Fungible Inc.

Secure Boot Processor (SBP) Crypto Engine FIPS 140-2 Non-Proprietary Security Policy

Firmware Version: 95b53165a1

Sub-chip Hardware Versions: F1 1.0.0 and S1 1.0.1

Date: March 31, 2023

Prepared for:	**Prepared by:**
Fungible, Inc.	**Acumen Security, LLC.**
3201 Scott Blvd.	2400 Research Blvd.
	Suite 395
Santa Clara, CA 95054	Rockville, MD 20850
United States of America	United States of America
	Phone: +1 703 375 9820
www.fungible.com	www.acumensecurity.net

Introduction

Federal Information Processing Standards Publication 140-2 — Security Requirements for Cryptographic Modules specifies requirements for cryptographic modules to be deployed in a Sensitive but Unclassified environment. The National Institute of Standards and Technology (NIST) and Canadian Centre for Cyber Security (CCCS) Cryptographic Module Validation Program (CMVP) run the FIPS 140 program. The NVLAP accredits independent testing labs to perform FIPS 140 testing; the CMVP validates modules meeting FIPS 140 validation. Validated is the term given to a module that is documented and tested against the FIPS 140 criteria.

More information is available on the CMVP website at:
https://csrc.nist.gov/groups/STM/cmvp/index.html

About this Document

This non-proprietary Cryptographic Module Security Policy for SBP Crypto Engine from Fungible Inc. provides an overview of the product and a high-level description of how it meets the overall Level 1 security requirements of FIPS 140-2.

The SBP Crypto Engine may also be referred to as the "module" in this document.

Notices

This document may be freely reproduced and distributed in its entirety without modification.

Table of Contents

List of Tables

List of Figures

1. Overview

The Fungible Inc. Secure Boot Processor (SBP) Crypto Engine (hereafter referred to as the “module”) is defined as a Sub-chip cryptographic subsystem consisting of circuitry cores and associated firmware which represents a sub-chip cryptographic subsystem boundary of a single-chip hardware module per FIPS 140-2 Implementation Guidance 1.20. The module resides in the Fungible F1 and S1 Data Processing Unit (DPU) SoCs (System-on-a-Chip). From the validation perspective, the SBP Crypto Engine consists of hardware-based cryptographic engines and associated firmware which represents a sub-chip cryptographic subsystem boundary. The cryptographic services provided by the SBP Crypto Engine module to calling applications and components are as follows:

- Random bit generation;
- Key transport; and
- Asymmetric key generation, signing and verification.

Please refer to Table 6 for the algorithm certificates of the FIPS approved algorithms listed.

2. FIPS 140-2 Security Levels

The following table lists the level of validation for each area in FIPS 140-2:

FIPS 140-2 Section Title	Validation Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services, and Authentication	1
Finite State Model	1
Physical Security	1
Operational Environment	N/A
Cryptographic Key Management	1
Electromagnetic Interference / Electromagnetic Compatibility	1
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	N/A
Overall Level	1

Table 1 – Validation Level by FIPS 140-2 Section

3. Cryptographic Module Specification

3.1 Cryptographic Boundary

The module is comprised of hardware and firmware components which perform cryptographic functions and acceleration capabilities in Fungible's DPUs.

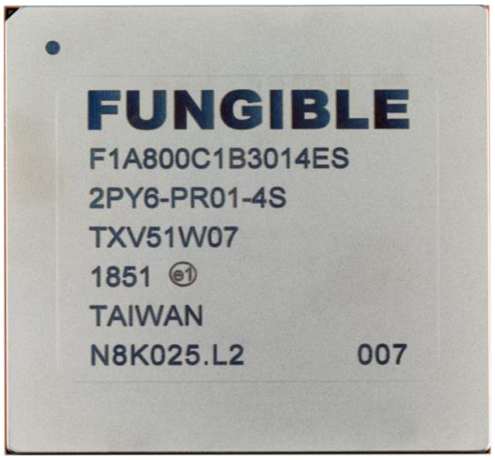

Figure 1 – Fungible F1 Data Processing Unit (Fungible F1 rev A0)

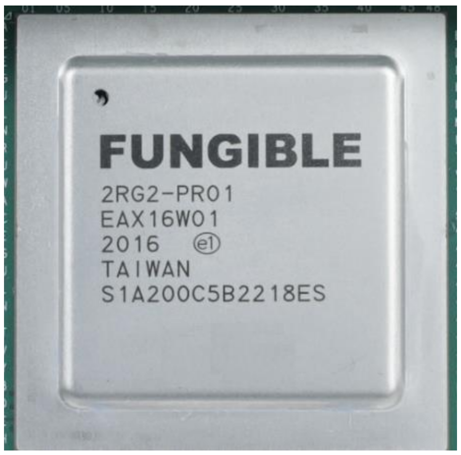

Figure 2 -Fungible S1 Data Processing Unit (Fungible S1 rev A0)

All SBP hardware services are accessed through a messaging interface provided by the firmware component as shown in the block diagram below. Figure 3 also shows the logical relationship of the cryptographic module to the other firmware and hardware components of the host SoC:

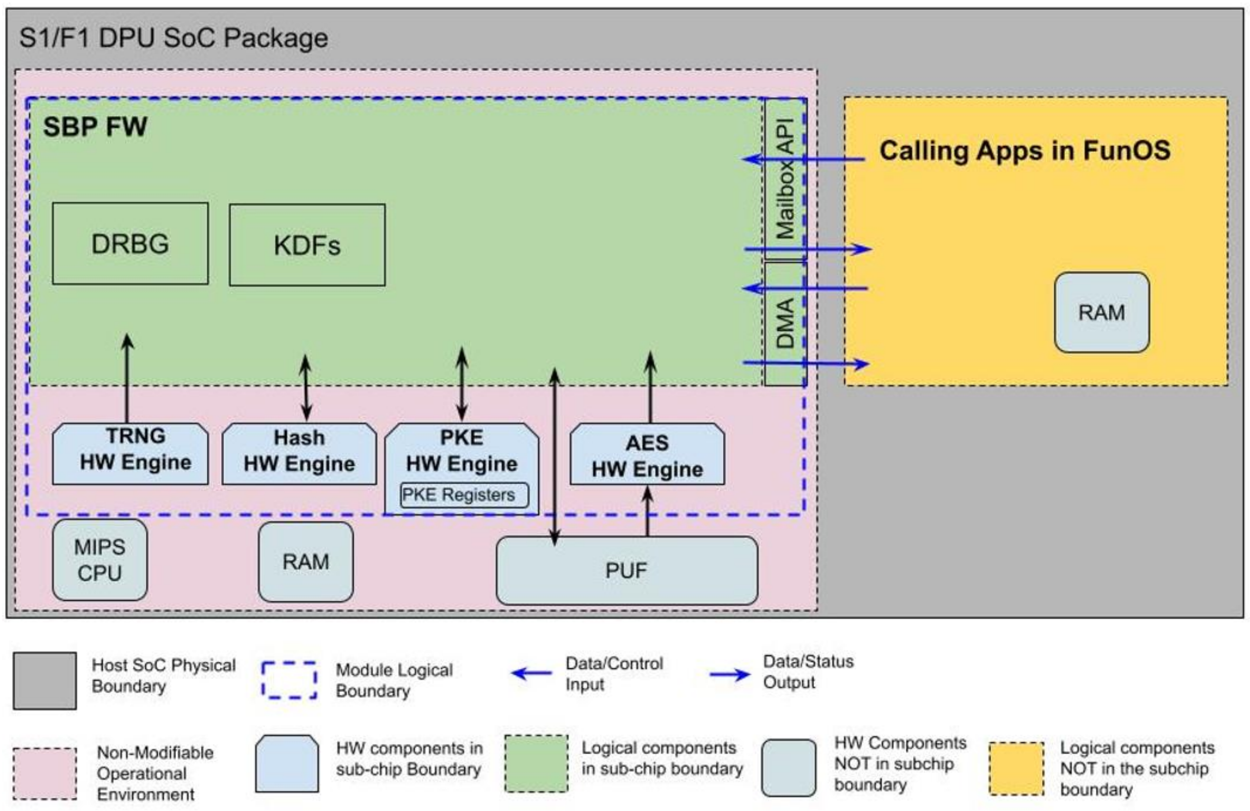
S1/F1 DPU SoC Package
SBP FW
DRBG
KDFs
Mailbox API
DMA
Calling Apps in FunOS
RAM
TRNG HW Engine
Hash HW Engine
PKE HW Engine
PKE Registers
AES HW Engine
MIPS CPU
RAM
PUF
Host SoC Physical Boundary
Module Logical Boundary
Data/Control Input
Data/Status Output
Non-Modifiable Operational Environment
HW components in sub-chip Boundary
Logical components in sub-chip boundary
HW Components NOT in subchip boundary
Logical components NOT in the subchip boundary

Figure 3 – Cryptographic Boundary for SBP Crypto Module in the S1/F1 DPU

The hardware physical boundary of the module is defined as a single-chip, corresponding to the Host DPU SoC package in which the module sub-chip components are integrated. The Host SoC includes the CPU and operational memory on which the firmware is executed. The firmware is stored in FLASH memory directly connected to the chip. The sub-chip cryptographic subsystem includes a single firmware binary which utilizes the cryptographic primitives in hardware and implements higher level cryptographic functions. The firmware component provides the only interface to calling applications. All related functionality to the module is contained within the physical boundary as shown in Figure 1 and 2.

Module Component	Description
SBP Firmware (version: 95b53165a1)	The module's firmware is stored in Flash and loaded into dedicated RAM, and the module firmware is executed on a dedicated MIPS 32-bit CPU.
TRNG HW Engine (F1 version: 1.0.0, S1 version: 1.0.1)	The host SoC contains a TRNG hardware block. The TRNG can be accessed directly as a service and is also used as the source of entropy for seeding the module's Approved DRBG. The TRNG collects noise directly from hardware circuits without Host interaction, by sampling the output of fast free-running ring oscillators.

Module Component	Description
Hash HW Engine (F1 version: 1.0.0, S1 version: 1.0.1)	The Hash Engine hardware block implements SHA1, SHA224, SHA256, SHA384, SHA512, HMAC-SHA1, HMAC-SHA224, HMAC-SHA256 and HMAC-SHA512. The primitive algorithms are utilized by the firmware as part of hashing and higher-level key-hash, asymmetric and key agreement functions.
PKE HW Engine (F1 version: 1.0.0, S1 version: 1.0.1)	The Public Key Engine (PKE) hardware block operates as a large integer arithmetic logic unit. It supports mathematical and logical operations required for implementing public-key algorithms based on the discrete-logarithm problem, the integer factorization problem, and the prime-field elliptic-curve discrete logarithm problem.
AES HW Engine (F1 version: 1.0.0, S1 version: 1.0.1)	The AES engine block implements the primitive AES CBC, CMAC and GCM used in key derivation and key wrapping. The AES engine can load AES keys from a directly attached Physical Unclonable Function (PUF) hardware block without exposing these keys to the SBP CPU or RAM. These PUF keys are derived entirely in the PUF (located outside of the module's boundary but within the host SoC) and wired directly to the AES engine via a direct private interface.

Table 2 – Sub-Chip Module Components

4. Modes of Operation

The module supports two modes of operation: Approved and Non-Approved. The module will be in FIPS-approved mode when all power up Self-Tests have completed successfully, and only Approved or allowed algorithms are invoked. See Table 6 for a list of the supported Approved algorithms and Table 7 for the Allowed algorithms. The non-Approved mode is entered when a non-Approved algorithm is invoked. See Table 8 for a list of non-Approved algorithms.

5. Cryptographic Module Ports and Interfaces

The Data Input interface consists of the input parameters of the API functions in the firmware portion of the module. The Data Output interface consists of the output parameters of the API functions. The Control Input interface consists of the actual API input parameters. The Status Output interface includes the return values of the API functions.

FIPS Interface	Physical Ports	Logical Interfaces
Data Input	Registers; DMA.	Mailbox API input parameters DMA API input parameters

FIPS Interface	Physical Ports	Logical Interfaces
Data Output	Registers; DMA.	Mailbox API output parameters and return values DMA API output parameters and return values
Control Input	Register DMA	Mailbox API input parameters DMA API input parameters
Status Output	Registers DMA	Mailbox API return values DMA API return values
Power Input	Physical pin utilized for power on the host SoC	Power

Table 3 – Ports and Interfaces

All the data output is inhibited during self-test, FIPS error or zeroization. When the module is performing self-tests, or is in an error state, all output on the module's logical data output interfaces is inhibited.

6. Roles, Services and Authentication

The cryptographic module implements both User and Crypto Officer (CO) roles. The module does not support user authentication. The User and CO roles are implicitly assumed by the entity accessing services implemented by the module. A user is considered the owner of the thread that instantiates the module and, therefore, only one concurrent user is allowed.

The Approved services supported by the module and access rights within services accessible over the module's public interface are listed in the table below:

Service	Description	Approved security functions	Keys and/or CSPs	Roles	Access rights to keys and/or CSPs
Module Initialization	Validate integrity of firmware image, POSTs	N/A	N/A	CO	N/A
Signature Generation and Verification	Perform signature generation and verification functions	ECDSA, DSA, RSA, SHA	Certificate Client Private Key, Client Public Key	User	R/W
Key Agreement	Generates ECDH and DH key pairs and shared secrets for key agreement	ECDH, DH	Ephemeral Client Private Key, Client Public Key, EC Diffie-Hellman/ Diffie-Hellman Shared Secret	User	G/R/W/E
Import Private Key	Import a private key using PKCS#8 (plaintext) or CMS	ANSI X9.63, ECDH, SHA	Certificate Client Private Key, ANSI X9.63 Shared Secret, CMS Key Encryption Key	User	G/R/W/E

Service	Description	Approved security functions	Keys and/or CSPs	Roles	Access rights to keys and/or CSPs
Key Transport	Transports keys outside the module using KTS-RSA	KTS-RSA	RSA keys	User	G/R/W/E
Key Output	Output encrypted private keys to the calling application.	KBKDF, AES-GCM	CMAC KEK, Client Private Keys	User	R/W/E
TRNG Entropy Output	Provides entropy data to the calling application	ENT-P	N/A	User	G/R/W
Self-tests	Perform self-tests by power-cycling	N/A	N/A	User, CO	N/A
Zeroization	Zeroize CSPs	N/A	All keys	User, CO	Z
Get PUF Public Key	Gets the ECDH public key from the PUF	N/A	PUF ECDH Public Key	User, CO	R
Get Status	Returns module status	N/A	N/A	User, CO	N/A
Get Image Version	Returns the version number of all images stored on the chip	N/A	N/A	User, CO	N/A
Get Serial Number	Returns the 24-byte serial number of the chip	N/A	N/A	User, CO	N/A
Get Time Stamp	Returns the value of the SBP Real Time Clock	N/A	N/A	User, CO	N/A

Table 4 – Approved Services, Roles and Access Rights

Table 4 defines the relationship between access to CSPs and the different module services. The modes of access shown in the table are defined as:

- G = Generate: The module generates the CSP.
- R = Read: The module reads the CSP. The read access is typically performed before the module uses the CSP.
- E = Execute: The module executes using the CSP.
- W = Write: The module writes the CSP. The write access is typically performed after a CSP is imported into the module, when the module generates a CSP, or when the module overwrites an existing CSP.
- Z = Zeroize: The module zeroizes the CSP.

The module provides the following non-Approved services, which utilize algorithms listed in Table 8:

Service	Non-Approved Functions
Public Key Algorithms	ECDSA (non-compliant), RSA (non-compliant), KAS EC DH (non-compliant)
Key Agreement	KAS EC DH (non-compliant)

Table 5 – Non-Approved Services

7. Physical Security

The module is a sub-chip module embedded in a single-chip host SoC, the Fungible DPU (F1 or S1) SoC, which conforms to the Level 1 requirements for physical security. The host SoC is constructed from production-grade components.

8. Operational Environment

SBP's physical embodiment is a single chip hardware module. The procurement, build and configuring procedure are controlled. Therefore, the operational environment is considered non-modifiable.

9. Cryptographic Algorithms & Key Management

9.1 Approved Cryptographic Algorithms

The module implements the following FIPS 140-2 Approved algorithms:

CAVP Cert #	Algorithm	Standard	Mode/Method/Size	Use
AES HW Engine				
A2318 A2323	AES CBC	FIPS 197 SP 800-38A	128, 192, 256	Encryption, Decryption
A2318 A2323	AES CMAC	SP 800-38B	128, 192, 256	Authenticated Generation, Authenticated Verification
A2318 A2323	AES GCM	SP 800-38D	128 Note: 192-bit and 256-bit were tested but are not used by the module.	Authenticated Encryption, Authenticated Decryption
A2318 A2323	KTS	SP 800-38D, IG D.9	128 (key establishment methodology provides 128 bits of encryption strength	Key transport using AES-GCM
TRNG Engine				
A2321 A2322	AES ECB	FIPS 197 SP 800-38A	128	Encryption, Decryption
	ENT (P)	SP 800-90B	AES-CBC-MAC (128-bit)	Vetted Conditioning Component
Hash HW Engine				
A2319 A2320	HMAC	FIPS 198-1	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	Keyed-Hash Message Authentication
A2319 A2320	SHS	FIPS 180-4	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	Hashing
PKE HW Engine				
A2349 A2350	DSA	FIPS 186-4	2048,224, 2048,256, 3072,256 With SHA-1*, SHA2-224, SHA2-256, SHA2-384, SHA2-512 *SHA-1 is used in Signature verification only	Signature Generation, Signature Verification

CAVP Cert #	Algorithm	Standard	Mode/Method/Size	Use
A2349 A2350	ECDSA	FIPS 186-4	P-224, P-256, P-384, P-521. P-192 for PKV and Signature Verification With SHA-1*, SHA2-224, SHA2-256, SHA2-384, SHA2-512 *SHA-1 is used in Signature verification only	Key Generation[1], Public Key Verification, Signature Generation, Signature Verification
A2349 A2350	KAS-ECC-SSC	SP 800-56Arev3	ECC (P-224, P-256, P-384, P-521)	ephemeralUnified
A2349 A2350	ECC CDH (CVL)	SP 800-56Arev3	P-224, P-256, P-384, P-521	KAS-ECC CDH-Component
A2349 A2350	KAS-FFC-SSC	SP 800-56Arev3	MODP-2048, MODP-3072, MODP-4096, ffdhe2048, ffdhe3072, ffdhe4096	dhEphem
A2349 A2350	KTS-RSA	SP 800-56Brev2	rsakpg1-basic, rsakpg1-crt and rsakpg1-prime-factor Scheme: KTS-OAEP-basic (2048, 3072, 4096) (key establishment methodology provides between 112 and 150 bits of encryption strength)	Key Wrapping/Key Unwrapping
A2349 A2350	RSA (CVL)	SP 800-56Brev2	CRT	Signature Primitive

[1] ECDSA Key Generation is used only as supporting functionality to SP 800-56Arev3 EC Diffie Hellman Key Generation.

CAVP Cert #	Algorithm	Standard	Mode/Method/Size	Use
A2349 A2350	RSA	FIPS 186-4	pkcs1v1.5/PSS (2048, 3072, 4096) With SHA-1*, SHA2-224, SHA2-256, SHA2-384, SHA2-512 *SHA-1 is used in Signature verification only	Signature Generation, Signature Verification
Firmware Crypto Engine				
A2342	DRBG	SP 800-90A	Hash (SHA2-512[2])	Random Bit Generation
A2342	KBKDF	SP 800-108	KDF Mode: Counter (AES CMAC 128-bit)	Key-Based Key Derivation Function
A2342	KDF (CVL)	SP 800-135	ANSI X9.63 KDF (SHA2-224, SHA2-256, SHA2-384, SHA2-512)	Hash-based Key Derivation Function
Vendor Affirmed	CKG	SP 800-133rev2	N/A	Asymmetric and Symmetric Key Generation

Table 6 – Approved Algorithms and CAVP Certificates

9.2 Non-Approved but Allowed Cryptographic Algorithms

Algorithm	Use
PKCS#8 (no security claimed)	Importing private keys into the module. The private keys are considered to be input in plaintext.

Table 7 – Allowed Algorithms

Note: See section 12.1 for further details on the use of the PKCS#8 cryptographic function.

9.3 Non-Approved and Non-Allowed Cryptographic Algorithms

The module employs the methods listed in Table 8, which are not allowed for use in a FIPS-Approved mode. Their use will result in the module operating in a non-Approved mode.

Algorithm	Use
ECDSA (Signature Generation; Signature Verification) (non-compliant)	● Binary curves (B and K series: B-233, K-233, B-283, K-283, B-409, K-409, B-571, K-571). ● Edwards curves: Ed25519 (EdDSA). ● Arbitrary curves on binary and prime fields.

[2] SHA2-224, SHA2-256 and SHA2-384 were also tested but are not used by the module

Algorithm	Use
KAS EC DH (non-compliant)	● Binary curves (B and K series: B-233, K-233, B-283, K-283, B-409, K-409, B-571, K-571). ● Montgomery X22519, X448. Edwards E-521. ● Arbitrary curves on binary and prime fields.
RSA (Signature Generation; Signature Verification) (non-compliant)	● 1024-bit Signature Generation ● 1024-bit Signature Verification

Table 8 – Non-Approved Algorithms

9.4 Key & CSP Table

The following table shows the Keys and CSPs utilized by the module:

Key/CSP Name	Key Description	Generated/ Input	Output	Storage	Zeroization
PUF ECDH Public Key	ECC P-256 key. Published in a certificate. Used in CMS key import and for key agreement	Generated outside of the module in the PUF. Passed in as part of the Get PUF Public Key function.	N/A	RAM	Power Cycle

Key/CSP Name	Key Description	Generated/ Input	Output	Storage	Zeroization
Client Public Key	Public keys used for asymmetric operations including signature generation, verification, key transport and key agreement ECDSA (P-224, P-256, P-384, P-521), ECDH (P-224, P-256, P-384, P-521), RSA (2048, 3072 and 4096), DSA (n = 224, 256 and l= 2048 and 3072), DH (MODP-2048, MODP-3072, MODP-4096, ffdhe2048, ffdhe3072 and ffdhe4096)	Generated internally or passed in plaintext. ECDH keys are generated internally using FIPS 186-4 key generation method and the SP 800-90A DRBG. DH keys are generated internally using the SP 800-56Arev3 Safe Primes key generation method and the SP 800-90A DRBG ECDSA, RSA and DSA keys are only input and not generated by the module.	Passed as return value (when SBP generated)	PKE register and RAM	Power Cycle
Ephemeral Client Private Key	Ephemeral Private keys used for key agreement ECDH (P-224, P-256, P-384, P-521) DH (MODP-2048, MODP-3072, MODP-4096, ffdhe2048, ffdhe3072 and ffdhe4096)	ECDH keys generated internally using FIPS 186-4 key generation method and the SP 800-90A DRBG DH keys generated internally using the SP 800-56Arev3 Safe Primes key generation method and the SP 800-90A DRBG	Output encrypted with CMAC KEK	PKE register	Zeroized after the PKE operation is complete; or Power Cycle
Certificate Client Private Key	Private keys used for asymmetric operations (sign, decrypt, key agreement) ECDSA (P-224, P-256, P-384, P-521), ECDH (P-224, P-256, P-384, P-521), RSA(2048, 3072, 4096), DSA (n = 224, 256 and l= 2048 and 3072), DH (MODP-2048, MODP-3072, MODP-4096, ffdhe2048, ffdhe3072 and ffdhe4096)	Generated outside of SBP. Input as CMS or PKCS#8 (plaintext) payload.	Output encrypted with CMAC KEK	PKE register	Zeroized after the PKE operation is complete; or Power Cycle

Key/CSP Name	Key Description	Generated/ Input	Output	Storage	Zeroization
EC Diffie-Hellman/ Diffie-Hellman Shared Secret	KAS-SCC for FFC and ECC. Shared Secret Computation on behalf of the Client	Generated according to NIST SP 800-56Ar3	Output as an API parameter to the Client	PKE register and RAM	Power cycle
KTS (OAEP) Wrapping Key	RSA Public Key for the purpose of transporting client keys with RSA as specified in NIST SP 800-56B. 2048, 3072, 4096 bits	Generated outside of the module. Passed in as an API parameter in plaintext.	N/A	RAM	Power cycle
KTS (OAEP) Unwrapping Key	RSA Private Key (CRT) for the purpose of transporting keys with RSA as specified in NIST SP 800-56B**.** 2048, 3072, 4096 bits	Generated outside of the module. Imported as argument using either CMS or PKCS#8 (plaintext).	Output encrypted with CMAC KEK	RAM	Power cycle
CMAC KEK	AES GCM 128-bit key used to encrypt/decrypt Client Private Keys	Derived internally according to SP 800-108 KDF with CMAC using the PUF AES Key[3].	N/A	RAM	Power cycle
CMS Key Encryption Key	AES key used to decrypt the Client Private Key when sent in CMS format.	Derived from the ANSI X9.63 KDF.	N/A	RAM	Power cycle
ANSI X9.63 Shared Secret	ECDH or DH derived secret value used in the hashed-based key derivation function X9.63	Generated internally according to SP800-135 and RFC 3278.	N/A	RAM	Power cycle
Entropy Input String	1024-bit[4] Seed used to initialize the internal DRBG used for key generation.	TRNG seed used to instantiate a SHA 512 Hash DRBG (security strength 256)	N/A	RAM	Power cycle
DRBG Internal	Internal V and C	Generated internally by the SP 800-90A DRBG	N/A	RAM	Power cycle

Table 9 – Keys and CSPs

9.5 Key Generation and Entropy

The module is a hardware module which includes a firmware-based Hash_DRBG conformant to SP 800-90A, which is seeded by an SP 800-90B compliant entropy source. The module's DRBG is seeded with 888 bits. The module’s entropy source is consistent with Scenario 1 (b) described in FIPS 140-2 IG 7.14.

For generating EC Diffie-Hellman keys the module implements asymmetric key generation services compliant with FIPS 186-4 and using a DRBG compliant with SP 800-90A. Diffie Hellman keys are generated using the SP 800-56Ar3 Safe Primes key generation method and using a DRBG compliant with

[3] The PUF AES key never enters the module. It is stored in output registers of the PUF and is used by accessing the address space via the PUF hardware channel.

[4] 1024 bits is a conservative value to make sure we have at least the minimum 256 bits of entropy for the target 256-bit security strength (as defined in SP800-57)

SP 800-90A. The random value used in asymmetric key generation is obtained from the DRBG. The public and private key pairs used in the Diffie-Hellman and EC Diffie-Hellman key agreement schemes are generated internally by the module using the same ECC and FCC key generation schemes compliant with FIPS 186-4 and SP 800-56Ar3.

Symmetric keys may also be derived from the shared secret established by EC Diffie-Hellman in a manner that is compliant to SP 800-56Ar3, SP 800-108 or SP 800-135.

In accordance with FIPS 140-2 IG D.12, the cryptographic module performs Cryptographic Key Generation (CKG) for symmetric and asymmetric keys as per SP 800-133rev2 Section 5.2 and Section 6.2.2 (vendor affirmed). The generation mechanism uses an unmodified output from the approved DRBG.

9.6 Key Storage

The cryptographic module itself does not perform persistent storage of keys. Keys and CSPs are passed to the module by components (such as the PUF depicted in Figure 3) which are outside of the logical boundary but reside in the physical boundary. Ephemeral keys and CSPs are stored in volatile memory in plaintext. Keys and CSPs residing in internally allocated data structures (during the lifetime of an API call) can only be accessed using the module's defined API. The non-modifiable operational environment in the DPU (denoted in Figure 1 and Figure 2) protects memory and process space from unauthorized access.

9.7 Key Zeroization

The module does not store keys persistently. The calling application is responsible for parameters passed in and out of the module. The Operating System and the calling application are responsible to clean up temporary or ephemeral keys.

The module explicitly erases the portions of RAM which may hold keys, CSPs and intermediate values. The PKE registers are cleared on every cryptographic operation and following power-on reset.

All CSPs can be zeroized by power cycling or by rebooting the host platform.

10. Self-tests

FIPS 140-2 requires the module to perform self-tests to ensure the integrity of the module and the correctness of the cryptographic functionality at start up. The supported tests are listed and described in this section.

10.1 Power-On Self-Tests

Power-on self-tests are run upon the initialization of the module and do not require operator intervention to run. If any of the tests fail, the module will not initialize. The module will enter an error state and no services can be accessed.

The module implements the following power-on self-tests:

Type	Test
Integrity Test	HMAC-SHA-256
Known Answer Test	AES CBC encrypt/decrypt KAT (128-bits)

Type	Test
	AES CMAC generation/verification KAT (128-bits)
	AES GCM encrypt/decrypt KAT (128-bits)
	HMAC-SHA-1 KAT
	HMAC-SHA-512 KAT
	SHA-1 KAT
	SHA-512 KAT
	Hash_DRBG KAT (SHA-512) (Includes SP 800-90A Section 11.3 Health Checks)
	SP 800-108 KBKDF KAT
	ANSI X9.63 KDF KAT (SHA-1)
	SP800-56A rev3 KAT for ECC and FFC ● ECDH primitive "Z" KAT (Curves used: P-256) ● DH primitive "Z" KAT (Moduli: 2048 bit)
	RSA PKCS1 v1.5 Sign/Verify KAT (2048-bit)
	RSA OAEP Encrypt/Decrypt KAT (2048-bit)
	ECDSA Sign/Verify PCT (P-256)
	DSA Sign/Verify KAT (2048 bit)
Entropy Tests	Repetition Count Test (RCT) on TRNG
	Adaptive Proportion Test (APT) on TRNG

Table 10 – Power-on Self-tests

The power-on self-tests can be run on demand by calling the function FIPS_powerupSelfTest() or by power-cycling the host platform. The module outputs "FIPS self-tests: Success!" when all self-tests have passed successfully.

10.2 Conditional Self-Tests

Conditional self-tests are run during operation of the module. If any of these tests fail, the module will enter an error state, where no services can be accessed by the operators. The module can be reinitialized to clear the error and resume FIPS mode of operation. The module performs the following conditional self-tests:

Type	Test
Firmware Load Test	HMAC-SHA-256 performed over firmware image when loaded into memory.
Repetition Count Test (RCT) for TRNG	This test is intended to identify if the noise source is repeating a given value continuously. The test is implemented per the details of SP 800-90B section 4.4.1.
Adaptive Proportion Test (APT) for TRNG	The test continuously measures the local frequency of occurrence of a sample value in a sequence of noise source samples to determine if the sample value occurs too frequently. The test is implemented per the details of SP 800-90B section 4.4.2.

Table 11 – Conditional Self-Tests

11. Mitigation of other Attacks

The module is not designed to mitigate against attacks which are outside of the scope of FIPS 140-2.

12. Security Rules and Guidance

The module design corresponds to the following security rules. This section documents the security rules enforced by the cryptographic module to implement the FIPS 140-2 security requirements for the Level 1 module.

1. The module does not provide authentication.
2. The operator shall be capable of commanding the module to perform the power-on self-tests by cycling power.
3. Power-on self-tests do not require any operator action.
4. Data output shall be inhibited during self-tests, zeroization, and error states.
5. Output related to keys and their use is inhibited until the key concerned has been fully generated.
6. Status information does not contain CSPs or sensitive data that if misused could lead to a compromise of the module.
7. There are no restrictions on which keys or CSPs are zeroized by the zeroization service.
8. The module does not support concurrent operators.
9. The module does not have any external input/output devices used for entry/output of data.
10. The module does not enter or output plaintext CSPs from the module's physical boundary.
11. The module does not output intermediate key values.

12.1 Import of PKCS#8 files

Private keys can be imported "obfuscated" into the module from the calling application by providing a PKCS#8 file and its associated password to the module. The module will recover the parameters from the PKCS#8 file and use these together with the separately provided password to reconstitute the key used to encrypt the private key stored in the PKCS#8.

It will then use this key together with the encryption parameters to decrypt the private key stored in the PKCS#8. This is not an approved method of establishing a private key. The module is not claiming any security from this method per IG 1.23 Scenario 1. The Private Key is considered to be sent into the module in plaintext from the FIPS perspective. PKCS#8 does not share the same keys or CSPs that are used by an approved or allowed algorithm for any cryptographic operation in either the approved, or non-approved mode.

The recovered private key (Client Private Key) is immediately re-encrypted using the KEK derived by SP 800-108 KDF using CMAC and re-exported to the application in this new encrypted format. This format allows the encrypted private key to be very efficiently used by SBP (and only this SBP) to perform subsequent asymmetric private key operations.

12.2 Import of Cryptographic Message Syntax (CMS) files

Private keys can be imported into the module encrypted from the calling application by providing a CMS file. A shared secret is derived using the Client Public Key and the PUF ECDH Public Key. The derived shared secret is input into the ANSI X9.63 KDF to derive a key encryption key. The private key is imported into the module encrypted by the key encryption key. The CMS file provides the details of the hash algorithm used by the ANSI X9.63 KDF. When the module receives the encrypted private key, it is able to derive the same key encryption key using the shared secret, ANSI X9.63 KDF and encryption parameters (algorithm, IV) stored in the CMS file. Per SP 800-135, the hash function used in the ANSI X9.63 shall be a hash function from FIPS 180-4 and meet the security strength(s) required by the cryptographic function(s) for which the keying material is being generated.

12.3 Usage of AES-GCM

In the case of AES-GCM, the module's IV is generated internally by the module's Approved DRBG. The DRBG seed is generated inside the module's physical boundary. The IV is 96-bits in length per NIST SP 800-38D, Section 8.2.2 and FIPS 140-2 IG A.5 scenario 2.

13. References and Standards

The following Standards are referred to in this Security Policy:

Abbreviation	Full Specification Name
FIPS 140-2	Security Requirements for Cryptographic modules
FIPS 180-4	Secure Hash Standard (SHS)
FIPS 197	Advanced Encryption Standard
FIPS 198-1	The Keyed-Hash Message Authentication Code (HMAC)
IG	Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program
SP 800-38A	Recommendation for Block Cipher Modes of Operation
SP 800-38B	Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication
SP 800-38D	Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC
SP 800-56A	Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography
SP 800-56B	Recommendation for Pair-Wise Key-Establishment Using Integer Factorization Cryptography
SP 800-90A	Recommendation for Random Number Generation Using Deterministic Random Bit Generators
SP 800-108	Recommendation for Key Derivation Using Pseudorandom Functions
SP 800-132	Recommendation for Password-Based Key Derivation: Part 1: Storage Applications
SP 800-133rev2	Recommendation for Cryptographic Key Generation
SP 800-135	Recommendation for Existing Application-Specific Key Derivation Functions

Table 12 – References and Standards

14. Acronyms and Definitions

Acronym	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher-Block Chaining
CCCS	Canadian Centre for Cyber Security
CMVP	Crypto Module Validation Program
CMS	Cryptographic Message Syntax
CO	Cryptographic Officer
CSP	Critical Security Parameter
CTR	Counter-mode
DH	Diffie-Hellman
DPU	Data Processing Unit
DRBG	Deterministic Random Bit Generator
ECDH	Elliptic Curve Diffie-Hellman
ECB	Electronic Code Book

ECC	Elliptic Curve Cryptography
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FCC	Federal Communications Commission
FIPS	Federal Information Processing Standards
GCM	Galois/Counter Mode
GMAC	Galois Message Authentication Code
HMAC	Key-Hashed Message Authentication Code
IG	Implementation Guidance
KAT	Known Answer Test
KDF	Key Derivation Function
LLC	Limited Liability Company
N/A	Not Applicable
NIST	National Institute of Standards and Technology
NVLAP	National Voluntary Lab Accreditation Program
PKE	Public Key Engine
RAM	Random Access Memory
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SP	Special Publication

Table 13 – Acronyms and Definitions