Mocana Cryptographic Suite B Module

Software Version 6.5.2f

Non-Proprietary Security Policy

Document Version 3.18

September. 29, 2023

DigiCert, Inc.

2801 North Thanksgiving Way

Suite 500

Lehi, Utah 84043

1-800-896-7973

Table of Contents

List of Tables

List of Figures

1. Module Specifications

The Mocana Cryptographic Suite B Module (Software Version 6.5.2f) is a software only, multi-chip standalone cryptographic module that runs on a general-purpose computer. The primary purpose of this module is to provide FIPS Approved cryptographic routines to consuming applications via an Application Programming Interface. The physical boundary of the module is the case of the general-purpose computer. The logical boundary of the cryptographic module is the single shared object (SO), libmss.so.

The cryptographic module runs on the following operating environments:

Table 1 - Operational Environments

SW Version	Operating System	Platform	CPU
6.5.2f	Wind River Linux 9.0 (32-bit)	Xerox Explorer 6.5	Intel Atom E3950 with and without AES-NI (PAA)
6.5.2f	Wind River Linux 9.0 (64-bit)	Xerox Explorer 6.5	Intel Atom E3950 with and without AES-NI (PAA)
6.5.2f	Extreme VOSS 8 (32-bit)	Extreme VSP 5420	ARM Cortex A72 without AES-NI (PAA)
6.5.2f	Extreme VOSS 8 (32-bit)	Extreme VSP 7200	Freescale P2020 without AES-NI (PAA)
6.5.2f	Extreme VOSS 8 (32-bit)	KVM Yocto Linux 4.14 running on Extreme XA 1440	Intel C3558 without AES-NI (PAA)
6.5.2f	Extreme VOSS 8 (64-bit)	KVM Yocto Linux 4.14 running on Extreme XA 1440	Intel C3558 with and without AES-NI (PAA)
6.5.2f	Extreme VOSS 8 (32-bit)	KVM Yocto Linux 4.14 running on Extreme XA 1480	Intel C3758 without AES-NI (PAA)
6.5.2f	Extreme VOSS 8 (64-bit)	KVM Yocto Linux 4.14 running on Extreme XA 1480	Intel C3758 with and without AES-NI (PAA)
6.5.2f	Extreme VOSS 8 (32-bit)	Extreme VSP 4900	Intel C3338 without AES-NI (PAA)
6.5.2f	Extreme VOSS 8 (64-bit)	Extreme VSP 4900	Intel C3338 with and without AES-NI (PAA)
6.5.2f	Extreme VOSS 8 (32-bit)	Extreme VSP 7400	Intel C3758 without AES-NI (PAA)

6.5.2f	Extreme VOSS 8 (64-bit)	Extreme VSP 7400	Intel C3758 with and without AES-NI (PAA)
6.5.2f	Extreme VOSS 8 (32-bit)	Extreme VSP 8400	Freescale P2020 without AES-NI (PAA)
6.5.2f	Extreme VOSS 8 (32-bit)	Extreme VSP 8600	Freescale T1042 without AES-NI (PAA)
6.5.2f	Extreme VOSS 8 (32-bit)	ExtremeSwitching 5520	ARM Cortex A72 without AES-NI (PAA)
6.5.2f	Yocto Linux 4.1.15 (32-bit)	Honeywell Xenon 1952	NXP i.MX 6ULL without AES-NI (PAA)
6.5.2f	Extreme VOSS 8 (32-bit)	Extreme VSP 4900	Intel C3538 without AES-NI (PAA)
6.5.2f	Extreme VOSS 8 (64-bit)	Extreme VSP 4900	Intel C3538 with and without AES-NI (PAA)

The cryptographic module is also supported on the following operating environment for which operational testing was not performed:

- Yocto Linux 4.4.11 on DVF101 using ARM Cortex-A9
- Yocto Linux 4.4.11 on DVF99 using ARM926EJ-S
- Wind River Linux 2.6.27.18 on BCM111XX using ARM1176JZF-S
- Android 9 on Avaya Vantage™ K155 using
  Dual-core ARM Cortex-A72/Quad-core ARM Cortex-A53
- Android 9 on Avaya Vantage™ K175 using
  Dual-core ARM Cortex-A72/Quad-core ARM Cortex-A53
- Android 10 on Avaya Collaboration Unit CU360 using
  Dual-core ARM Cortex-A72/Quad-core ARM Cortex-A53
- Yocto Linux 4.9 on TI AM4376 using ARM Cortex-A9

Note: the CMVP makes no statement as to the correct operation of the module on the operational environments for which operational testing was not performed.

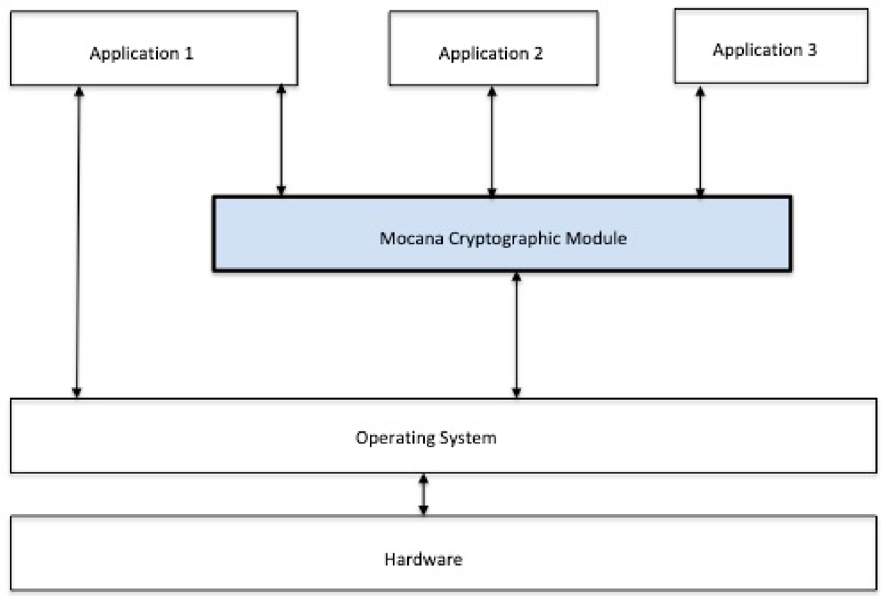

Figure 1: Cryptographic Module Interface Design

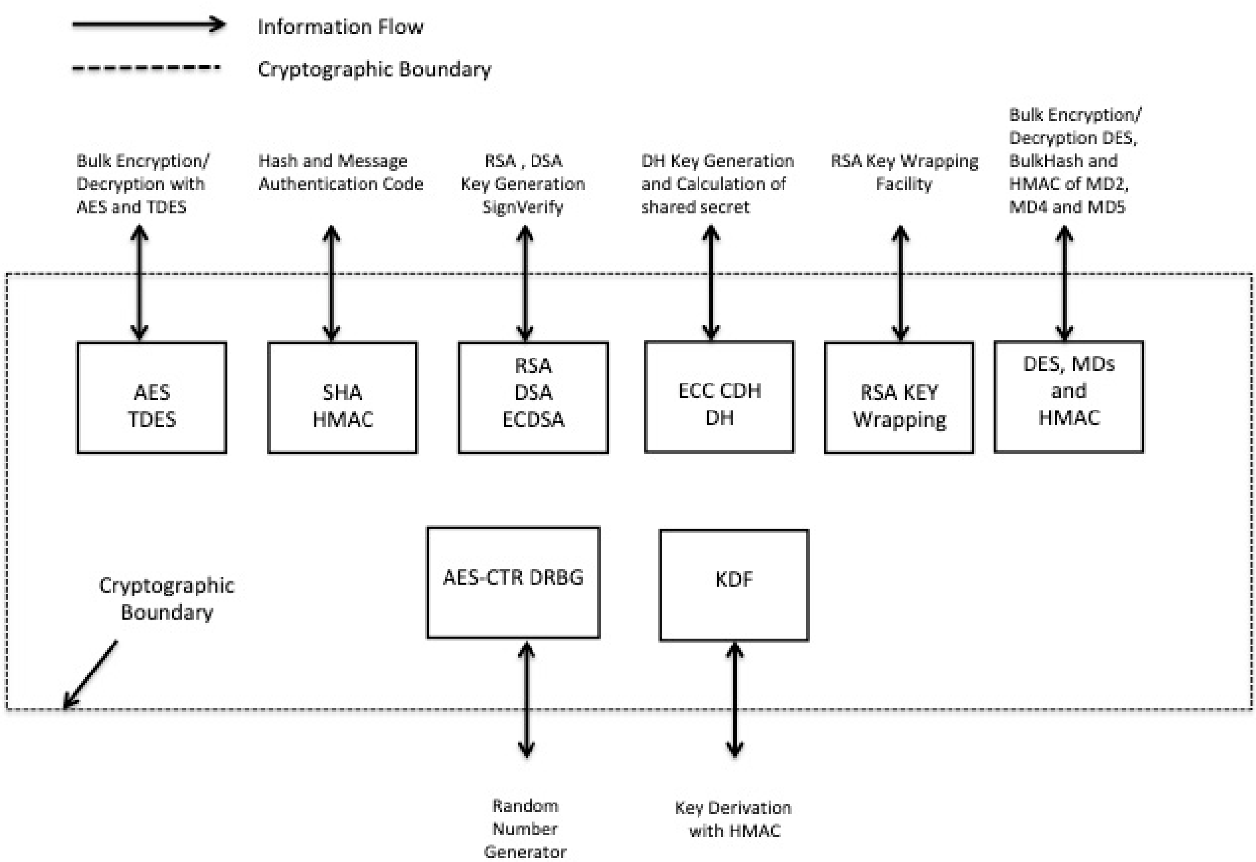

Figure 2: Logical Cryptographic Boundary

2. Security Level

The cryptographic module meets the overall requirements applicable to Security Level 1 of FIPS 140-2.

Table 2 - Module Security Level Specification

Security Requirements Section	**Level**
Cryptographic Module Specification	1
Module Ports and Interfaces	1
Roles, Services and Authentication	1
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	N/A

3. Modes of Operation

Approved Mode of Operation

During module initialization, a consuming application can configure the module to utilize all of the following FIPS Approved algorithms:

Table 3 - Approved Algorithms

Algorithm	Mode/Method/Strength	CAVP Cert #
AES [FIPS 197]	ECB, CBC, OFB, CFB128 and CTR modes; E/D; 128, 192 and 256	C1810, A661
AES [FIPS 197]	CCM 128, 192 and 256, encryption/decryption	C1810, A661
AES [FIPS 197]	CMAC 128, 192 and 256, generation/verification	C1810, A661
AES [FIPS 197]	GCM 4K and GCM 64K, 128, 192 and 256, decryption GMAC 4K and GMAC 64K, 128, 192 and 256, verification	C1811, C1812, A662, A663
AES [FIPS 197]	XTS (128 and 256), encryption/decryption [1]	C1810, A661
DRBG [SP 800-90A]	AES-CTR based DRBG- AES-128, AES-192, AES-256 derivation function enabled: both yes and no	C1810, A661
DSA [FIPS 186-4]	Key Pair Gen: 2048/N=224, 2048/N=256, 3072/N=256 PQG Gen: ○ 2048/N=224 using SHA-224, SHA-256, SHA-384, SHA-512 ○ 2048/N=256 using SHA-256, SHA-384, SHA-512 ○ 3072/N=256 using SHA-256, SHA-384, SHA-512 PQG Ver: ○ 10240/N=160* using SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 ○ 2048/N=224 using SHA-224, SHA-256, SHA-384, SHA-512 ○ 2048/N=256, 3072/N=256 using SHA-256, SHA-384, SHA-512 Sig Gen: ○ 2048/N=224 using SHA-1**, SHA-224, SHA-256, SHA-384, SHA-512 ○ 2048/N=256 using SHA-1**, SHA-224, SHA-256, SHA-384, SHA-512 ○ 3072/N=256 using SHA-1**, SHA-224, SHA-256, SHA-384, SHA-512 Sig Ver: ○ 1024-bit* using SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 ○ 2048/N=224 using SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 ○ 2048/N=256 using SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 ○ 3072/N=256 using SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 * These options are supported for legacy-use only ** Digital Signature Generation using SHA-1 is Approved for use within	C1810, A661

	protocols only.	
ECDSA [FIPS 186-4]	Key Pair: CURVES P; 224, 256, 384, 521 Key Ver: CURVES P; 192, 224, 256, 384, 521 Sig Gen: CURVES P; 224, 256, 384, 521 using SHA-224, SHA-256, SHA-384, SHA-512 (SHA-1 tested but not used) Sig Ver: CURVES P; 192*, 224, 256, 384, 521 using SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 PKV: CURVES P; 192*, 224, 256, 384, 521 * These options are supported for legacy-use only	C1810, A661
HMAC [FIPS 198]	HMAC-SHA-1; HMAC-SHA-224; HMAC-SHA-256; HMAC-SHA-384; HMAC-SHA-512	C1810, A661
KAS-SSC [SP800-56Arev3] with DH, ECC CDH	6.1.2.1 dhEphem Parameter sets: FB and FC, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 Key agreement; key establishment methodology provides between 112 bits and 192 bits of encryption strength 6.1.2.2 Ephemeral Unified Diffie-Hellman Curves: P-224, -256, -384, -521 Key agreement; key establishment methodology provides between 112 and 256 bits of encryption strength	Vendor Affirmed
KBKDF [SP800-108]	KDF in Feedback Mode using HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512, HMAC-SHA3-224**, HMAC-SHA3-256**, HMAC-SHA3-384**, HMAC-SHA3-512** The location of the 8-bit Counter is after the fixed input data ** Tested but not used for the two Wind River Linux OEs (A661)	C1810, A661
RSA [FIPS 186-4]	Key generation: 2048, 3072, 4096-bit** PKCS #1 1.5 and PSS signature generation: 2048, 3072, and 4096-bit** using SHA-224, SHA-256, SHA-384, SHA-512 PKCS #1 1.5 and PSS signature verification: 1024*, 2048, 3072, and 4096-bit** using SHA-1*, SHA-224, SHA-256, SHA-384, SHA-512 * These options are supported for legacy-use only ** 4096-bit does not apply to the two Wind River Linux OEs (C1810)	C1810, A661
SHS [FIPS180-4]	SHA-1 SHA-2: SHA-224; SHA-256; SHA-384; SHA-512	C1810, A661
Triple-DES [SP 800-67]	3-key; TCBC; E/D	C1810, A661

[1] Per IG A.9, the module explicitly checks that AES XTS Key 1 is not equal to Key 2.

During module initialization, a consuming application can configure the module to utilize all, or any subset of the above Approved algorithms. The module's FIPS_powerupSelfTest_Ex() function, which is called during module startup, takes a parameter that points to a configuration table data structure. This data structure contains an array of booleans indexed by an internal Algorithm-ID that will indicate to the module which FIPS algorithms should be initialized for use. The only configuration that was tested as part of the FIPS validation is the configuration which utilized ALL of the Approved algorithms. The CMVP makes no statement as to the correct operation of the module for all other configurations for which operational testing was not performed.

Non-FIPS Approved but Allowed Algorithms

Within the FIPS Approved mode of operation, the module supports the following allowed algorithms:

- RSA (key wrapping; key establishment methodology provides between 112 and 128 bits of encryption strength). Note: Per IG D.9 the module wraps data sent by the requesting application via an API call. Data being wrapped is unknown.

Non-FIPS Approved Mode

In addition to the above algorithms, the following algorithms are available in the non- FIPS Approved mode of operation:

- AES EAX
- AES GCM 4K/GMAC 4K with 128-bit, 192-bit, and 256-bit, encryption/generation (external IV non-compliant per IG A.5)
- AES GCM 64K/GMAC 64K, with 128-bit, 192-bit, and 256-bit, encryption/generation (external IV non-compliant per IG A.5)
- AES GCM (encryption/decryption for 256K implementation)
- AES XCBC
- DES
- Diffie-Hellman (key agreement; key establishment methodology provides less than 112 bits of encryption strength; non-compliant)
- DSA Sig Gen 2048/N=224 using SHA-1
- Elliptic Curve Cryptography Cofactor Diffie Hellman (key agreement; key establishment methodology provides less than 112 bits of encryption strength; non-compliant)
- HMAC-MD5
- MD2, MD4 and MD5
- RNG (FIPS 186-2)
- RSA (FIPS 186-4; 4096-bit usage for Wind River Linux 9.0 only, both 32-bit and 64-bit)
- RSA (key wrapping; key establishment methodology provides less than 112 bits of encryption strength; non-compliant)
- RSA PKCS #1 v2.1 RSAES-OAEP encryption/decryption
- Triple-DES, 2 key

Note: All the various AES modes, (e.g., EAX, XCBC, XTS, etc.) use the same underlying AES implementation as the approved AES Certs. #C1810, and #A661.

During operation, the module can switch service by service between an Approved mode of operation and a non-Approved mode of operation. The module will transition to the non- Approved mode of operation when one of the above non-Approved security functions is utilized in lieu of an Approved one. The module can transition back to the Approved mode of operation by utilizing an Approved security function.

4. Ports and Interfaces

The physical ports of the module are provided by the general-purpose computer on which the module is installed. The logical interfaces are defined as the API of the cryptographic module. The module's API supports the following logical interfaces: data input, data output, control input, and status output.

Table 4 - Logical Interface Mapping

FIPS 140-2 INTERFACE	Logical Interface
Data Input	Input parameters of API function calls
Data Output	Input parameters of API function calls
Control Output	API Function Calls
Status Output	For FIPS mode, function calls returning status information and return codes provided by API function calls
Power	None

5. Identification and Authentication Policy

Assumption of Roles

The Mocana Cryptographic Suite B Module shall support two distinct roles (User and Cryptographic Officer). The cryptographic module does not provide any identification or authentication methods of its own. The Cryptographic Officer and the User roles are implicitly assumed based on the service requested.

Table 5 - Roles and Required Identification and Authentication

Role	Type of Authentication	Authentication Data
User	N/A	N/A
Cryptographic Officer	N/A	N/A

6. Access Control Policy

Roles and Services

Table 6 - Services Authorized for Use in the Approved Mode of Operation

Role	Authorized Services
User	• Self-tests • Show Status • Read Version
Cryptographic Officer	• DH Key Generation • DH Key Exchange • ECC CDH Key Exchange • ECC CDH Key Generation • RSA Key Generation • RSA Signature Generation • RSA Signature Verification • RSA Key Wrapping Encryption • RSA Key Wrapping Decryption • DSA Key Generation • DSA Signature Generation • DSA Signature Verification • ECDSA Key Generation • ECDSA Key Verification • ECDSA Signature Generation • ECDSA Signature Verification • AES Encryption • AES Decryption • AES Message Authentication Code • Triple-DES Encryption • Triple-DES Decryption • SHA-1 • SHA-224/SHA-256 • SHA-384/SHA-512 • HMAC-SHA1 Message Authentication Code • HMAC-SHA224/256 Message Authentication Code • HMAC-SHA384/512 Message Authentication Code • AES-CTR DRBG Random Number Generation • HMAC-KDF Key Derivation • Key Destruction

Other Services

Table 7 - Services Authorized for Use in the non-Approved Mode of Operation

Role	Authorized Services
User	• Self-tests • Show Status • Read Version
Cryptographic-Officer	• DH Key Generation • DH Key Exchange • ECC CDH Key Exchange • ECC CDH Key Generation • RSA Key Generation • RSA Signature Generation • RSA Signature Verification • DES Encryption • DES Decryption • AES Message Authentication Code • MD2 Hash • MD4 Hash • MD5 Hash • AES EAX Encryption • AES EAX Decryption • AES XCBC Encryption • AES XCBC Decryption • RSA PKCS #1 v2.1 RSAES-OAEP Encryption • RSA PKCS #1 v2.1 RSAES-OAEP Decryption • FIPS 186-2 Random Number Generation • HMAC (with Non-Approved algorithms and key lengths) • AES GCM 4K and GCM 64K and other AES implementation (AES GCM 256K) • Triple-DES 2 key • DSA Sig Gen 2048/N=224 using SHA-1 • HMAC-KDF Key Derivation (with Non-Approved algorithms and key lengths)

The cryptographic module supports the following service that does not require an operator to assume an authorized role:

- Self-tests: This service executes the suite of self-tests required by FIPS 140-2. It is invoked by reloading the library into executable memory.

Definition of Critical Security Parameters (CSPs)

The following are CSPs that may be contained in the module:

Table 8 - CSP Information

Key	Description/Usage	Generation	Storage	Entry / Output	Destruction
DH Private Components	Used to derive the secret session key during DH key agreement protocol	Internally using the AES-CTR DRBG	Temporarily in volatile RAM (Plaintext)	N/A	An application program which uses the API may destroy the key. The Key Destruction service zeroizes this CSP.
ECC CDH Private Components	Used to derive the secret session key during ECC CDH key agreement protocol	Internally using the AES-CTR DRBG	Temporarily in volatile RAM (Plaintext)	N/A	An application program which uses the API may destroy the key. The Key Destruction service zeroizes this CSP.
DRBG Entropy Input	Used to seed the DRBG for key generation	Externally generated	Temporarily in volatile RAM (Plaintext)	Entry: Plaintext	Automatically after use
V and Key DRBG values	Used by the DRBG to generate random bits	Internally generated.	Temporarily in volatile RAM (Plaintext)	Entry: N/A Output: N/A	Automatically after use
RSA Private Key	Used to create RSA digital signatures	May be generated internally using the AES-CTR DRBG or generated externally	Temporarily in volatile RAM (Plaintext)	Entry: Plaintext if generated externally Output: Plaintext	An application program which uses the API may destroy the key. The Key Destruction service zeroizes this CSP.
RSA Key Wrapping Private Key	Used for RSA Key Wrapping decryption operation	May be generated internally using the AES-CTR DRBG or generated externally	Temporarily in volatile RAM (Plaintext)	Entry: Plaintext if generated externally Output: Plaintext	An application program which uses the API may destroy the key. The Key Destruction service zeroizes this CSP.

Key	Description/Usage	Generation	Storage	Entry / Output	Destruction
DSA Private Key	Used to create DSA digital signatures	May be generated internally using the AES-CTR DRBG or generated externally	Temporarily in volatile RAM (Plaintext)	Entry: Plaintext if generated externally Output: Plaintext	An application program which uses the API may destroy the key. The Key Destruction service zeroizes this CSP.
ECDSA Private Key	Used to create ECDSA digital signatures	May be generated internally using the AES-CTR DRBG or generated externally	Temporarily in volatile RAM (Plaintext)	Entry: Plaintext if generated externally Output: Plaintext	An application program which uses the API may destroy the key. The Key Destruction service zeroizes this CSP.
TDES Key	Used during Triple-DES encryption and decryption	Externally.	Temporarily in volatile RAM (Plaintext)	Entry: Plaintext Output: N/A	An application program which uses the API may destroy the key. The Key Destruction service zeroizes this CSP.
AES Keys	Used during AES encryption, decryption, CMAC and GMAC operations	Externally.	Temporarily in volatile RAM (Plaintext)	Entry: Plaintext Output: N/A	An application program which uses the API may destroy the key. The Key Destruction service zeroizes this CSP.
HMAC Keys	160-bit keys used during HMAC- SHA-1, 224, 256, 384, 512 operations	Externally.	Temporarily in volatile RAM (Plaintext)	Entry: Plaintext Output: N/A	An application program which uses the API may destroy the key. The Key Destruction service zeroizes this CSP.
HMAC Key Derivation	Used in deriving other keys per SP 800-108 with HMAC-SHA2-224, 256, 384, 512 operations.	Externally.	Temporarily in volatile RAM (Plaintext)	Entry: Plaintext Output: N/A	An application program which uses the API may destroy the key. The Key Destruction service zeroizes this CSP.

Notes:

-Key Entry and Output refers to keys crossing the logical boundary of the cryptographic module and not the physical boundary of the general-purpose computer.

Definition of Public Keys

The following are the public keys contained in the module:

Table 9 - Public Key Information

Key	Description/Usage	Generation	Storage	Entry/Output
DH Public Components	Used to derive the secret session key during DH key agreement protocol	Internally using the AES-CTR DRBG	Temporarily in volatile RAM	Entry: Receive Client Public Component during DH exchange. Output: Transmit Host Public Component during DH exchange
ECC CDH Public Components	Used to derive the secret session key during ECC CDH key agreement protocol	Internally using the AES-CTR DRBG	Temporarily in volatile RAM	Entry: Receive Client Public Component during DH exchange. Output: Transmit Host Public Component during DH exchange
RSA Public Key	Used to verify RSA signatures	May be generated internally using the AES-CTR DRBG or generated externally	Temporarily in volatile RAM	Input: Plaintext if generated externally t Output: Plaintext
RSA Key Wrapping Public Key	Used for RSA Key Wrapping encryption operation	May be generated internally using the AES-CTR DRBG or generated externally	Temporarily in volatile RAM	Input: Plaintext if generated externally Output: Plaintext
DSA Public Key	Used to verify DSA signatures	May be generated internally using the AES-CTR DRBG or generated externally	Temporarily in volatile RAM	Input: Plaintext if generated externally Output: Plaintext
ECDSA Public Key	Used to verify ECDSA signatures	May be generated internally using the AES-CTR DRBG or generated externally	Temporarily in volatile RAM	Input: Plaintext if generated externally Output: Plaintext

Definition of CSPs Modes of Access

Table 10 defines the relationship between access to CSPs and the different module services.

Table 10 - CSP Access Rights within Roles & Services

Role		Service	Cryptographic Keys and CSPs Access Operation
C.O.	**User**		
X		DH Key Generation	Use DH Parameters Generate DH Key pair
X		DH Key Exchange	Use DH Private Component Generate DH shared secret
X		ECC CDH Key Exchange	Use ECC CDH Private Component Generate ECC CDH shared secret
X		ECC CDH Key Generation	Use ECC CDH Parameters Generate ECC CDH Key pair
X		RSA Key Generation	Generate RSA Public/Private Key pair
X		RSA Signature Generation	Use RSA Private Key Generate RSA Signature
X		RSA Signature Verification	Use RSA Public Key Verify RSA Signature
X		RSA Key Wrapping Encryption	Use RSA Public Key Performs Key Wrapping Encryption
X		RSA Key Wrapping Decryption	Use RSA Private Key Performs Key Wrapping Decryption
X		DSA Key Generation	Generate DSA Key Pair for Signature Generation/Verification
X		DSA Signature Generation	Use DSA Private Key Generate DSA Signature
X		DSA Signature Verification	Use DSA Public Key Verify DSA Signature
X		ECDSA Key Generation	Generate ECDSA Key Pair for Signature Generation/Verification
X		ECDSA Key Verification	ECDSA Key Verification
X		ECDSA Signature Generation	Use DSA Private Key Generate ECDSA Signature

X		ECDSA Signature Verification	Use ECDSA Public Key Verify ECDSA Signature
X		AES Encryption	Use AES Key

Role		Service	Cryptographic Keys and CSPs Access Operation
C.O.	User		
X		AES Decryption	Use AES Key
X		AES Message Authentication Code	Use AES Key
X		Triple-DES Encryption	Use Triple-DES Key
X		Triple-DES Decryption	Use Triple-DES Key
X		SHA-1	Generate SHA-1 Output; no CSP access
X		SHA-224/256	Generate SHA-224/256 Output; no CSP access
X		SHA-384/512	Generate SHA-384/512 Output; no CSP access
X		HMAC-SHA-1 Message Authentication Code	Use HMAC-SHA-1 Key Generate HMAC-SHA-1 Output
X		HMAC-SHA-224/256 Message Authentication Code	Use HMAC-SHA-224/256 Key Generate HMAC-SHA-224/256 Output
X		HMAC-SHA-384/512 Message Authentication Code	Use HMAC-SHA-384/512 Key Generate HMAC-SHA-384/512 Output
X		AES-CTR DRBG Random Number Generation	Use V and Key values to generate random number Destroy V and Key values after use
X		HMAC-KDF Key Derivation Function	Use key material Generate HMAC-KDF Output
X		Key Destruction	Destroy All CSPs
	X	Show Status	N/A
	X	Self-Tests	N/A
	X	Read Version	N/A

7. Operational Environment

The FIPS 140-2 Area 6 Operational Environment requirements are applicable because the Mocana Cryptographic Suite B Module operates in a modifiable operational environment.

Please refer to Table 1 for a list of environments for which operational testing of the module was performed.

Integrity Check at Application Start

During the load of the shared object, the integrity check of the library code and constants occurs in the module startup function. It verifies the integrity by executing the HMAC-SHA 256 fingerprint algorithm on the shared library .so file and comparing the result with the signature file. This integrity check is performed as part of the function FIPS_powerupSelfTest(). This function is called automatically by the host O/S upon loading the shared object into memory via the code snippet below.

```
#ifdef __
ENABLE_MOCANA_FIPS_LIB_CONSTRUCTOR___
static void FIPS_constructor() __
attribute__((constructor));
void FIPS_constructor()
{
     FIPS_powerupSelfTest();
}
#endif
```

Figure 3: Code Example for Self-Test

8. Security Rules

The Mocana Cryptographic Suite B Module design corresponds to the following security rules. This section documents the security rules enforced by the cryptographic module to implement the security requirements of this FIPS 140-2 Level 1 module.

1. The cryptographic module provides two (2) distinct roles. These are the User role and the Cryptographic Officer role.
2. The cryptographic module does not provide any operator authentication.
3. The cryptographic module shall encrypt/decrypt message traffic using the Triple-DES or AES algorithms.
4. The cryptographic module shall perform the following self-tests:

Table 11 - Power-up Self-Tests

Ty	Detail
Software Integrity Check	• HMAC-SHA-256
Known Answer Tests	• AES-ECB-256, CBC-256, OFB-128, CFB-128, CCM-128, and CTR-256 encrypt/decrypt (AES Certs. #A661 and #C1810) • AES-GCM (4K and 64K), GMAC encrypt/decrypt • AES-XTS-256 encrypt/decrypt (AES Certs. #A661 and #C1810) • AES-CMAC-128 generation/verification (AES Certs. #A661 and #C1810) • Triple-DES CBC encrypt/decrypt • HMAC-SHA-1 • HMAC-SHA-224 • HMAC-SHA-256 • HMAC-SHA-384 • HMAC-SHA-512 • SHA-1 • SHA-224 • SHA-256 • SHA-384 • SHA-512 • RSA Sign and Verify KATs (2048-bit modulus, no hash) • AES-CTR DRBG (including SP800-90A Health Checks • ECC CDH P-256 • DH 2048 bit • KBKDF (HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512)
Pairwise Consistency Tests	• DSA (L=2048/N=256) • ECDSA (on EC-P224 curve)

Table 12 - Conditional Self-Tests

Type	Detail
Pairwise Consistency Tests	• DSA • ECDSA • RSA
Continuous RNG Tests	• AES-CTR DRBG Continuous Test

5. At any time, the operator shall be capable of commanding the module to perform the power-up self-tests by reloading the cryptographic module into memory.
6. The cryptographic module is available to perform services only after successfully completing the power-up self-tests.
7. Data output shall be inhibited during key generation, self-tests, zeroization, and error states. Because the logical interface is defined as the API of the crypto module and the API of the crypto module is single-threaded, key generation or zeroization must be complete before the API returns control to the calling application.
8. Status information shall not contain CSPs or sensitive data that if misused could lead to a compromise of the module.
9. In the event of a self-test failure, the module will enter an error state and a specific error code will be returned indicating which self-test or conditional test has failed. The module will not provide any cryptographic services while in this state.
10. The operating system is restricted to a single operator mode of operation (i.e., concurrent operators are explicitly excluded). The application that makes calls to the modules is the single user of the modules, even when the application is serving multiple clients.
11. The calling application of the module shall use entropy sources that meet the security strength required for the random bit generation mechanism. A minimum of 112 bits of entropy must be provided by the calling application.
12. All algorithms of Table 7 are not allowed for use in the FIPS Approved mode of operation. When these algorithms are used, the module is no longer operating in the FIPS Approved mode of operation. It is the responsibility of the consuming application to zeroize all keys and CSPs prior to and after utilizing these non-Approved algorithms. CSPs shall not be shared between the Approved and non-Approved modes of operation.
13. The calling application of the module must ensure that the same Triple-DES key is not used to encrypt more than 2^{16} 64-bit blocks of data (IG A.13).
14. The calling application of the module must generate RSA key pairs of at least 2048 bits to operate in Approved Mode.

15. The calling application of the module must generate ECC keys using a P-Curve with a security strength of at least 112 bits to operate in the Approved Mode of operation.

9. Physical Security

The FIPS 140-2 Area 5 Physical Security requirements are not applicable because the Mocana Cryptographic Module is software only.

10. Mitigation of Other Attacks Policy

The module has not been designed to mitigate any specific attacks outside the scope of FIPS 140-2 requirements.

11. Key Management

The application that uses the module is responsible for appropriate destruction and zeroization of the keys. The library provides API calls for key allocation and destruction. These API calls overwrite the memory occupied by the key information with zeros before that memory is de-allocated. See Key Destruction Service paragraph below.

Key/CSP Authorized Access and Use

An authorized application acting as the User has access to all key data generated during the operation of the module.

Key/CSP Storage

Private and public keys are provided to the module by the calling process and are destroyed when released by the appropriate API function calls. The module does not perform persistent storage of keys.

Key/CSP Zeroization

The application is responsible for calling the appropriate destruction functions from the API. These functions overwrite the memory with zeros and de-allocate the memory. In case of abnormal termination, the Linux kernel overwrites the keys in physical memory before the physical memory is allocated to another process.

Key Destruction Service

There is a context structure associated with every cryptographic algorithm available in this module. Context structures hold sensitive information such as cryptographic keys. These context structures must be destroyed via respective API calls when the application software no longer needs to use a specific algorithm any more. This API call will zeroize all sensitive information including cryptographic keys before freeing the dynamically allocated memory. This will occur while the application process is still in memory, but no longer needs the specific algorithm, which sufficiently protects the keys from compromise. See the *Mocana Cryptographic API Reference* for additional information.

Random Number Generation

The module implements a CTR-based DRBG. The DRBG generates blocks of random numbers with more than 15 bits. During each generation of random numbers, the newly created bits are compared with the previously created bits. If they are not the same, then the newly created bits are saved to be used in a subsequent bit generation comparison test, however, if they are the same, then the module enters the error state.

The module accepts input from entropy sources external to the cryptographic boundary for use as seed material for the module's Approved DRBG's. External entropy can be added via several APIs available to the crypto-module client application:

MOCANA_addEntropyBit () and MOCANA_addEntropy32Bits().

Module users (the calling applications) shall use entropy sources that meet the security strength required for the random number generation mechanism.

12. Guidance

Cryptographic Officer Guidance

The operating system running the Mocana Cryptographic Suite B Module must be configured in a single-user mode of operation.

The Cryptographic Officer will install the crypto module and associated signature of the module into the proper location within the computer system. For example, the shared memory library and signature file may be installed in the /usr/local/lib directory, which is protected by Linux access control mechanisms. The module is protected from modification by the integrity self-test performed during startup. The module is initialized by the operating system upon loading the module (kernel module or shared library) into memory for use by calling applications.

User Guidance

The module must be operated in FIPS Approved mode to ensure that FIPS 140-2 validated cryptographic algorithms and security functions are used.

13. Definitions and Acronyms

Table 13 - Acronyms and Terms

Acronym	Term
AES	Advanced Encryption Standard
API	Application Program Interface
CKG	Cryptographic Key Generation
CO	Cryptographic Officer
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
DES	Data Encryption Standard
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Algorithm
ECC CDH	Elliptic Curve Cryptography Cofactor Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FIPS	Federal Information Processing Standard
HMAC	Keyed-Hash Message Authentication Code
KDF	Key Derivation Function
KVM	Kernel-based Virtual Machine
PAA	Processor Algorithm Accelerators
RAM	Random Access Memory
RNG	Random Number Generator
RSA	Rivest, Shamir and Adleman Algorithm
TDES	Triple-DES
SHA	Secure Hash Algorithm
SO	Shared Object