Cryptographic Module for Intel® Converged Security and Manageability Engine (CSME)

Hardware Version 3.0

Firmware Versions 2.5 and 2.6

FIPS 140-2 Non-Proprietary Security Policy

Document Version: 1.4

Updated: May 1, 2023

Prepared by:

atsec information security corporation

9130 Jollyville Road, Suite 260

Austin, TX 78759

www.atsec.com

Contents

1. Cryptographic Module Specification 4
1.1. Module Overview 4
1.2. Block Diagrams 7
1.3. Modes of operation 8
1.4. FIPS Approved Algorithms 8
1.5. Non-Approved Algorithms 11
2. Cryptographic Module Ports and Interfaces 13
3. Roles, Services and Authentication 14
3.1. Roles 14
3.2. Services 14
3.3. Operator Authentication 15
4. Physical Security 16
5. Operational Environment 17
5.1. Applicability 17
5.2. Policy 17
6. Cryptographic Key Management 18
6.1. Random Number Generation 19
6.2. Key Generation 19
6.3. Key Establishment 19
6.4. Key Entry / Output 20
6.5. Key / CSP Storage 20
6.6. Key / CSP Zeroization 20
7. Self-Tests 21
7.1. Power-Up Tests 21
7.1.1. Integrity Tests 21
7.1.2. Cryptographic algorithm tests 21
7.2. On-Demand self-tests 22
7.3. Entropy Health Tests 22
7.4. Conditional Tests 23
8. Guidance 24
8.1. Operator's Guidance 24
8.2. Delivery Procedure 24
9. Mitigation of Other Attacks 25

Introduction

This document is the non-proprietary FIPS 140-2 Security Policy for Cryptographic Module for Intel® 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 140-2 (Federal Information Processing Standards Publication 140-2) for a Security Level 1 Firmware-Hybrid module.

1. Cryptographic Module Specification

1.1. Module Overview

The Cryptographic Module for Intel(R) Converged Security and Manageability Engine (CSME) (hereafter referred to as 'the module') is classified as a multiple-chip standalone firmware-hybrid module for FIPS 140-2 purpose. The module consists of both hardware and firmware. The hardware portion is the Converged Security Engine (CSE) and the firmware portion is the crypto driver process of the Manageability Engine (ME). The two portions form the logical cryptographic boundary, and they combine as Converged Security and Manageability Engine (CSME) to perform cryptographic functions for applications executing on the CSME. The module is embodied in the Intel Platform Controller Hub (PCH) “Cannon Point PCH” chipsets shown in Figures 1a and 1b below.

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

Component	Type	Version Number	Description
CSME Crypto Driver	Firmware	2.5 [1], 2.6 [2][1]	It is a firmware running in an internal customized proprietary O/S to communicate with the hardware components of the module in the Intel PCH chipset.
Converged Security Engine (CSE)	Hardware	3.0	It includes AES, SHA-1 and SHA-256 security engines, and big number arithmetic support for implementing asymmetric algorithms in firmware. It is embedded within the Intel PCH chipset.
run_bist	File	N/A	The Crypto Driver will read this file from battery-backed non-volatile storage. This file tracks if the Crypto Driver has previously passed the full set of power-up self-tests.
fips_hmac	File	N/A	This is a file located in the file system of the internal customized proprietary O/S to contain the HMAC-SHA-256 hash value for integrity check of the module.

Table 1 - Cryptographic Module Components

[1] Refer to Table 2 for the corresponding platform configuration.

Figure 1a – Intel Cannon Point PCH with Whiskey Lake (WHL) chipset

Figure 1b – Intel Cannon Point PCH with Coffee Lake (CFL) chipset

The module has been tested on the following multichip standalone platforms:

Platform	*Processor*	*Operating System*
Cannon Point PCH [1]	Coffee Lake CPU	embedded IA-32 customized proprietary O/S dedicated to support the functionality of the CSME, version 12.0.67.1579.
Cannon Point PCH [2]	Whiskey Lake CPU	Embedded IA-32 dedicated to support the functionality of the CSME, version 12.0.70.1652.

Table 2 - Tested Platforms

Note: Per IG G.5, CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when so ported if the specific operational environment is not listed on the validation certificate.

The table below shows the security level claimed for each of the eleven sections that comprise the FIPS 140-2 standard:

FIPS 140-2 Section		**Security Level**
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1
3	Roles, Services and Authentication	1
4	Finite State Model	1
5	Physical Security	1
6	Operational Environment	N/A
7	Cryptographic Key Management	1
8	EMI/EMC	1
9	Self-Tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	1
Overall Level		1

Table 3 - Security Levels

1.2. Block Diagrams

The physical boundary of the module is the physical boundary of the Intel PCH chipset that contains the module. Consequently, the embodiment of the module is a multi-chip standalone cryptographic module.
The module provides cryptographic services to applications through an application program interface (API). The cryptographic logical boundary consists of the firmware component (i.e., CSME Crypto Driver), the files for integrity check, and the hardware components (i.e., CSE). The logical block diagrams below show the module, its interfaces with the operational environment and the delimitation of its logical boundary which are colored in **BLUE**:

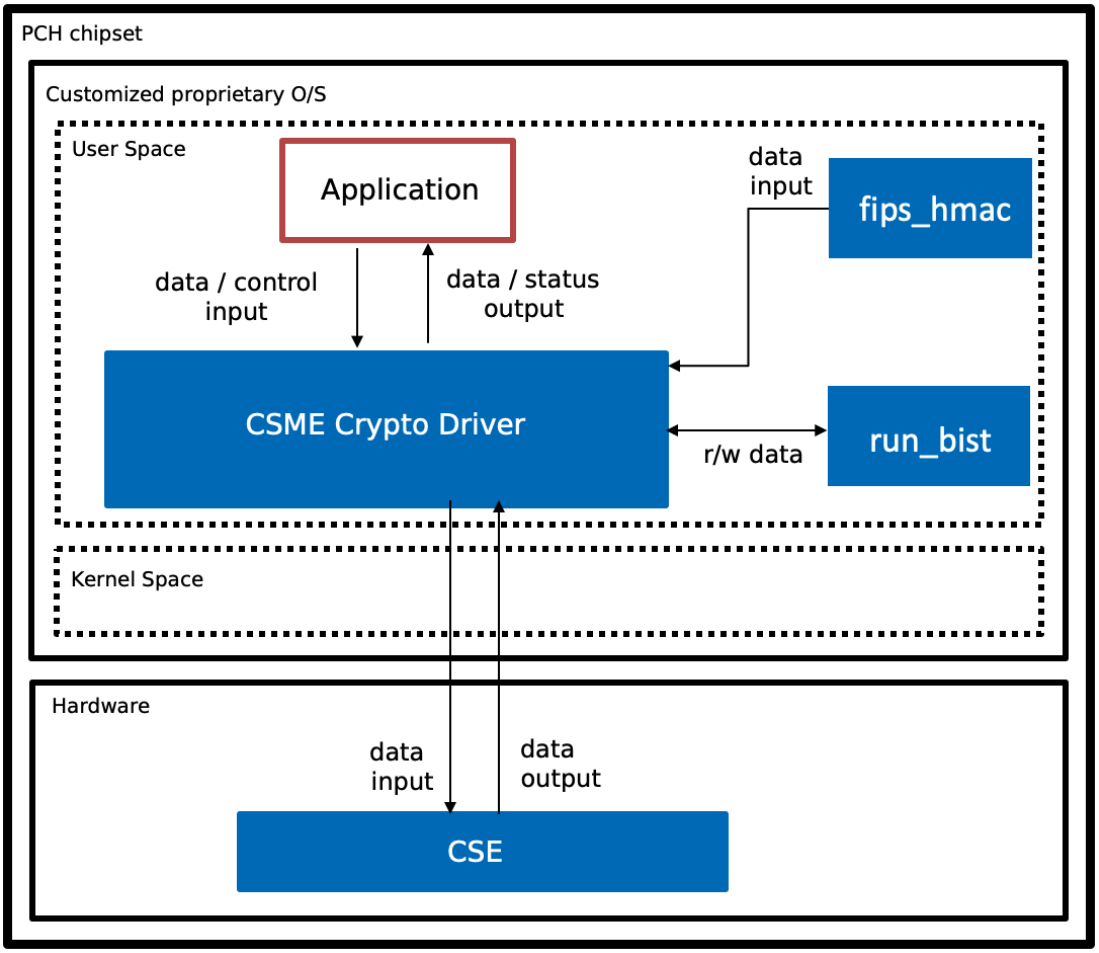

Figure 2 - Logical Block Diagram

1.3. Modes of operation

The module supports two modes of operation:

- In "FIPS mode" (the FIPS Approved mode of operation) only approved security functions with sufficient security strength can be used.
- In "non-FIPS mode" (the non-Approved mode of operation) only non-approved security functions can be used.

There is a difference between entering/exiting FIPS mode and enabling/disabling FIPS operations in BIOS. To enable FIPS operations, the module must first be configured as a FIPS 140-2 module from within the BIOS menu setting at power-on. The system will be reset in order to transition to/from FIPS operational mode after the setting is configured. Once FIPS operations are enabled in the BIOS settings, the module is initialized as a FIPS 140-2 validated module. This means the module will now implicitly be in FIPS mode if the user is requesting a FIPS algorithm. And the module is implicitly not in FIPS mode if the user is requesting a non-FIPS algorithm. If FIPS operations are not enabled within the BIOS settings, then the module is not a 140-2 validated module and cannot enter FIPS mode which means no FIPS services will be available. For more details on the services available in FIPS mode of operation and non-FIPS mode of operation, please refer to Table 6 - Services in FIPS mode of operation and Table 7 - Services in non-FIPS mode of operation in 3.2 Services.

Critical security parameters used or stored in FIPS mode are not used in non-FIPS mode, and vice versa.

The module supports the following Approved cryptographic algorithms:

1.4. FIPS Approved Algorithms

Algorithms	Standards	CAVS Certificates
AES[†] encryption and decryption Key Size: • 128 • 256 mode: • ECB • CBC • CFB128 • CMAC • Counter (CTR) • GMAC • OFB	FIPS 197, SP800-38A, 38B, 38D, 38F	C1769, C1770
AES Key Wrapping Key Size: • 128 • 256 mode: • AES-KW	SP800-38F	C1769, C1770

[†] The module only uses the general-purpose AES engine from the CSE for AES cryptographic operations (i.e., encryption and decryption). Any other AES engine from the hardware component of the module is considered as a dead path since it is not callable via the interface of the CSME crypto driver.

Algorithms	Standards	CAVS Certificates
DRBG • CTR mode AES-256 with Derivation Function without Prediction Resistance	SP 800-90A	C1769, C1770
SHA: • SHA-1 • SHA-224 • SHA-256 • SHA-384 • SHA-512	FIPS 180-4	C1769, C1770
HMAC with: • SHA-1 • SHA-256 • SHA-224 • SHA-384 • SHA-512	FIPS 198-1	C1769, C1770
ECDSA Key Pair Generation / Public Key Verification Curve: • P-256 • P-384	FIPS 186-4	C1769, C1770
ECDSA Signature Generation Curve: • P-256 • P-384 Hash Algorithm: • SHA-224 • SHA-256 • SHA-384 • SHA-512		
ECDSA Signature Verification Curve: • P-256 • P-384 Hash Algorithm: • SHA-1 • SHA-224 • SHA-256 • SHA-384 • SHA-512		

Algorithms	Standards	CAVS Certificates
Key Agreement Scheme - Shared Secret Computation (KAS-SSC) EC Diffie-Hellman (ECDH) Scheme: Ephemeral Unified Shared Secret Computation "EC": Hash Algorithm: SHA2-256 Curve: P-256 "ED": Hash Algorithm: SHA2-384 Curve: P-384	SP 800-56Ar3	A688
Key-Based Key Derivation (KBKDF) Capabilities: KDF Mode: Counter MAC Mode: HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512 Supported Lengths: 8, 160, 1984, 2048 Fixed Data Order: Before Fixed Data Counter Length: 32	SP 800-108	C1769, C1770
RSA Signature Generation Signature Type • PKCS#1 v1.5 • PSS Modulus: • 2048 • 3072 Hash Algorithm: • SHA-224 • SHA-256 • SHA-384 • SHA-512	FIPS 186-4	C1769, C1770
RSA Signature Verification Signature Type: • PKCS#1 v1.5 • PSS Modulus: • 1024 • 2048 • 3072 Hash Algorithm: • SHA-1 • SHA-224 • SHA-256 • SHA-384 • SHA-512		

Algorithms	Standards	CAVS Certificates
RSA X9.31 Key Generation Modulus: • 2048 Public Exponent Mode: Fixed Fixed Public Exponent: 0x10001	FIPS 186-4 Appendix B.3.3	
RSA Signature Generation Signature Type: • PKCS#1 v1.5 • PSS Modulus: • 4096 Hash Algorithm: • SHA-224 • SHA-256 • SHA-384 • SHA-512	FIPS 186-2	C1769, C1770
RSA Decryption Primitive Modulus: 2048	SP 800-56B	CVL: C1769, C1770
RSA-based Key Transport (KTS-RSA) Mode: • KTS-OAEP Modulus: • 2048-bit modulo • both key encapsulation, un-encapsulation methods are supported	SP 800-56B	Vendor Affirmed
Password-Based Key Derivation (PBKDFv2) with HMAC-SHA-1 and HMAC-SHA-256	SP 800-132	Vendor Affirmed
ENT(P) (entropy source)	SP 800-90B	N/A

Table 4 – Validated Cryptographic Algorithm

1.5. Non-Approved Algorithms

Algorithms	Use
AES GCM	Authenticated Data Encryption and Decryption
RC4, SM4	Data Encryption and Decryption
MD5, SM3	Message Digest
HMAC-MD5, HMAC with keys <112 bits	Message Authentication Code
RSA with 1024 bits modulus size	Key Generation
RSA using MD5, SHA-1, or SM3	Digital Signature Generation
RSA using MD5 or SM3	Digital Signature Verification
KTS-RSA using MD5	Key Transport
KTS-RSA using 3072 or 4092-bit modulus	

Algorithms	Use
Trusted Computing Group (TCG) digital signature scheme using elliptic curves: EC Schnorr, EC commit computation, EC DAA	Digital Signature Generation and Verification
Barreto Naehrig, SECG P256k1, Brainpool384 or SM2	ECC cofactor Diffie-Hellman (ECC CDH) shared secret computation
	ECDSA Key Pair Generation and Public Key Verification
	ECDSA Signature Generation and Verification
ECDSA using P-224 or P-521	ECDSA Signature Generation and Verification
ECDH using One-Pass DH	Key Agreement Scheme
SP800-56C one-step KDF	Key Derivation Function

Table 4b – Non-Approved Cryptographic Algorithms

2. Cryptographic Module Ports and Interfaces

As the module is a firmware-hybrid, the data should enter and exit the module via the logical interface through firmware. The logical interfaces are the application program interfaces (API) through which applications request services from the firmware (i.e. Crypto Driver). The hardware interfaces are the interface for data in and out of the hardware components of the module. The following table summarizes the four logical interfaces:

FIPS 140-2 Interface	Logical Interface	Hardware Interface
Data Input	API input parameters pointing to data stored in RAM, fips_hmac file, run_bist file.	DMA
Data Output	API output parameters pointing to RAM locations for storing output data.	DMA
Control Input	API function calls.	IOSF
Status Output	API return codes.	IOSF

Table 5 - Ports and Interfaces

3. Roles, Services and Authentication

3.1. Roles

The module supports the following roles:

- **User role**: performs all services (in both FIPS mode and non-FIPS mode of operation), except module installation and configuration.
- **Crypto Officer role**: performs module initialization.

The User and Crypto Officer roles are implicitly assumed by the entity accessing the module services.

3.2. Services

The module provides services to users that assume one of the available roles. All services are described in detail in the user documentation. Table 6 lists the Approved services in FIPS mode of operation, the roles that can request the service, the Critical Security Parameters involved and how they are accessed:

Service	Role	Key/CSP	Access
Symmetric encryption and decryption	User	AES key	Read
AES key wrapping (KTS)	User	AES key	Read
RSA key generation	User	RSA public/private keys	Create
RSA key validation			Read
RSA digital signature generation based on PKCS#1 v1.5 and PSS scheme			Read
RSA digital signature verification based on PKCS#1 v1.5 and PSS scheme			Read
RSA key wrapping (KTS-OAEP) based on SP800-56B			Read
RSA decryption primitives (RSADP; SP 800-56B Section 7.1)			Read
ECDSA key generation	User	ECDSA public/private keys	Create
ECDSA public key verification			Read
ECDSA signature generation			Read
ECDSA signature verification			Read
SHS Message digest generation	User	None	None
MAC generation and verification	User	At least 112 bits HMAC key, AES key	Read
Random Number Generation	User	DRBG seed (consisting of Entropy Input String), V and Key	Read, Update
Password-based Key Derivation Function	User	Password, Derived Keys	Read, Create
EC Diffie-Hellman Shared Secret Computation (KAS-SSC)	User	ECDSA public/private keys, shared secret	Read, Create

Service	Role	Key/CSP	Access
Secure Key Storage (SKS) To load the 128- or 256-bits key directly from SKS in the register for AES or HMAC operations.	User	AES 128, 256-bit keys, HMAC 128, 256-bit keys, AES master key	Read, Write
Show status	User	None	None
Self-Tests	User	None	None
Zeroization	User	All keys and CSPs	Zeroize
Module Initialization	Crypto Officer	None	None

Table 6 - Services in FIPS mode of operation

The following table lists the services available in non-FIPS mode of operation.

Service	Role
AES-GCM with external IV	User
Message digests using MD5 or SM3	User
MAC generation using HMAC-MD5	User
MAC generation using less than 112-bit HMAC key	User
Symmetric data encryption / decryption using RC4 or SM4	User
Key generation using RSA with 1024 bits modulus size	User
Signature generation using RSA with MD5, SM3 or SHA-1 for any modulus size	User
Signature verification using RSA with MD5, or SM3 for any modulus size	User
Key Transport using RSA with MD5 or with any modulus not listed in Table 4	User
Trusted Computing Group (TCG) digital signature scheme using elliptic curves: EC Schnorr, EC commit computation, EC DAA	User
ECC cofactor Diffie-Hellman (ECC CDH) shared secret computation with Barreto Naehrig, SECG P256k1, Brainpool384 and SM2 curves	User
ECDSA Key Pair Generation and Public Key Verification with Barreto Naehrig, SECG P256k1, Brainpool384 and SM2 curves	User
ECDSA Signature Generation and Verification with Barreto Naehrig, SECG P256k1, Brainpool384 and SM2 curves	User
ECDSA Signature Generation and Verification using NIST P-224 or P-521	User
EC Diffie-Hellman Key Agreement	User
SP800-56C Key Derivation Function	User

Table 7 - Services in non-FIPS mode of operation

3.3. Operator Authentication

The module does not implement authentication. The role is implicitly assumed based on the service requested.

4. Physical Security

The Cryptographic Module is a firmware-hybrid module that operates on a multi-chip standalone platform which conforms to the Level 1 requirements for physical security. The hardware portion of the cryptographic module is a production grade component. The cryptographic module must be used in a commercial-off-the-shelf (COTS) computer device. The computer device shall be comprised of production grade components with standard passivation (a sealing coat applied over the chip circuitry to protect it against environmental and other physical damage) and a production grade enclosure that completely surrounds the cryptographic module.

5. Operational Environment

5.1. Applicability

The module operates in a limited operational environment per FIPS 140-2 level 1 specifications. The module runs on an internal customized proprietary O/S within the Intel PCH chipset.

5.2. Policy

The operating system is restricted to a single operator; concurrent operators are explicitly excluded. The application that requests cryptographic services is the single user of the module.

6. Cryptographic Key Management

The following table summarizes the Keys and Critical Security Parameters (CSPs) that are used by the cryptographic services implemented in the module:

Name	Generation / Entry	Storage	Zeroization
AES keys	The key is passed into the module via API input parameters. The key can also be loaded from the SKS directly to the register.	Stored as plaintext in the RAM or SKS.	Called memset_secure() to replace zero in the memory.
HMAC keys	The key is passed into the module via API input parameters. The key can also be loaded from the SKS directly to the register.	Stored as plaintext in the RAM or SKS.	Called memset_secure() to replace zero in the memory.
RSA key pair	The prime number is generated using SP 800-90A DRBG. The RSA public-private keys are generated using FIPS 186-4 RSA Key Generation method. The key pair is passed into the module via API input parameters.	Stored as plaintext in the RAM.	Called memset_secure() to replace zero in the memory.
ECDSA key pair	The ECDSA public-private keys are generated using FIPS 186-4 ECDSA Key Generation method and the random value used in key generation is generated using SP 800-90A DRBG. The key pair is passed into the module via API input parameters.	Stored as plaintext in the RAM.	Called memset_secure() to replace zero in the memory.
Shared Secret	The shared secret is generated in the EC Diffie-Hellman shared secret computation (KAS-SSC).	Stored as plaintext in the RAM.	Called memset_secure() to replace zero in the memory.
Entropy Input String for DRBG seed and reseed	Obtained from hardware ENT(P) outside of the module's logical boundary but within its physical boundary	Stored as plaintext in the RAM.	Zeroized during the power cycle of the module.
DRBG internal V and Key	Generated internally in the DRBG.	Stored as plaintext in the RAM.	Zeroized during the power cycle of the module.
PBKDF Password	The password is passed into the module via API input parameters.	Stored as plaintext in the RAM.	Called memset_secure() to replace zero in the memory.
PBKDF Derived Key	The PBKDF key is derived following SP800-132	Stored as plaintext in the RAM.	Called memset_secure() to replace zero in the memory.
AES master key	The key is passed into the module via API input parameters.	Stored as plaintext in the SKS	Zeroized by replacing new values.

Table 8 - Life cycle of Keys and Critical Security Parameters (CSP)

The following sections describe how CSPs, in particular cryptographic keys, are managed during its life cycle.

6.1. Random Number Generation

The module implements a CTR_DRBG with AES-256 with derivation function and without prediction resistance. 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.

To seed the CTR_DRBG, the module uses the raw entropy source of a true random number generator TRNG as the entropy source. The entropy source is compliant to IG 7.18 and as such is annotated as “ENT(P)”. The entropy source is implemented outside of the module’s logical boundary but within the Intel PCH chipset physical boundary. The TRNG provides a minimum entropy rate of 0.618 bits per bit. The module collects 576 bits of data from the entropy source for generating the initial seed during initialization of the CTR_DRBG, and reseeding which occurs less than 2^{28} times of DRBG service request.

The module uses Intel’s Online Health Test (OHT) to perform the same functionality of the SP800-90B Continuous Health Tests) on the output of the ENT(P) to ensure that it has not degraded.

6.2. Key Generation

For generating RSA and ECDSA keys the module implements asymmetric key generation services compliant with [SP800-133] and [SP800-90A]. The random value used in asymmetric key generation is obtained from the DRBG. In accordance with FIPS 140-2 IG D.12, the cryptographic module performs Cryptographic Key Generation (CKG) for asymmetric keys as per SP800-133 (vendor affirmed) The module does not offer a dedicated service for generating keys for symmetric algorithms or for HMAC. NOTE: FIPS approved RSA Key Generation will only use public key exponent $e = 2^{16} + 1$

6.3. Key Establishment

The module supports the Key Agreement Method

- SP800-56A EC Diffie-Hellman Key Agreement primitive (KAS-SSC) with the following curves: P-256, P-384.

The module implements the following Key Transport methods

- FIPS approved AES-key wrapping according to SP800-38F
- Vendor Affirmed RSA-OAEP as defined in Section 9.2 of SP800-56B
- RSA decryption primitive component (RSADP CVL) as defined in SP800-56B Section 7.1.

The module implements following Key Derivation methods:

- Password-Based Key Derivation version 2 (PBKDFv2) as defined in [SP800-132].

The PBKDFv2 function is provided as a service and returns the key derived from the provided password to the caller. The module supports HMAC-SHA-1 and HMAC-SHA-256 for PBKDF. Option 1a described in section 5.4 of [SP800-132] is provided to protect data using the derived key. The caller shall observe all requirements and should consider all recommendations specified in SP800-132 with respect to the strength of the generated key, including the quality of the password, the quality of the salt as well as the number of iterations. The security guidance of the PBKDFv2 function is described in section 8.1.
Note: The keys derived from passwords, as shown in SP 800-132, may only be used in storage applications.
CAVEAT:

- EC Diffie-Hellman shared secret computation provides 128 or 192 bits of security strength depending on the curve.
- RSA key wrapping provides 112 bits of encryption strength.
- AES-Key Wrapping provides 128 or 256 bits of encryption strength

6.4. Key Entry / Output

The module does not support manual key entry or intermediate key generation key output. In addition, the module does not produce key output in plaintext format outside its physical boundary.

6.5. Key / CSP Storage

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

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

The module provides Secure Key Storage (SKS) services. The keys stored in the SKS are in plaintext or encrypted with the AES 256 bits master key. The keys stored in SKS are directly loaded into the register for AES or HMAC operations at the hardware level and cannot be retrieved by the CSME Crypto Driver or calling application.

The HMAC key used for integrity test is stored in the fips_hmac file and relies on the operating system for protection.

6.6. Key / CSP Zeroization

The memory occupied by keys is stored in static arrays or allocated by regular memory allocation operating system calls. The firmware of the module (i.e., CSME Crypto Driver) calls the memset_secure() functions to overwrite the memory occupied by keys with “zeros” before deallocating the memory with the regular memory deallocation operating system call. At the hardware level, two Memory-Map I/O (called “MMIO” in short) registers are accessible from CSME Crypto Driver to store the keys temporarily: HCU_KEY and AES_KEY. These two registers are write-only (i.e., user cannot read the keys from the registers) and they are mapped to the CSME Crypto Driver only (i.e., no other process is able to access these registers); therefore, the keys are protected by the hardware architecture before the key zeroization occurs. The keys are zeroized during the power-cycle of the module or replaced by other new keys. All other keys in the hardware components for the cryptographic operations are provided via the SKS which is wired hardware-internally to the cipher engines.

7. Self-Tests

When the module is powered-up and FIPS mode is enabled, the kernel of the operating environment obtains the HMAC value of the module for integrity check from the fips_hmac file and sends the HMAC value to the Crypto Driver. The Crypto Driver will read the run_bist file from battery-backed non-volatile storage. This file tracks if the Crypto Driver has previously passed the full set of power-up self-tests.

Pursuant with IG 9.11 requirements, the full set of power-up self-tests will *not* be run if the run_bist file indicates that the full set tests had previously passed. Instead, the only tests executed at power up are the integrity self-test and the entropy self-test.

If any self-test fails, the module enters Error state by calling a panic() function. When the host PCH (i.e Coffee Lake and Whiskey Lake PCH chipsets) detects the module is halted it triggers a reset to reboot the module and rerun the power-up self-test again. No data output and cryptographic operation are allowed in Error state.

7.1. Power-Up Tests

When the module performs power-up tests, these tests ensure that the module is not corrupted and that the cryptographic algorithms work as expected. While the module is executing the power-up tests, services are not available, and input and output are inhibited. The module does not return control to the calling application until the power-up tests are completed. Once the power-up tests are completed successfully, the module will be operational.

7.1.1. Integrity Tests

The integrity of the module is verified by comparing an HMAC-SHA-256 value calculated at run time with the HMAC value stored in the fips_hmac file that was computed at build time. If the HMAC values do not match, the test fails, and the module enters the Error state.

7.1.2. Cryptographic algorithm tests

When the module performs self-tests on all FIPS-Approved cryptographic algorithms supported in the approved mode of operation, it will use the known answer tests (KAT) and pair-wise consistency test (PCT), shown in the following table:

Algorithm	Power-Up Tests
AES	• KAT AES ECB, encrypt • KAT AES ECB, decrypt • KAT AES CMAC, encrypt
DRBG	• KAT AES CTR_DRBG
HMAC	• KAT HMAC-SHA-1 • KAT HMAC-SHA-512
SHS	• KAT SHA-1 is covered in the KAT for HMAC-SHA-1 as allowed with IG 9.1 • KAT SHA-224 is not required per IG 9.4 • KAT SHA-256 is covered in the Integrity Test which is allowed with IG 9.3 • KAT SHA-384 is not required per IG 9.4 • KAT SHA-512 is covered in the KAT for HMAC-SHA-512 as allowed with IG 9.1
ECDSA	• PCT ECDSA (NIST P-256/ SHA-256) signature generation and signature verification
RSA	• KAT RSA 2048-bit key PKCS#1 v1.5 with SHA-256 signature generation

Algorithm	Power-Up Tests
	• KAT RSA 2048-bit key PKCS#1 v1.5 with SHA-256 signature verification • KAT RSA 2048-bit key OAEP encrypt • KAT RSA 2048-bit key OAEP decrypt
KBKDF	• KAT KDF in CTR mode
KAS-SSC	• KAT Primitive “Z” Computation

Table 9- Self-Tests

For the KAT, the module calculates the result and compares it with the known value. If the answer does not match the known answer, the KAT is failed, and the module enters the Error state.
For the PCT, if the signature generation or verification fails, the module enters the Error state.

7.2. On-Demand self-tests

The on-demand self-tests can be invoked by the user performing these two steps:

1. Reboot and then disable FIPS in BIOS
2. Reboot, then re-enable FIPS in BIOS which will cause POST/KAT to run

During the execution of the on-demand self-tests, no services are available, and no data output or input is possible.

7.3. Entropy Health Tests

FIPS 140-2 section 4.9.2 states that a Continuous Random Number Generator Test (CRNGT) shall be used on the output of any random number generator. The Crypto Driver Firmware of this module has implemented such a test which verifies that each block of data generated by the ENT(P) is not a duplicate of the previous block of data.

In addition, 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 entropy source into the OHT where it tracks the entropy health.

The OHT is designed to have the same functionality and health coverage as:

- 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. The final output from the ENT(P) provides full entropy of 256 bits.

7.4. Conditional Tests

The module performs conditional tests on the cryptographic algorithms using the tests shown in Table 10.

Algorithm	Test
ECDSA key generation	• PCT using SHA-256 signature generation and verification
RSA key generation	• PCT using SHA-256 signature generation and verification
ENT(P) (SP800-90B)	• CRNGT, OHT

Table 10 - Conditional Tests

8. Guidance

8.1. Operator's Guidance

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

- There is a difference between entering/exiting FIPS mode and enabling/disabling FIPS operations in BIOS. To enable FIPS operations, the module must first be configured as a FIPS 140-2 module from within the BIOS menu setting at power-on. The system will be reset in order to transition to/from FIPS operational mode after the setting is configured. If FIPS operations are not enabled within the BIOS settings, then the module is not a 140-2 validated module and cannot enter FIPS mode which means no FIPS services will be available.
- Once FIPS operations are enabled in the BIOS settings, the module is initialized as a FIPS 140-2 validated module following power-on. This means the module will now implicitly be in FIPS mode if the user is requesting a FIPS service. And the module is implicitly not in FIPS mode if the user is requesting a non-FIPS service.
- The operator of the module can call the API call of crypto_drv_fips_mode_status() to check if the module is a FIPS 140-2 validated module. If API call returns 1, then the module is a FIPS 140-2 validated module; if it returns 0, then the module is not a FIPS 140-2 validated module.

The following security guidance for the User role is described below:

- The security guidance of the PBKDFv2 function are described below:
  - The length of the derived keying material shall be at least 112 bits long;
  - The length of the salt generated by an Approved DRBG shall be at least 128 bits long;
  - The iteration count shall be selected as large as possible, at least 100 is recommended;
  - The password shorter than 10 characters are usually considered to be weak;
  - Easily accessed personal information shall not be used directly as a password.
- One RSA key pair shall be used for one cryptographic purpose, such as digital signature and key transport. New RSA key pair is required for different cryptographic purpose.
- In order to use the ECDH shared secret computation in compliance with [800-56Arev3] and IG D.8, the caller shall adhere to the following requirements:
  - The key agreement service shall only be use with keys generated by the module.
  - The keys shall be generated as closet to its time of use as possible.
  - The key shall only be used for one transaction.
  - The key shall be destroyed when no longer needed.

8.2. 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 are able to download this firmware.

The hardware component of the module is contained within the Intel Cannon Point Platform Controller Hub (PCH). The Intel Cannon Point PCH 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 to ensure the systems have been designed and constructed with the proper components including CPU and Intel Cannon Point PCH. Intel's brand validation tool would detect any mismatch of CPU and PCH 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 the Intel Cannon Point PCH that contains the module.

9. Mitigation of Other Attacks

The module provides mechanism to guard against the RSA timing attack. The CSE's 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 CSE'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 CSE for private key operations. This implies that the computation time using the private key is constant, hence mitigating timing attacks.

Appendix A. Glossary and Abbreviations

AES	Advanced Encryption Standard
API	Application Program Interface
CAVS	Cryptographic Algorithm Validation System
CBC	Cipher Block Chaining
CMVP	Cryptographic Module Validation Program
CRNGT	Continuous Random Number Generator Test
CSE	Converged Security Engine (i.e. hardware)
CSME	Converged Security and Manageability Engine (i.e. firmware)
CSP	Critical Security Parameter
CTR	Counter Mode
CVL	Component Validation List
DMA	Direct Memory Access
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
FIPS	Federal Information Processing Standards Publication
HMAC	Hash Message Authentication Code
IG	Implementation Guidance
IOSF	Intel® On-Chip System Fabric
KAS	Key Agreement Schema
KAT	Known Answer Test
MAC	Message Authentication Code
MMIO	Memory-Mapped I/O
NIST	National Institute of Science and Technology
OAEP	Optimal Asymmetric Encryption Padding
OEM	Original Equipment Manufacturers
PBKDF	Password-Based Key Derivation Function
PCH	Platform Controller Hub
PCT	Pair-wise Consistency Test
PSS	Probabilistic Signature Scheme
SHA	Secure Hash Algorithm
RSA	Rivest, Shamir, Addleman
SKS	Secure Key Storage
SKU	Stock Keeping Unit

Appendix B. References

FIPS140-2	**FIPS PUB 140-2 - Security Requirements For Cryptographic Modules** May 2001 http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf
FIPS140-2_IG	**Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program** August 15, 2020 http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.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
SP800-56A	**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
SP800-56B	**NIST Special Publication 800-56B - Recommendation for Pair Wise Key Establishment Using Integer Factorization Cryptography** August 2009 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-56b.pdf
SP800-67	**NIST Special Publication 800-67 Revision 1 - Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher** January 2012 http://csrc.nist.gov/publications/nistpubs/800-67-Rev1/SP-800-67-Rev1.pdf

SP800-90A	**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
SP800-131A	**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
SP800-132	**NIST Special Publication 800-132 - Recommendation for Password-Based Key Derivation - Part 1: Storage Applications** December 2010 http://csrc.nist.gov/publications/nistpubs/800-132/nist-sp800-132.pdf
SP800-133	**NIST Special Publication 800-133 - Recommendation for Cryptographic Key Generation** June 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r2.pdf