.2.0.0

Prepared by:

atsec information security corporation

4516 Seton Center Parkway, Suite 250

Austin, TX 78759

www.atsec.com

Table of Contents

Copyrights and Trademarks

1. General

This document is the non-proprietary FIPS 140-3 Security Policy of the Crypto Module for Intel® Alder Point PCH Converged Security and Manageability Engine (CSME). 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 for a Hybrid Firmware module at an overall security level 2. The table below shows the security level claimed for each of the security requirement area that comprise the FIPS 140-3 standard:

ISO/IEC 24759 Section 6 [Number Below]	FIPS 140-3 Section Title	Security Level
1	General	2
2	Cryptographic Module Specification	2
3	Cryptographic Module Interfaces	2
4	Roles, Services and Authentication	2
5	Software/Firmware security	2
6	Operational environment	N/A
7	Physical security	2
8	Non-invasive security	N/A
9	Sensitive security parameter management	2
10	Self-tests	2
11	Life-cycle assurance	2
12	Mitigation of other attacks	2
	Overall Level	2

Table 1 - Security Levels

2. Cryptographic Module Specification

2.1. Module Overview

The Crypto Module for Intel® Alder Point PCH Converged Security and Manageability Engine (CSME) is classified as a Hybrid Firmware module operating in a single-chip environment. In this document, “CSME”, and “module” are used interchangeably. They all refer to the Crypto Module for Intel® Alder Point PCH Converged Security and Manageability Engine (CSME).

The module consists of both hardware (AES, ECC, and HCU hardware cryptographic engines) and firmware providing interface to the engines.

2.2. Module Components

The components of the hybrid cryptographic module are specified in the following table:

Component	Type	Version Numbers	Description
Converged Security Management Engine (CSME) Driver	Firmware	5.2.0.0	Firmware running in CSME to interface with the hardware components of the module.
CSME ROM	Hardware	4.6.0.0	Non-modifiable code which initiates and bootstraps the CSME firmware.
Offload and Crypto Subsystem (OCS)	Hardware	4.6.0.0	AES, ECC, and HCU hardware cryptographic engines embedded within the Intel PCH chipset.
fips_hmac	File	N/A	file containing the integrity check value of the module.

Table 2 - Cryptographic Module Components

The module has been tested on the following single-chip platform:

Operating System	Hardware Platform	Processor	PAA/Acceleration
CSME OS running firmware version 16.1.25.2124	Alder Point PCH-S	Alder Lake S	None
CSME OS running firmware version 16.1.25.2124	Alder Point PCH-M/P	Alder Lake M	None
CSME OS running firmware version 16.1.25.2124	Alder Point PCH-S	Raptor Lake S	None
CSME OS running firmware version 16.1.25.2124	Alder Point PCH-S	Raptor Lake HX	None
CSME OS running firmware version 16.1.25.2124	Alder Point PCH-M/P	Raptor Lake P	None

Table 3 - Operational Environments

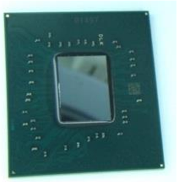

Figure 1: Alder Point PCH-S with Alder Lake-S

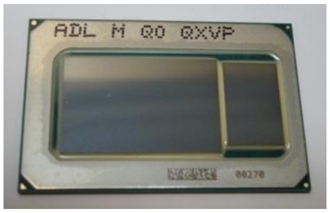

Figure 2: Alder Point PCH-M/P with Alder Lake M

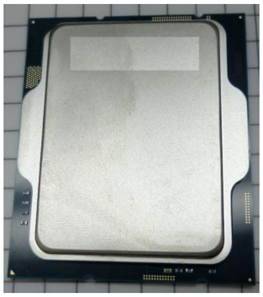

Figure 3: Alder Point PCH-S with Raptor Lake S / Raptor Lake HX[1]

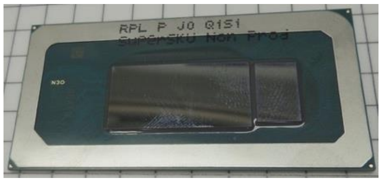

Figure 4: Alder Point PCH-M/P with Raptor Lake P

2.3. Cryptographic Boundary

The module is a firmware hybrid module implemented in the physical embodiment of either a Alder Point PCH-S with Alder Lake S (referred to as ADL-S), Raptor Lake S (RPL-S) or Raptor Lake HX (RPL-

[1] RPL-HX is the same exact HW as RPL-S but using a LGA socket.

HX) CPU or an Alder Point PCH-M/P with Alder Lake M (ADL-M) or Raptor Lake P (RPL-P) CPU. The Tested Operational Environment's Physical Perimeter (TOEPP) is represented by the dashed purple lines in the block diagram shown below.

The module provides cryptographic services to operators through an application program interface (API). The cryptographic boundary consists of the OCS ROM, CSME Driver FW, the AES, ECC, and HCU hardware cryptographic engines along with the fips_hmac integrity file. The cryptographic boundary is represented by the dashed red lines below.

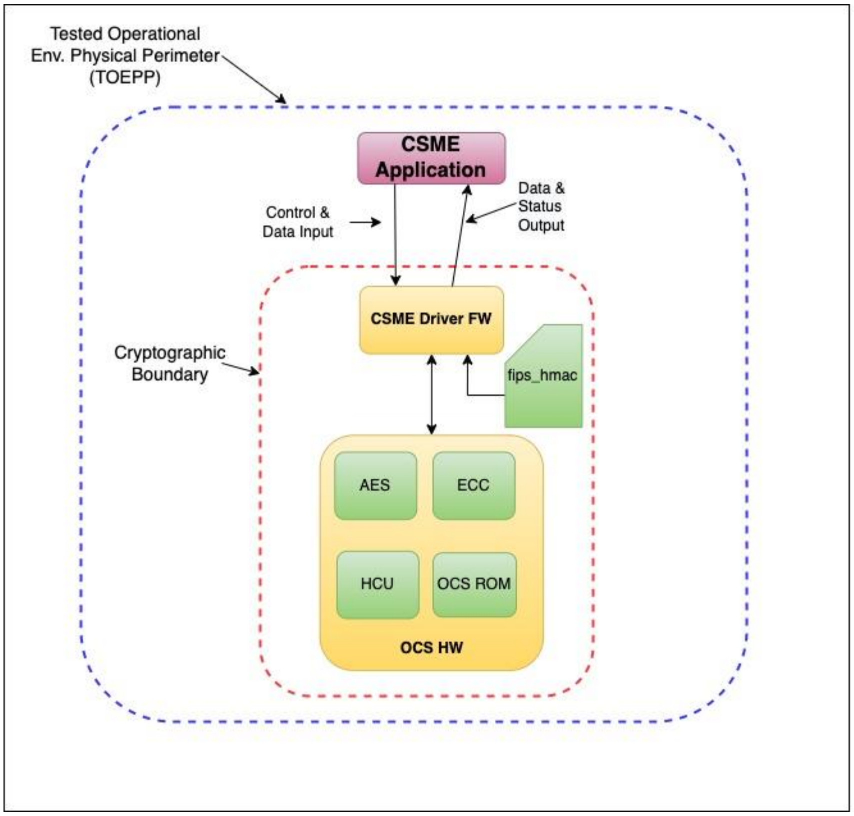

Figure 5 – Logical cryptographic boundary and physical boundary block diagram

2.4. Modes of operation

The module supports two modes of operation:

- In "Approved mode" (the Approved mode of operation) only approved or allowed security functions with sufficient security strength can be used. Table 4 lists the approved security functions.
- In "Non-approved mode" (the non-Approved mode of operation) only non-approved security functions can be used. Table 5 lists the non-FIPS functions.

The mode of operation is implicitly assumed contingent on the service that is being requested. In other words, if an approved service is requested, the module assumes the approved mode of operation; if a non-approved service is requested, the module assumes the non-approved mode of operation.

The module does not implement a degraded mode of operation.

2.5. Approved Algorithms

The module supports the following Approved cryptographic algorithms. Table 4 lists the algorithms validated by the CAVP Certificate:

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
A3362	AES [FIPS 197] [SP800-38A]	CBC, CFB128, CTR, ECB, OFB	Description: Data Encryption and Decryption using AES with CBC, CFB128, CTR, ECB and OFB modes Key Size(s): 128, 256 bits Strength: 128 and 256 bits	AES Encryption / AES Decryption
A3362	AES [FIPS 197] [SP800-38D]	GCM Internal IV (Mode 8.2.2)	Description: Authenticated Encryption and Decryption using AES with GCM mode and internally generated IV Key Size(s): 128, 256 bits Strength: 128 and 256 bits	AES Encryption / AES Decryption
A3362	AES [FIPS 197] [SP800-38B]	CMAC	Description: Message Authentication Code using AES with CMAC mode Key Size(s): 128, 256 bits Strength: 128 and 256 bits	AES Message Authentication Code Generation and Verification
A3362	HMAC [FIPS 198-1]	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512	Description: Message Authentication Code using HMAC with SHS Key Size(s): 112 bits or greater Strength: 112 bits or greater	HMAC Message Authentication Code Generation
A3362	CTR_DRBG [SP800-90Arev1]	AES-256	Description: No Prediction Resistance, With Derivation Function Enabled Key Size(s): 256 bits Strength: 256-bits	Random Number Generation
A3362	ECDSA KeyGen [FIPS 186-4] [ANSI X9.62]	Testing Candidates	Description: ECDSA Key Generation using Testing Candidates generation mode Key Size(s): Curves P-256, P-384 Strength: 128 or 192 bits	ECDSA key-pair Generation
A3362	ECDSA KeyVer [FIPS 186-4] [ANSI X9.62]	N/A	Description: ECDSA public key verification Key Size(s): Curves P-256, P-384 Strength: 128 or 192 bits	ECDSA public-key verification
A3362	ECDSA SigGen [FIPS 186-4] [ANSI X9.62]	N/A	Description: ECDSA Digital Signature Generation with SHA-256 or SHA-384 message digest Key Size(s): Curves P-256, P-384 Strength: 128 or 192 bits	ECDSA digital signature generation

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
A3362	ECDSA SigVer [FIPS 186-4] [ANSI X9.62]	N/A	Description: ECDSA Digital Signature Verification with SHA-256 or SHA-384 message digest Key Size(s): Curves P-256, P-384 Strength: 128 or 192 bits	ECDSA digital signature verification
A3362	KAS-ECC [SP 800 56Arev3]	Ephemeral Unified	Description: ECDH Key Agreement with Full Validation, Key Pair Generation; KAS Role: Initiator, Responder; KDF Methods: One Step KDF; Auxiliary Function Methods: SHA-224, SHA-256, SHA-384, SHA-512 Key Size(s): Curves P-256, P-384 Strength: 128 or 192 bits	ECDH key agreement
A3362	KBKDF [SP 800-108rev1]	Counter	Description: Key Based Key Derivation with Counter Length: 32-bits; MAC Mode: HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512; Fixed Data Order: Before Fixed Data Key Size(s): 112-bits or greater Strength: Min 112 bits	Key Derivation
A3362	KTS-RSA (KTS-IFC) [SP 800-56Brev2]	KTS-OAEP-basic	Description: Key Encapsulation using 2048-, 3072- or 4096-bit modulus with SHA-224, SHA-256, SHA-384 or SHA-512 message digest Key Sizes: 2048, 3072, 4096 bits Strength: 112 to 150 bits	RSA Key Transport (key encapsulation)
A3362	KTS-RSA (KTS-IFC) [SP 800-56Brev2]	KTS-OAEP-basic	Description: Key Un-encapsulation using 2048-bit modulus with SHA-224, SHA-256, SHA-384 or SHA-512 message digest Key Size(s): 2048[3] bits Strength: 112 bits	RSA Key Transport (key un-encapsulation)

[3] Although other sizes were validated by CAVP, only modulus size 2048 is considered approved for key un-encapsulation. See Section 6.3.1 for more details.

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
A3362	RSA KeyGen [FIPS 186-4]	B.3.3 Generation of Random Primes that are Probably Prime	Description: RSA Key Generation with 2048-bit modulus using random probable prime generation method Key Size(s): 2048 bits Strength: 112 bits	RSA key-pair generation
A3362	RSA SigGen [FIPS 186-4]	PKCS#1 v1.5 RSA-PSS	Description: RSA Digital Signature Generation using PKCS#1 v1.5 or RSA-PSS scheme with SHA-224, SHA-256, SHA-384 or SHA-512 message digest Key Size(s): 2048, 3072, 4096 bits Strength: 112 to 150 bits	RSA digital signature generation
A3362	RSA SigVer [FIPS 186-4]	PKCS#1 v1.5 RSA-PSS	Description: RSA Digital Signature Generation using PKCS#1 v1.5 or RSA-PSS scheme with 2048-, 3072- or 4096-bit modulus and SHA-1, SHA-224, SHA-256, SHA-384 or SHA-512 message digest Key Size(s): 1024, 2048, 3072, 4096 bits Strength: 80 to 150 bits	RSA digital signature verification
A3362 (CVL)	RSA (Signature Primitive) [FIPS 186-4]	PKCS#1 v1.5 RSA-PSS	Description: RSA Digital Signature Generation using PKCS#1 v1.5 or RSA-PSS scheme with 2048-bit modulus on a pre-hashed message digest Key Size(s): 2048 bits Strength: 112 bits	RSA digital signature generation primitive
A3362 (CVL)	RSA Decryption Primitive [SP 800-56Brev2]	N/A	Description: RSA Un-encapsulation Primitive decryption using 2048-bit modulus Key Size(s): 2048 bits Strength: 112 bits	RSA decryption primitive
A3362	SHS [FIPS 180-4]	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	Description: Message Digest Generation using SHA-1, SHA-224, SHA-256, SHA-384 or SHA-512 Key Size(s): N/A Strength: N/A	SHS message digest generation

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strength(s)	Use / Function
Vendor Affirmed	Cryptographic Key Generation (CKG) for asymmetric keys [SP800-133rev2]	SP800-133rev2 section 5.1 for ECDSA and RSA key pairs, SP800-133rev2 section 5.2 for ECDH	Description: Vendor Affirmed Asymmetric Key Generation for RSA and ECDSA Key Size(s): RSA: 2048 bits ECC: Curves P-256, P-384 Strength: RSA: 112 bits ECC: 128 or 192 bits	ECDSA key-pair generation / RSA key-pair generation
N/A	ENT (P) [SP800-90B]	N/A	Description: Random Number Generation from a Physical Entropy Source Key Size(s): N/A Strength: 256	Random Number Generation

Table 4 – Approved Algorithms

2.6. Non-Approved Algorithms

The module does not implement any Non-Approved Algorithms Allowed in the Approved Mode of Operation nor Non-Approved Algorithms Allowed in the Approved Mode of Operation with No Security Claimed. The module implements the following non-Approved algorithms **Not** Allowed in the Approved Mode of Operation listed in the table below.

Algorithm	Use/Function
MD5	Message Digest generation using MD5 algorithm
SM2	SM2 digital signature generation
	SM2 digital signature verification
SM3	SM3 message digest generation
SM4	SM4 data encryption and decryption
HMAC-MD5	HMAC-MD5 MAC generation
ECC Commit Computation	ECC commit computation used in ECSCHNORR and ECDAA Signature Generation
ECDAA	ECDAA digital signature generation
	ECDAA digital signature verification
ECSCHNORR	ECSCHNORR digital signature generation
	ECSCHNORR digital signature verification
AES-GCM	AES-GCM data encryption using an externally generated IV
HMAC	HMAC MAC generation using with keys less than 112 bits
ECDSA	ECDSA key-pair generation using secp256k1 curve
ECDHE Shared Secret Computation	ECDHE shared secret computation using brainpoolp384r1 curve

RSA	RSA key-pair generation using 1024 bits modulus size
	RSA digital signature generation using MD5, SM3 or SHA-1 hash algorithm
	RSA digital signature verification using MD5 or SM3 hash algorithm
RSA	RSA key encapsulation and un-encapsulation using MD5 or SHA-1
	RSA key encapsulation and un-encapsulation using 1024-bit modulus
	RSA key un-encapsulation using 3072 or 4096-bit modulus

Table 5 - Non-Approved Algorithms ***Not*** *Allowed in the Approved Mode of Operation*

3. Cryptographic Module Ports and Interfaces

The cryptographic module is defined as a Firmware-Hybrid module. The logical interfaces are the application program interface (API) through which operators request services from the module (i.e., CSME Crypto Driver). All data and control inputs to the module, and data and status outputs from the module are provided through the module's API (i.e., logical interface). The SRAM and power interfaces are provided by the computing platform on which it runs. The module does not implement a control output interface. The following table summarizes the four logical interfaces:

Physical Port	Logical Interface[4]	Data that passes over port/interface
SRAM	Data Input	All data (except control data entered via the control input interface) that is input to and processed by a cryptographic module
SRAM	Data Output	All data (except status data output via the status output interface and control data output via the control output interface) that is output from a cryptographic module
SRAM	Control Input	Control data used to control the operation of a cryptographic module
SRAM	Status Output	All status data used to indicate the status of a cryptographic module
Provided by the underlying SoC/PCH.	Power Input	external electrical power that is input to a cryptographic module

Table 6 - Ports and Interfaces

[4] Control Output Interface is not implemented in the module and thus omitted from this table.

4. Roles, Services and Authentication

4.1. Roles

The module supports the following roles:

- **User Role:** Performs all the services (in both approved mode and non-approved mode), except for the module installation and configuration.
- **Crypto Officer role:** performs module initialization (installation, configuration).

The module does not support a maintenance role The User and Crypto Officer roles are assumed by the entity accessing the module services contingent on the authentication, as described in Section 4.3. The Crypto Officer and User Roles are separated by function. The Crypto Officer is only available during initialization of the module as instructed in Section 11.1. After the module is operational, only the User Role is available to request services of the module.

Role	Service	Input	Output
User	AES data encryption	Key and Plaintext Data	Ciphertext Data
User	AES data decryption	Key and Ciphertext Data	Plaintext Data
User	AES message authentication code generation	Key and Message	Message Authentication Code
User	AES message authentication code verification	Key and Message Authentication Code	True or False
User	HMAC message authentication code generation	Key and Message	Message Authentication Code
User	RSA key pair generation	Key Size	RSA Key Pair
User	RSA public key validation	RSA Public Key	True or False
User	RSA digital signature generation	RSA Private Key and Message	Digital Signature
User	RSA digital signature verification	RSA Public Key and Digital Signature	True or False
User	RSA Key Transport (Key encapsulation)	RSA Public Key and Plaintext Key	Encrypted Key
User	RSA Key Transport (Key un-encapsulation using a key size of 2048 bits)	RSA Private Key and Encrypted Key	Plaintext Key
User	ECDSA key pair generation	EC curve	ECDSA Key Pair
User	ECDSA public key verification	ECDSA Public Key	True or False
User	ECDSA signature generation	ECDSA Private Key and Message	Digital Signature

User	ECDSA signature verification	ECDSA Public Key and Digital Signature	True or False
User	ECDH key agreement	EC Curve, Party U’s ephemeral private key, and Party V’s ephemeral public key	Derived Key
User	SHS Message digest generation	Message	Message Digest
User	Random Number Generation	Entropy input string and nonce	Random Numbers
User	Key Derivation	Key Derivation Key	Derived Key
User	Show Module Version	None	Module Base Name + Module Version Number
User	Show Status	None	Operational/Error status
User	On Demand Self-Test (Pre-operational Integrity Test and Conditional Algorithm Self-Test)	None	Pass/Fail status
User	Zeroization	None	None
Crypto Officer	Module Initialization	BIOS settings	None
User	(Non-Approved) Data Encryption using AES-GCM with externally generated IV	Key and Plaintext Data	Ciphertext Data
User	(Non-Approved) Data Encryption and Decryption using SM4	Key and Plaintext Data for Encryption; Key and Ciphertext Data for Decryption	Ciphertext Data for Encryption; Plaintext Data for Decryption
User	(Non-Approved) Message digests using MD5 or SM3	Message	Message Digest
User	(Non-Approved) MAC generation using HMAC-MD5	Key and Message	Message Authentication Code
User	(Non-Approved) MAC generation using less than 112 bits HMAC key	Key and Message	Message Authentication Code
User	(Non-Approved) Key generation using RSA with 1024 bits modulus size	Key Size	RSA Key Pair
User	(Non-Approved) Key generation using ECDSA using secp256k1 curve	EC curve	RSA Key Pair

User	(Non-Approved) Digital signature generation using RSA with MD5, SM3 or SHA-1 for any modulus size	Private Key and Message	Digital Signature
User	(Non-Approved) Digital signature verification using RSA with MD5, or SM3 for any modulus size	Public Key and Digital Signature	True or False
User	(Non-Approved) Digital signature generation using ECSCHNORR, ECDAA, or SM2 algorithm	Private Key and Message	Digital Signature
User	(Non-Approved) Digital signature verification using ECSCHNORR, ECDAA, or SM2 algorithm	Public Key and Digital Signature	True or False
User	(Non-Approved) Key Transport (key encapsulation and un-encapsulation) using RSA-OAEP with MD5 or SHA-1	Public Key and Plaintext Key for encapsulation; Private Key and Encrypted Key for un-encapsulation	Encrypted Key for encapsulation; Plaintext Key for un-encapsulation
User	(Non-Approved) Key Transport (key encapsulation and un-encapsulation) using RSA-OAEP with 1024-bit modulus	Public Key and Plaintext Key for encapsulation; Private Key and Encrypted Key for un-encapsulation	Encrypted Key for encapsulation; Plaintext Key for un-encapsulation
User	(Non-Approved) Key Transport (key un-encapsulation) using RSA-OAEP with 3072 or 4096-bit modulus	Private Key and Encrypted Key	Plaintext Key
User	(Non-Approved) Shared secret computation using ECDHE with brainpoolp384r1 curve	Party U’s ephemeral private key, and Party V’s ephemeral public key	Shared Secret
User	(Non-Approved) ECC Commit Computation using ECSCHNORR and ECDAA (used in ECSCHNORR and ECDAA Signature Generation)	EC Curve, Hash function, public and private keys	ECC Commit Computation

Table 7 - Roles, Service Commands, Input and Output

4.2. Services

The module provides services to the operator that assumes one of the authorized roles, User role after operator is authenticated or CO role which is not authenticated but can only perform initialization service. All services are described in detail in the user documentation. For Approved services, a fips_indicator flag is enabled in the crypto_ioctl_status_t structure and is set to FIPS_APPROVED_SEC_FUN when a function is an approved service. After the module completes the requested service, it will report the status as an output parameter indicating whether the service was Approved (i.e., set to "1").

The following table lists the Approved services and the non-Approved but allowed services in approved mode of operation, the roles that can perform the service, the Critical Security Parameters involved and how they are accessed:

The access rights to keys and SSPs have the following interpretation:

Generate: The module generates or derives the SSP.

Read: The SSP is read from the module (e.g., the SSP is output).

Write: The SSP is updated, imported, or written to the module.

Execute: The module uses the SSP in performing a cryptographic operation.

Zeroise: The module zeroises the SSP.

Service	Description	Approved Security Functions	Keys and/or SSPs	Role	Access rights to Keys and/or SSPs	Indicator
AES data encryption	Data Encryption using Advanced Encryption Standard algorithm	AES-CBC, AES-CFB128, AES-CTR, AES-ECB, AES-OFB, AES-GCM	AES keys	User	Write, Execute, Zeroise	1
AES data decryption	Data Decryption using Advanced Encryption Standard algorithm	AES-CBC, AES-CFB128, AES-CTR, AES-ECB, AES-OFB, AES-GCM	AES key	User	Write, Execute, Zeroise	1
AES message authentication code generation	Message Authentication Code Generation using Advanced Encryption Standard algorithm	AES-CMAC	AES key	User	Write, Execute, Zeroise	1

Service	Description	Approved Security Functions	Keys and/or SSPs	Role	Access rights to Keys and/or SSPs	Indicator
AES message authentication code verification	Message Authentication Code Verification using Advanced Encryption Standard algorithm	AES-CMAC	AES key	User	Write, Execute, Zeroise	1
HMAC message authentication code generation	Message Authentication Code Generation using Hashed Message Authentication Code algorithm	HMAC-SHA-1 HMAC-SHA-224 HMAC-SHA-256 HMAC-SHA-384 HMAC-SHA-512	HMAC key	User	Write, Execute, Zeroise	1
RSA key-pair generation	Asymmetric Key Pair Generation	RSA Key Generation, (CKG Vendor Affirmed), DRBG	Module Generated RSA public key / Module Generated RSA private key	User	Generate, Read	1
RSA public key validation	Public Key Validation of RSA public key	RSA Key Validation	Module Generated RSA public key/RSA public key	User	Write, Zeroize	1
RSA digital signature generation	Digital Signature Generation using RSA	PKCS#1 v1.5 RSASSA-PSS, DRBG, SHA	Module Generated RSA private key/RSA private key	User	Write, Execute, Zeroise	1
RSA digital signature generation primitive	Digital Signature Generation on a pre-hashed message using RSA	PKCS#1 v1.5 RSASSA-PSS, DRBG	Module Generated RSA private key/RSA private key	User	Write, Execute, Zeroise	1
RSA digital signature verification	Digital Signature Verification using RSA	PKCS#1 v1.5 RSASSA-PSS, SHA	Module Generated RSA public key/RSA public key	User	Write, Execute, Zeroise	1

Service	Description	Approved Security Functions	Keys and/or SSPs	Role	Access rights to Keys and/or SSPs	Indicator
RSA key transport (key encapsulation)	Plaintext Key Encryption (encapsulation) using KTS-RSA	KTS-OAEP-basic, DRBG	RSA public key	User	Write, Execute, Zeroise	1
RSA key transport (key un-encapsulation)	Encrypted Key Decryption (un-encapsulation) using KTS-RSA with a key size of 2048 bits	KTS-OAEP-basic	Module Generated RSA private key	User	Write, Execute, Zeroise	1[5]
RSA decryption primitive	RSA decryption primitive as defined in SP800-56BRev2 Section 7.1.2.1	RSADP	Module Generated RSA private key/RSA private key	User	Write, Execute, Zeroise	1
ECDSA key-pair generation	Asymmetric Key Pair Generation	ECDSA Key Generation (CKG Vendor Affirmed), DRBG	ECDSA public key, ECDSA private key	User	Generate, Read	1
ECDSA public key validation	Public Key Validation of ECDSA public key	ECDSA Public Key Validation	ECDSA public key		Write, Zeroise	1
ECDSA digital signature generation	Digital Signature Generation using ECDSA	ECDSA Digital Signature Generation, DRBG, SHA	ECDSA private key		Write, Execute, Zeroise	1
ECDSA digital signature verification	Digital Signature Verification using ECDSA	ECDSA Digital Signature Verification, SHA	ECDSA public key		Write, Execute, Zeroise	1
ECDH key agreement	Diffie-Hellman Key Agreement using Elliptic Curve	KAS-ECC	ECDH private key and remote public key	User	Write, Execute,	1
			shared secret		Generate	

[5] RSA key pair generation service shall also return “1”.

Service	Description	Approved Security Functions	Keys and/or SSPs	Role	Access rights to Keys and/or SSPs	Indicator
	Cryptography		SP 800-56C KDF derived key		Generate, Read	
SHS message digest generation	Message Digest Generation using Secure Hash Standard algorithm	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512	None	User	None	1
Random Number Generation	Deterministic Random Number Generation	CTR_DRBG	Entropy Input String	User	Write, Execute	1
			DRBG Seed		Write, Read, Zeroize	
			DRBG internal states (V and Key)		Write, Read, Zeroize	
Key Derivation	Derive key using KBKDF in counter mode	KBKDF	KBKDF key derivation key	User	Write, Execute, Zeroise	1
			KBKDF derived key		Generate, Execute	
Show Module Version	Output Module Name and Module Hardware and Firmware Version Numbers	None	None	User	None	N/A
Show Status	Outputs Operational / Error Status of the Module	None	None	User	None	N/A
On Demand Self-Test (Pre-operational Integrity Test and Conditional Algorithm Self-Test)	Performs On Demand Self-Test	See Section 10	None	User	None	N/A
Zeroization	Zeroizes all CSPs	See Section 9.6	All CSPs	User	Zeroize	N/A

Service	Description	Approved Security Functions	Keys and/or SSPs	Role	Access rights to Keys and/or SSPs	Indicator
Module Initialization	Configure BIOS to Initialize in FIPS Validated Configuration	None	None	Crypto Officer	None	N/A

Table 8 – Approved Services

The following table lists the services only available in non-approved mode of operation:

Service	Description	Algorithms Accessed	Role	Indicator
Data Encryption using AES-GCM with externally generated IV	Data Encryption using AES-GCM with externally generated IV	AES-GCM	User	N/A
Data Encryption and Decryption using SM4	Data Encryption and Decryption using SM4	SM4	User	N/A
Message digests using MD5 or SM3	Message digests using MD5 or SM3	MD5 or SM3	User	N/A
MAC generation using HMAC-MD5	MAC generation using HMAC-MD5	HMAC-MD5	User	N/A
MAC generation using less than 112 bits HMAC key	MAC generation using less than 112 bits HMAC key	HMAC	User	N/A
Key generation using RSA with 1024 bits modulus size	Key generation using RSA with 1024 bits modulus size	RSA	User	N/A
Key generation using ECDSA using secp256k1 curve	Key generation using ECDSA using secp256k1 curve	ECDSA	User	N/A
Digital signature generation using RSA with MD5, SM3 or SHA-1 for any modulus size	Digital signature generation using RSA with MD5, SM3 or SHA-1 for any modulus size	RSA with MD5, SM3 or SHA-1	User	N/A
Digital signature verification using RSA with MD5, or SM3 for any modulus size	Digital signature verification using RSA with MD5, or SM3 for any modulus size	RSA with MD5, or SM3	User	N/A
Digital signature generation using ECSCHNORR, ECDAA, or SM2 algorithm	Digital signature generation using ECSCHNORR, ECDAA, or SM2 algorithm	ECSCHNORR, ECDAA, or SM2	User	N/A
Digital signature verification using ECSCHNORR, ECDAA, or SM2 algorithm	Digital signature verification using ECSCHNORR, ECDAA, or SM2 algorithm	ECSCHNORR, ECDAA, or SM2	User	N/A
Key Transport (key encapsulation and un-encapsulation) using RSA-OAEP with MD5 or SHA-1	Key Transport (key encapsulation and un-encapsulation) using RSA-OAEP with MD5 or SHA-1	RSA, MD5, SHA-1	User	N/A

Service	Description	Algorithms Accessed	Role	Indicator
Key Transport (key encapsulation and un-encapsulation) using RSA-OAEP with 1024-bit modulus	Key Transport (key encapsulation and un-encapsulation) using RSA-OAEP with 1024-bit modulus	RSA	User	N/A
Key Transport (key un-encapsulation) using RSA-OAEP with 3072 or 4096-bit modulus	Key Transport (key un-encapsulation) using RSA-OAEP with 3072 or 4096-bit modulus	RSA	User	N/A
Shared secret computation using ECDHE with brainpoolp384r1 curve	Shared secret computation using ECDHE with brainpoolp384r1 curve	KAS-ECC	User	N/A
ECC Commit Computation using ECSCHNORR and ECDAA (used in ECSCHNORR and ECDAA Signature Generation)	ECC Commit Computation using ECSCHNORR and ECDAA (used in ECSCHNORR and ECDAA Signature Generation)	ECSCHNORR and ECDAA	User	N/A

Table 9 – Non-Approved Services

4.3. Operator Authentication

The module implements role-based operator authentication to authenticate the User Role. The authentication mechanism is based on the RSASSA-PSS signature algorithm with 3072-bit modulus (Cert. #A3362). The Crypto Officer role is not authenticated. The Crypto Officer role can only perform the initialization service, which is a non-authenticated service and does not affect the security of the module per IG 4.1.A.

An operator with the User role is the CSME firmware application (Figure 5). There can be only one CSME application running and interfacing with the module during the module's operation, enforced by the design of the module. Thus, the module does not support concurrent operators.

The manufacturer utilizes their unique RSA private key to sign the CSME application images. The private signing keys are kept in a HSM (Hardware Security Module) within an Intel secured facility and remain unknown to both the module and the CSME application. The public key and the signature are provided as part of the firmware manifest. The hash of the public key is also hardcoded in the module. During runtime, the public key in the provided firmware manifest is first hashed, then compared with the hash stored in module. If the hash matches, the module proceeds to assert whether the signature of the CSME application verifies, using the public key. If the signature verification succeeds, the CSME application firmware is authenticated and hence can be loaded and executed. The module does not maintain authentication after computing platform power loss (power-off, reset, etc.).

The authentication process occurs within the physical perimeter of the module, and thus it is not visible outside of this perimeter. The authentication data is essentially composed of public data (signature and public key); therefore, their disclosure does not affect the security of the authentication mechanism.

4.3.1. Strength of Authentication

The digital signature verification authentication mechanism using RSA PSS with 3072-bit modulus provides an encryption strength of 128 bits. The strength of this mechanism is equivalent to the probability of correctly guessing the private signing key, and this probability is $1/2^{128}$ (or 2.94e-39).

If attempts are made to authenticate an operator by guessing the private key and presenting the corresponding signature of the CSME firmware, we may suppose a rate of 1µs per attempted authentication (i.e., per guess of the private key and respective signature). This rate would allow 60,000,000 consecutive attempts per minute. The probability of successfully authenticating at this rate is less than or equal to 60,000,000 * $1/2^{128}$ (≤1.7632e-31).

Role	Authentication Method	Authentication Strength
Crypto Officer	N/A (IG 4.1.A)	N/A
User	Role-based	128-bits

Table 10 - Roles and Authentication

5. Software/Firmware Security

5.1. Integrity Techniques

A firmware integrity test is performed on the runtime image of the module. The HMAC-SHA256 implemented in the module is used as an approved algorithm for the integrity test. If the test fails, the module enters an error state where no cryptographic services are provided, and data output is prohibited i.e., the module is not operational.

The OCS ROM component of the module is a non-reconfigurable memory (specifically masked ROM), which is exempt from the requirements of integrity test. The vendor performed memory degradation testing to assert that the memory will not degrade before 10 (ten) years of manufacture date, thus complying with the requirements of IG 5.A.

5.2. On-Demand Integrity Test

The on-demand integrity self-tests can be invoked by the user performing reboot, the device which will cause pre-operational and conditional self-tests to run.

5.3. Executable Code

The Converged Security Management Engine (CSME) Driver i.e., module’s firmware, is made up of a single component, provided in the form of binary executable code. The firmware wholly contains the executable form without further compilation.

6. Operational Environment

6.1. Applicability

The module operates in a non-modifiable operational environment per FIPS 140-3 security level 2 specifications. The operator cannot modify the firmware component of the module. The module runs on an internal customized proprietary OS (i.e., CSME OS) within the Intel SoC.

7. Physical Security

The module is a hybrid firmware module that operates on a single-chip standalone platform which conforms to the Level 2 requirements for physical security. The single chip cryptographic module is a production grade component that include standard passivation (e.g., a sealing coat applied over the chip circuitry to protect it against environmental and other physical damage). The layering process which is used to embed the die into the PCB of the single chip computing platform also provides opacity that prevents viewing internal construction within the visible spectrum. The single chip enclosure prevents accessing of the module’s hardware components without leaving physical tamper evidence.

8. Non-invasive Security

This module does not implement any non-invasive security mechanism, and therefore this section is not applicable.

9. Sensitive Security Parameter Management

Keys residing in internal storage can only be accessed using the defined API. Memory and process space is protected from unauthorized access by the operating system. Only the process that creates or imports keys can use or export them.

According to FIPS 140-3, Sensitive Security Parameters (SSPs) consist of Critical Security Parameters (CSPs) and Public Security Parameters (PSPs). The following table summarizes all CSPs, and PSPs employed by the cryptographic module:

SSP Name	Strength	Security Function Cert Number	Generation	Import / Export	Establishment	Storage	Zeroization	Use & related keys
AES keys	128-bits and 256-bits	AES-ECB AES-CMAC AES-CBC, AES-CFB128, AES-CTR, AES-GCM AES-OFB A3362	N/A	Input: API input parameters Output: N/A, the module does not output any AES keys; MD/EE	N/A	Stored as plaintext in the RAM.	Keys in RAM are automatically zeroized at the conclusion of every service or by power-cycling (e.g., reset, power-off).	**Use**: Symmetric encryption and decryption; Message authentication code (MAC) generation and verification **Related keys**: N/A
HMAC keys	Minimum of 112-bits	HMAC A3362	N/A	Input: API input parameters Output: N/A, the module does not output any HMAC keys; MD/EE	N/A	Stored as plaintext in the RAM.	Keys in RAM are automatically zeroized at the conclusion of every service or by power-cycling (e.g., reset, power-off).	**Use**: Message authentication code (MAC) generation and verification **Related keys**: N/A
Module generated RSA public key (Including intermediate keygen values)	112-bits	RSA CTR_DRBG A3362	The RSA public key can be generated using FIPS 186-4 RSA Key Generation method. The prime number used in	Input: N/A Output: API output parameters; MD/EE	N/A	Stored as plaintext in the RAM.	Keys in RAM are automatically zeroized at the conclusion of every service or by power-cycling (e.g., reset, power-off).	**Use**: RSA key generation, RSA public key validation **Related keys**: DRBG internal state, Module generated RSA private key

SSP Name	Strength	Security Function Cert Number	Generation	Import / Export	Establishment	Storage	Zeroization	Use & related keys
			the RSA Key Generation is generated using SP 800-90Arev1 DRBG.					(Including intermediate keygen values)
Module generated RSA private key (Including intermediate keygen values)	112-bits	RSA CTR_DRBG A3362	The RSA public key can be generated using FIPS 186-4 RSA Key Generation method. The prime number used in the RSA Key Generation is generated using SP 800-90Arev1 DRBG.	Input: N/A Output: N/A	N/A	Stored as plaintext in the RAM.	Keys in RAM are automatically zeroized at the conclusion of every service or by power-cycling (e.g., reset, power-off).	**Use**:RSA key generation, RSA digital signature generation, RSA Key Transport, RSA Decryption Primitive **Related keys**: DRBG internal state, Module generated RSA public key (Including intermediate keygen values)
RSA public key (including intermediate keygen values)	80-bits[6], 112-bits, 128-bits and 150-bits	RSA A3362	N/A	Input: API input parameters Output: N/A; MD/EE	N/A	Stored as plaintext in the RAM.	Keys in RAM are automatically zeroized at the conclusion of every service or by power-cycling (e.g., reset, power-off).	**Use**: RSA digital signature verification, RSA Key Transport, RSA public key validation. **Related keys**: RSA private key (including intermediate keygen values)

[6] Only for RSA Signature Verification.

SSP Name	Strength	Security Function Cert Number	Generation	Import / Export	Establishment	Storage	Zeroization	Use & related keys
RSA private key (including intermediate keygen values)	112-bits, 128-bits and 150-bits	RSA A3362	N/A	Input: API input parameters Output: N/A; MD/EE	N/A	Stored as plaintext in the RAM.	Keys in RAM are automatically zeroized at the conclusion of every service or by power-cycling (e.g., reset, power-off).	**Use**: RSA digital signature generation **Related keys**: RSA public key (including intermediate keygen values)
ECDSA/ ECDH (ECC) public key (including inter-mediate keygen values)	128-bits and 192-bits	ECDSA, KAS-ECC-SSC CTR_DRBG A3362	The ECC public key can be generated using FIPS 186-4 ECDSA Key Generation method and the random value used in key generation is generated using SP 800-90Arev1 DRBG.	Input: API input parameters Output: API output parameters; MD/EE	N/A	Stored as plaintext in the RAM.	Keys in RAM are automatically zeroized at the conclusion of every service or by power-cycling (e.g., reset, power-off).	**Use**: ECDSA digital signature verification, ECDSA key generation, ECDSA public key validation, Shared secret computation **Related SSPs**: DRBG internal state, EC Diffie-Hellman Shared Secret, ECDSA/ ECDH (ECC) private key (including inter-mediate keygen values)
ECDSA/ ECDH (ECC) private key (including inter-mediate	128-bits and 192-bits	ECDSA, KAS-ECC-SSC CTR_DRBG A3362	The ECC private key can be generated using FIPS 186-4	Input: API input parameters Output: API output parameters; MD/EE	N/A	Stored as plaintext in the RAM.	Keys in RAM are automatically zeroized at the conclusion of every	**Use**: ECDSA digital signature generation, ECDSA key generation

SSP Name	Strength	Security Function Cert Number	Generation	Import / Export	Establishment	Storage	Zeroization	Use & related keys
keygen values)			ECDSA Key Generation method and the random value used in key generation is generated using SP 800-90Arev1 DRBG.				service or by power-cycling (e.g., reset, power-off).	, Shared secret computation **Related SSPs**: DRBG internal state, EC Diffie-Hellman Shared Secret, ECDSA/ECDH (ECC) public key (including inter-mediate keygen values)
Shared Secret	128-bits and 192-bits	KAS-ECC-SSC A3362	N/A	Input: N/A Output: N/A	The shared secret is generated in the EC Diffie-Hellman key agreement function.	Stored as plaintext in the RAM.	Keys in RAM are automatically zeroized at the conclusion of every service or by power-cycling (e.g., reset, power-off).	**Use**: Shared Secret Computation **Related SSPs**: ECDSA/ECDH (ECC) public key (including inter-mediate keygen values), ECDSA/ECDH (ECC) private key (including inter-mediate keygen values), SP 800-56C KDF derived key
SP 800-56C KDF derived key	Contingent on key derivation key	Key Agreement Key Derivation A3362	Derived using 800-56C KDF algorith	Input: N/A Output: API output	N/A	Stored as plaintext in		**Use**: Key Derivation **Related SSPs**:

SSP Name	Strength	Security Function Cert Number	Generation	Import / Export	Establishment	Storage	Zeroization	Use & related keys
	and length of output		m in the EC Diffie-Hellman key agreement function	parameters; MD/EE		the RAM.		Shared Secret
KBKDF key derivation key	Contingent on key derivation key and length of output	Key-Based Key Derivation A3362	N/A	Input: API input parameters Output: N/A; MD/EE	N/A	Stored as plaintext in the RAM.	Keys in RAM are automatically zeroized at the conclusion of every service or by power-cycling (e.g., reset, power-off).	**Use**: Key Derivation **Related SSPs**: KBKDF derived key
KBKDF derived key	Contingent on key derivation key and length of output	Key-Based Key Derivation A3362	Derived using KBKDF algorithm	Input: N/A Output: API output parameters; MD/EE	N/A	Stored as plaintext in the RAM.		**Use**: Key Derivation **Related SSPs**: KBKDF key derivation key
Entropy Input String	256-bits	CTR_DRBG A3362	N/A	Input: Obtained from the hardware ENT (P) outside of the cryptographic boundary but within its TOEPP Output: N/A	N/A	Stored as plaintext in the RAM.	Zeroized during the power cycle of the module.	**Use**: Random number generation **Related SSPs**: Entropy Input, DRBG Seed
DRBG Seed	256-bits	CTR_DRBG A3362	Derived from the entropy string as defined by SP 800-90ARev1	Input: N/A Output: N/A	N/A	Stored as plaintext in the RAM.	Zeroized during the power cycle of the module.	**Use**: Random number generation **Related SSPs**: Entropy input, DRBG internal state: (V and Key values)

SSP Name	Strength	Security Function Cert Number	Generation	Import / Export	Establishment	Storage	Zeroization	Use & related keys
DRBG internal state: (V and Key values)	256-bits	CTR_DRBG A3362	Generated internally in the DRBG.	Input: N/A Output: N/A	N/A	Stored as plaintext in the RAM.	Zeroized during the power cycle of the module.	**Use**: Random number generation **Related SSPs**: DRBG Seed

Table 11 - Life cycle of Sensitive Security Parameters (SSP)

9.1. Random Bit Generation

The module provides a SP800-90Arev1 compliant CTR_DRBG (Cert. #A3362) with AES-256 with derivation function and without prediction resistance as the approved Random Number Generator. The CTR_DRBG is implemented in the firmware (i.e., CSME Crypto Driver) and provides between 128 and 65536 bits of output data per each request.

In accordance with FIPS 140-3 IG D.L, the 'Entropy input string', 'DRBG Seed', 'DRBG internal state (V and key values)' are considered CSPs by the module.

Non-DRBG functions cannot access the DRBG internal state.

The module uses the output of the SP 800-90B and IG D.J compliant ENT (P) as the entropy source for seeding the CTR_DRBG inside the module. The entropy source provides a 128-bit output. At the output of the entropy source, the available entropy is 128 bits per 128 bits, thus full entropy. The module collects 384 bits of entropy input from the entropy source to seed the DRBG providing at least 256 bits of entropy.

Entropy Source	Minimum number of bits of entropy	Details
SP800-90B ENT (P) Physical	128-bits per 128-bits	Hardware entropy source with CBC-MAC conditioning component (Cert. #A2542) located within the tested execution environment’s physical perimeter.

Table 12 - Non-Deterministic Random Number Generation (Entropy Source) Specification

9.2. SSP Generation

The module implements asymmetric key generation services for RSA[7], ECDSA and EC Diffie-Hellman keys, compliant to SP800-133rev2 Cryptographic Key Generation (CKG, vendor affirmed) in accordance with IG D.H. For generating RSA and ECDSA keys, a seed (i.e., random value) used in asymmetric key generation obtained directly from the module’s approved SP800-90Arev1 DRBG (i.e., CTR_DRBG, Cert. #A3362). This follows asymmetric key generation method compliant with FIPS186-4 and defined in Section 5.1 of SP 800-133rev2 . The EC Diffie-Hellman keys are generated internally by the module using the ECDSA key generation method compliant with FIPS186-4 and SP800-56Arev3 as defined in section 5.2 of SP800-133rev2.

The module does not offer a dedicated service for generating symmetric keys.

[7] Approved RSA Key Generation will only use public key exponent e = 2^{16} + 1

9.3. SSP Establishment

The module provides an approved [SP800-56Arev3] EC Diffie-Hellman Key Agreement Scheme. The key agreement scheme is compliant with IG D.F scenario 2 path (2). The CAVP testing was performed end-to-end, using the Ephemeral Unified Model with approved domain parameters (i.e., P-256 and P-384 curves and SHA-224, SHA-256, SHA-384, and SHA-512 auxiliary function) resulting in a KAS-ECC Cert. #A3362.

The module provides key derivation service using SP800-108 KBKDF.

The module supports SP800-56Brev2 Key Transport using KTS-OAEP-basic[8]. RSA-KTS encapsulation is approved with the key sizes of 2048, 3072 and 4096 bits in approved mode while RSA-KTS un-encapsulation is approved only with a key size of 2048 bits in the approved mode.

9.3.1. Assurances

The following statements explain the SP800-56Brev2 assurances found in its Section 5:

- Section 5.1 – The module uses an approved hash function (SHS, Cert. #A3362) for mask generation during RSA-OEAP encryption.
- Section 5.2 – N/A, the module does not implement key confirmation.
- Section 5.3 – The module uses an approved random bit generator (CTR_DRBG, Cert. #A3362) when generating random values.
- Section 5.4 and Section 5.5 – N/A, the module does not implement a key agreement scheme (i.e., KAS1).
- Section 5.6 – N/A, the module does not implement key confirmation.

In addition, the following statements explain the SP800-56Brev2 assurances found in its Section 6 (specifically SP800-56Brev2 *Section 6.4 Required Assurances*):

- Prior to the use of the key pair in a key-establishment transaction, assurances required by the key-pair owner are obtained in the following way (Note: only keys generated following this guidance shall be compliant to SP800-56Brev2):
  1) The entity requesting the RSA key unwrapping (decapsulation)service from the module, shall only use an RSA private key that was generated by an active FIPS validated module that implements FIPS 186-4 compliant RSA key generation service and performs the key pair validity and the pairwise consistency as stated in section 6.4.1.1 of the SP 800-56BRev2. Additionally the entity shall renew these assurances over time by using any method described in section 6.4.1.5 of the SP 800-56BRev2.
  2) For use of an RSA key wrapping (encapsulation) service in the context of key transport per IG D.G,
     - the entity using the module, shall verify the validity of the peer's public key using the public key validation service of the module.
     - the entity using the module, shall confirm the peer's possession of private key by using any method specified in section 6.4.2.3 of the SP 800-56BRev2.

     Only after the above assurances are successfully met, shall the entity use the peer's public key to perform the RSA key wrapping (encapsulation) service of the module.

CAVEAT:

- EC Diffie-Hellman key establishment methodology provides 128 or 192 bits of encryption strength.
- RSA key wrapping provides between 112 and 150 bits of encryption strength.

[8] KTS-OAEP-basic is the basic scheme without key confirmation defined in section 9.2.3 of SP800-56Brev2.

9.4. SSP Entry / Output

The module does not support manual key entry or intermediate key generation key output. SSPs entered into the module are electronically entered in plain text form. SSPs are output from the module in plain text form if required by the calling application.

9.5. SSP Storage

The symmetric keys and HMAC keys are provided to the module via API input parameters and are zeroized by the module before they are released in the memory. Asymmetric public and private keys are provided to the module via API input parameters and are destroyed by the module before they are released in the memory.

9.6. SSP Zeroization

The memory occupied by SSPs is stored in RAM during runtime, allocated by regular memory allocation operating system calls. Every service of the module performs a zeroization operation as the last step before exiting from the function. The zeroization operation overwrites the memory occupied by keys with “zeros” before de-allocating the memory with the regular memory de-allocation operating system call. In addition, the RAM is volatile so zeroization of all SSPs can be performed by power-cycling (i.e., reset) or power-off the computing platform.

Additionally, while zeroization is in progress, data output is inhibited. Once a SSP is zeroized, it is no longer retrievable. The module uses an implicit indicator to express the completion of the zeroization operation. Zeroization is performed by the module through the course of executing its services. The module implicitly indicates the success of the zeroization by accepting the proceeding request. If the module accepts the next service request, this implicitly indicates that the zeroization of the previous service request successfully completed.

Lastly, temporary SSPs are zeroized when they are no longer needed.

10. Self-Tests

The module performs pre-operational firmware integrity test and conditional algorithm self-tests to ensure the correctness of the cryptographic algorithm implementations within the module boundary. The pre-operational and conditional cryptographic algorithm self-tests (CAST) are performed automatically at power-up or reset without any user interaction and must successfully pass all tests prior to providing any services to the caller. While the module is performing self-test, all access through data input, data output, control input and status output are inhibited. The module executes functions sequentially. While the self-tests are executing, no interaction is possible. The module does not implement a Software/Firmware Load Test, Manual Entry Test, Conditional Bypass Test nor Conditional Critical Functions Test, as the related functions are not implemented.

If any test fails, the module reports an error message through the status interface and enters the Error State. No data output and cryptographic operation are performed while the module is in the Error State. To recover from the Error State, the module must transition to the power off state, then to the power on state by a power cycle and must successfully pass both pre-operational integrity test and conditional algorithm self-tests. That is to say, when an error condition is detected and the error state is entered, all data output via the data output interface is inhibited, until error recovery occurs.

10.1.Pre-operational Firmware Integrity Test

The module performs pre-operational firmware integrity tests automatically when the computing platform is powered on, and the module is loaded into memory. The module’s OCS ROM first performs an HMAC-SHA-256 conditional cryptographic algorithm self-test (CAST) and after successfully passing, the module performs an integrity test of the module’s firmware component (Converged Security Management Engine (CSME) Driver) by computing an HMAC-SHA-256 value of the binary and comparing it with the value stored in the module that was computed at build time. If the HMAC values do not match, the test fails, and the module enters the Error state. While the module is performing pre-operational firmware integrity test, the module’s data output is inhibited.

10.2.Conditional Cryptographic Algorithm Self-Tests

The module performs a conditional cryptographic algorithm self-test (CAST) on all FIPS-Approved cryptographic algorithms supported in the approved mode of operation and per IG 10.3.A. The conditional cryptographic algorithm self-tests are performed before the first use of the related algorithm: First the AES ECB, HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512,and KAS-SSC are self-tested in hardware, prior to the integrity test of the firmware component, and then the rest of the CASTs are automatically performed after the integrity test completes successfully (and before the module enters the operational state[9]). If any of the cryptographic algorithm self-test fails, the module will enter the Error State, wherein all data output is inhibited. The pre-operational and conditional algorithm tests performed are shown in the following table:

Algorithm	Conditional CAST Performed
AES	• AES-ECB Encryption KAT • AES-ECB Decryption KAT • AES-CMAC Generation (Encryption) KAT
HMAC	• HMAC-SHA-1 KAT • HMAC-SHA-256 KAT • HMAC-SHA-512 KAT

[9] Operational State is the state where the operator can request services of the module.

Algorithm	Conditional CAST Performed
SHS	• SHA-1 KAT • SHA-256 KAT • SHA-512 KAT
ECDSA	• ECDSA Signature Generation KAT using P-256 curve and SHA-256 message d • Signature Verification KAT using P-256 curve and SHA-256 message digest
DRBG	• CTR_DRBG KAT
RSA	• RSA PKCS#1 v1.5 Signature Generation KAT with 2048 modulus and SHA-256 message digest • RSA PKCS#1 v1.5 Signature Verification KAT with 2048 modulus and SHA-256 message digest
	• KTS-OAEP Encryption KAT with 2048 modulus • KTS-OAEP Decryption KAT with 2048 modulus
SP800-108rev1 KDF	• KBKDF KAT using HMAC-SHA-384
KAS-SSC	• KAS-ECC Shared Secret Computation KAT using P-256 curve
	• One-Step KDF KAT using HMAC-SHA-384
OHT	• Health Test for ENT (P)

Table 13 - Conditional Cryptographic Algorithm Self-Tests

10.2.1. Entropy Related Health Tests

Intel has designed a proprietary Online Health Test (OHT) for the ENT (P) that can detect when:

1. Some value is consecutively repeated more times than expected, given the assessed entropy per sample of the source.
2. Some value becomes much more common in the sequence of noise source outputs than expected, given the assessed entropy per sample of the source.

After each reset, the OHT is automatically started and runs continuously until power-off or the next reset. A constant stream of 256-bit samples is outputted from the noise source into the OHT where it tracks the entropy health.

The OHT is designed to have the same functionality and health coverage as the following health tests required by NIST SP 800-90B:

- Start-Up Tests
- Repetitive Counter Test (RCT)
- Adaptive Proportion Test (APT)

The OHT was designed to detect repeating patterns from the ENT (P). The OHT accomplishes this by continuously monitoring the statistical arrival rate of short bit patterns. As the bit patterns arrive, they are counted and then tested to see if the total pattern number lay within the expected binomial distribution.

10.2.2. Conditional Pair-wise Consistency Tests

The module performs a Conditional Pair-wise Consistency Tests upon generating RSA and ECDSA/ECDH asymmetric key pairs. The test is implemented by calculating a signature on a predetermined data and subsequently performing a verification of the signature. If the signature cannot be verified, the generated key-pair is discarded, and the module enters the Error State.

10.3.On-Demand and Periodic Self-Tests

The on-demand and periodic self-tests can be invoked by the user performing reboot, the device which will cause pre-operational and conditional self-tests to run.

10.4.Error State

The module implements one error state. If any of the self-tests described in sections above fails, the module indicates the error indicator associated with the specific error by invoking the crypto_fips_error_handler() function and causes the module to enter the error state. In the error state, no cryptographic services are provided, and data output is prohibited. When the module is in the error state, the only method to recover is to reset the computing platform which results in the module reperforming the pre-operational firmware integrity test and the conditional cryptographic algorithm self-tests. The module will only enter the operational state after successfully passing both.

11. Life-cycle Assurance

11.1. Operator’s Guidance

The following security guidance for Crypto Officer role is described below:

- To enable the module for use in a FIPS validated configuration, the Crypto Officer must first perform initialization of the module. If FIPS operations are not enabled within the BIOS settings, then the module is not a 140-3 validated module and cannot enter the approved mode which means no FIPS services will be available. The Crypto Officer shall perform the following steps to initialize the module.
  - Power on the Host Platform and enter the BIOS menu setting.
  - Enter “FIPS mode” submenu.
  - Set “FIPS Mode Select” to <Enabled>.
  - Save and exit the BIOS menu.

The Host Platform will the power-cycle (i.e., reset) and proceed to boot. Once “FIPS Mode Select” is Enabled, the CSME OS will set the crypto_fips_en file which serves the control input the module and initializing it as a FIPS 140-3 validated module following power-on.

The following security guidance for User role is described below:

- The User of the module can call the API function crypto_drv_fips_mode_status() to check if the module is an FIPS 140-3 validated module. If the call returns 1, the module is an FIPS 140-3 validated module; it returns 0 if the module is not an FIPS 140-3 validated module. Note, this is not the service indicator; the service indicator is provided as described in Section 4.2. Services.

11.2. Delivery Procedure

The firmware component of the module is distributed as part of the CSME Device Driver firmware. Firmware is released on VIP site https://platformsw.intel.com. Only Original Equipment Manufacturers (OEM) with signed Intel agreements can download this firmware.

The module is contained within one of the platforms listed in Table 2. These Intel platforms are a tightly coupled component of 12th Generation Intel® Core™ chipsets. These platforms can be bundled with CPU as a kit, or outside the CPU packages as a discreet component mounted on the Printed Circuit Board (PCB). Intel requires their Original Equipment Manufacturer (OEM) partners that create, market, and sell these systems to meet the brand validation requirements and testing to ensure they have been designed and constructed with the proper components including CPU and Intel chipsets. Intel’s brand validation tool would detect any mismatch of CPU and chipset for any system being designed.

Intel manages and implements security best practices throughout every step of their supply chain and works closely with their partners (i.e., Original Design Manufacturer and Original Equipment Manufacturer) to ensure that they meet Intel’s requirements for secure supply chain processes as specified in partner contract agreements. Therefore, end customers can be assured that any system had been designed and tested to conform to Intel’s requirements will always have an Intel PCH that contains the module.

11.3. AES GCM IV

The User shall consider the following requirements and restrictions when using the module. AES-GCM IV is constructed in accordance with SP800-38D in compliance with IG C.H scenario 2. GCM IV generation uses an approved CTR_DRBG that is internal to the module’s boundary and the IV length is at least 96 bits (per SP 800-38D).

11.4. End of Life

The module automatically performs secure sanitization at the conclusion of any service performed by the module. Since the module does not retain any persistent SSPs, the procedures for secure sanitization of the cryptographic module are inherently met.

12. Mitigation of Other Attacks

The module provides mechanism to protect against RSA timing attacks. The OCS's (i.e., module’s hardware component) big-number arithmetic can perform the modular exponentiation operations in constant time. This means, the time taken is only dependent on the size of operands and not dependent on the value of the operands. During modular exponentiation, an extra mathematical step is needed when the processed bit of the key is 1 compared to a zero bit. The OCS’s big-number arithmetic implements a “dummy” step when processing a zero bit from the key such that this processing time is identical to the processing time of a set bit. Using this approach, an observer is unable to determine the number of set and unset bits from observing the timing behavior of the modular exponentiation operation. The CSME Crypto Driver takes advantage of this feature by enabling the functionality in the OCS for private key operations. This implies that the computation time using the private key is constant, hence mitigating timing attacks.

The OCS’s AES block cipher in OCS supports an implementation that is resistant to DPA (Differential Power Analysis) attacks. The mechanism implemented is based on masking AES inputs at every stage with a pseudo-random mask. The seed for the pseudorandom mask generator is programmable.

The OCS’s ECC has protection against known DPA Attacks. This is achieved by randomizing the inputs so there is no correlation to the power consumed and ECC operations. This is done by transforming inputs from one coordinate system (Affine) to another coordinate system (Randomized Jacobian).

Appendix A. Glossary and Abbreviations

AES	Advanced Encryption Standard
ADL-P	Alder Lake P
ADL-S	Alder Lake S
API	Application Program Interface
CBC	Cipher Block Chaining
CMVP	Cryptographic Module Validation Program
CSME	Converged Security and Manageability Engine
CSP	Critical Security Parameter
CTR	Counter Mode
CVL	Component Validation List
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDAA	Elliptic Curve Direct Anonymous Attestation
ECSCHNORR	Schnorr-type Digital Signature Scheme over Elliptic Curve
FIPS	Federal Information Processing Standards Publication
HMAC	Hash Message Authentication Code
HCU	Hash Computation Unit
KAS	Key Agreement Scheme
KAT	Known Answer Test
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
OAEP	Optimal Asymmetric Encryption Padding
PCH	Platform Controller Hub
PCT	Pair-wise Consistency Test
PSS	Probabilistic Signature Scheme
RPL-M	Raptor Lake M
RPL-P	Raptor Lake P
RPL-S	Raptor Lake S
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm

Appendix B. References

FIPS140-3 **FIPS 140-3 - Derived Test Requirements (DTR)**
March 2020
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-140.pdf

FIPS140-3_IG **Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program**
October 7, 2022
https://csrc.nist.gov/CSRC/media/Projects/cryptographic-module-validation-program/documents/fips%20140-3/FIPS%20140-3%20IG.pdf

FIPS180-4 **Secure Hash Standard (SHS)**
March 2012
http://csrc.nist.gov/publications/fips/fips180-4/fips 180-4.pdf

FIPS186-4 **Digital Signature Standard (DSS)**
July 2013
http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

FIPS197 **Advanced Encryption Standard**
November 2001
http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

FIPS198-1 **The Keyed Hash Message Authentication Code (HMAC)**
July 2008
http://csrc.nist.gov/publications/fips/fips198 1/FIPS-198 1_final.pdf

PKCS#1 **Public Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1**
February 2003
http://www.ietf.org/rfc/rfc3447.txt

SP800-38A **NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques**
December 2001
http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

SP800-38B **NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques**
May 2005
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38b.pdf

SP800-56Arev3 **NIST Special Publication 800-56A Revision 3 - Recommendation for Pair Wise Key Establishment Schemes Using Discrete Logarithm Cryptography**
April 2018
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf

SP800-56Brev2 **NIST Special Publication 800-56B Revision 2 - Recommendation for Pair Wise Key Establishment Using Integer Factorization Cryptography**
March 2019
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Br2.pdf

SP800-90Arev1 **NIST Special Publication 800-90A Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators**
June 2015
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

SP800-90B **NIST Draft Special Publication 800-90B - Recommendation for the Entropy Sources Used for Random Bit Generation**
January 2018
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf

SP800-131Arev2 **NIST Special Publication 800-131A - Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths**
March 2019
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf

SP800-133rev2 **NIST Special Publication 800-133 Revision 2 - Recommendation for Cryptographic**

Key Generation
June 2020
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r2.pdf