Ezurio

Summit Linux FIPS Core Crypto Module

FIPS 140-3 Non-Proprietary Security Policy

Document Version: 1.1

Last Modified: 04/04/2025

Prepared by:

atsec information security corporation

4516 Seton Center Pkwy, Suite 250

Austin, TX 78759

www.atsec.com

Table of Contents

List of Tables

List of Figures

1 General

1.1 Overview

This document is the non-proprietary FIPS 140-3 Security Policy for the Summit Linux FIPS Core Crypto Module, firmware version 11.0. It contains the security rules under which the module must be operated and describes how this module meets the requirements as specified in FIPS 140-3 (Federal Information Processing Standards Publication 140-3) for a Security Level 1 module.

This Security Policy contains non-proprietary information. All other documentation submitted for FIPS 140-3 conformance testing and validation is proprietary and is releasable only under appropriate non-disclosure agreements.

1.1.1 How this Security Policy was prepared

The vendor has provided the non-proprietary Security Policy of the cryptographic module, which was further consolidated into this document by atsec information security together with other vendor-supplied documentation. In preparing the Security Policy document, the laboratory formatted the vendor-supplied documentation for consolidation without altering the technical statements therein contained. The further refining of the Security Policy document was conducted iteratively throughout the conformance testing, wherein the Security Policy was submitted to the vendor, who would then edit, modify, and add technical contents. The vendor would also supply additional documentation, which the laboratory formatted into the existing Security Policy, and resubmitted to the vendor for their final editing.

1.2 Security Levels

Section	Title	Security Level
1	General	1
2	Cryptographic module specification	1
3	Cryptographic module interfaces	1
4	Roles, services, and authentication	1
5	Software/Firmware security	1
6	Operational environment	1
7	Physical security	1
8	Non-invasive security	N/A
9	Sensitive security parameter management	1
10	Self-tests	1
11	Life-cycle assurance	1
12	Mitigation of other attacks	1
	Overall Level	1

Table 1: Security Levels

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

The Summit Linux FIPS Core Crypto Module (hereafter referred to as the “module”) is a firmware module supporting FIPS 140-3 Approved cryptographic algorithms. The module is composed by firmware components comprised of a kernel and OpenSSL library. These firmware components provide a C language application program interface (API) for use by other processes that require cryptographic functionality.

The module offers approved cryptographic functions in the approved mode for, among other uses:

- Algorithms for use in the Wi-Fi protocols CCMP and GCMP.
- Algorithms for use in the TLS protocol.
- Encryption and decryption for data at rest.

Module Type: Firmware

Module Embodiment: MultiChipStand

Cryptographic Boundary:

The Summit Linux FIPS Core Crypto Module is defined as a firmware, Multi-Chip Standalone module per the requirements within FIPS 140-3. The cryptographic boundary of the module consists of the firmware component files (kernel, FIPS Provider, and the fipscheck integrity test tool) and the integrity test HMAC files. For easier readability in the rest of the document “OpenSSL FIPS provider” has been shortened to “FIPS provider”.

Tested Operational Environment’s Physical Perimeter (TOEPP):

Figure 1 below depicts the physical representations of the tested platforms as the top and bottom view of the circuit board. The tested environments are further described in Section 6.

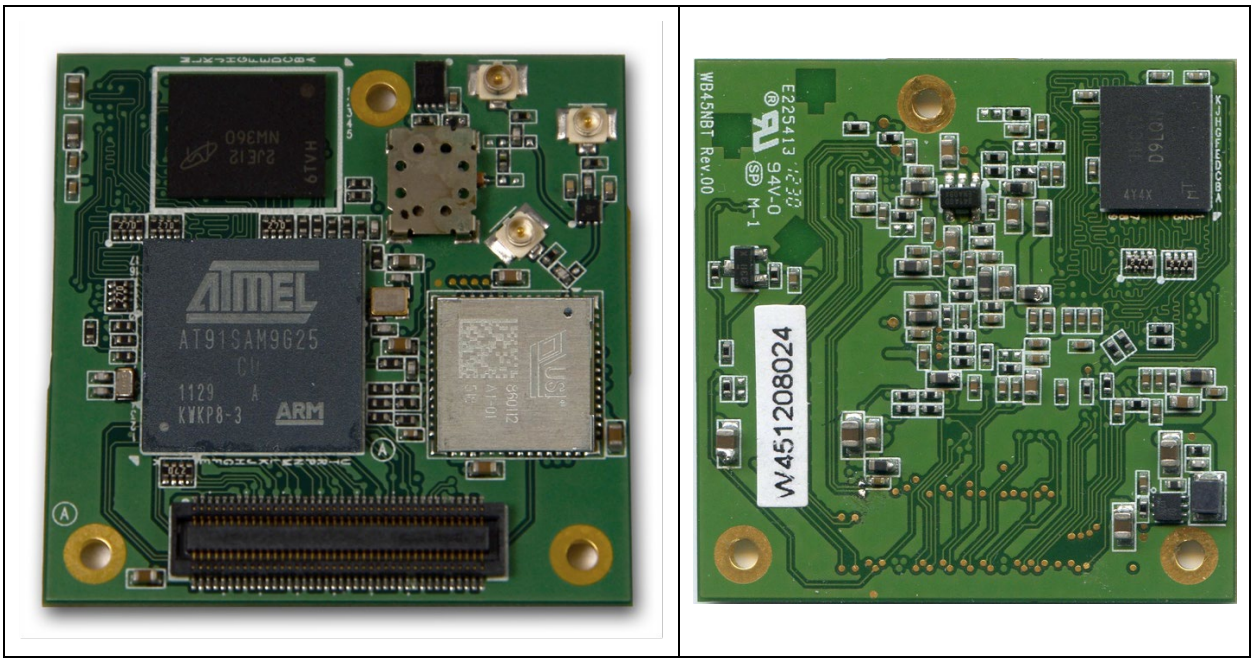

Figure 1: Physical configurations of the tested platform (top and bottom in order).

Figure 2 below shows the block diagram of the module. The cryptographic boundary is indicated with yellow blocks, distributed among firmware components. Blocks of another color do not belong to the cryptographic boundary. Users of the module interact through the API that are the logical interfaces data input, data output, control input, status output. A dotted line encompasses the module's components that interface through the API.

In Figure 2, users of the module are exemplified by applications. These applications may reside within the NAND Flash memory or may reside outside (but still within the physical perimeter), always interacting with the module's API.

The physical perimeter of the module is defined as the perimeter of the circuit board on which the module is installed. The filesystem and operating system reside on NAND Flash memory within the physical perimeter.

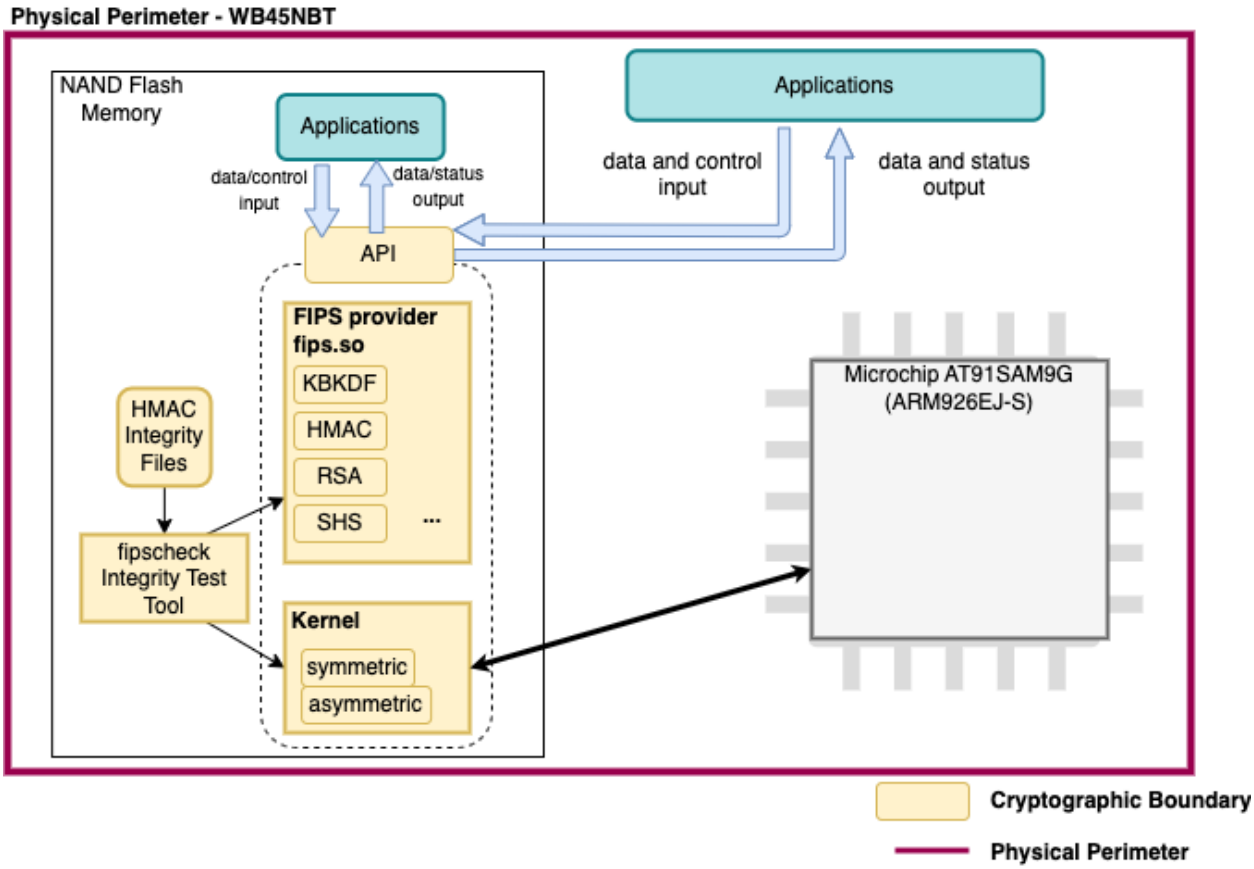

Figure 2: Block Diagram

2.2 Tested and Vendor Affirmed Module Version and Identification

Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets):

Package or File Name	Software/ Firmware Version	Features	Integrity Test
Image.lzma, fipscheck (application and library), fips.so	11.0	N/A	HMAC-SHA-256

Table 2: Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets)

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Summit Linux 11.0	Laird WB5NBT wireless bridge	Microchip AT91SAM9G (ARM926EJ-S), ARMv5-based	No	N/A	11.0

Table 3: Tested Operational Environments - Software, Firmware, Hybrid

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.

2.3 Excluded Components

There are no components within the cryptographic boundary excluded from the FIPS 140-3 requirements.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved mode	Automatically entered whenever an approved service is requested	Approved	Equivalent to the indicator of the requested service as defined in section 4.3
Non-approved mode	Automatically entered whenever a non-approved service is requested	Non-Approved	

Table 4: Modes List and Description

After passing all pre-operational self-tests and cryptographic algorithm self-tests executed on start-up, the module automatically transitions to the approved mode.

Mode Change Instructions and Status:

The module automatically switches between the approved and non-approved modes depending on the services requested by the operator. The status indicator of the mode of operation is equivalent to the indicator of the service that was requested.

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A4712, A4716, A5002, A5003, A5004	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC-CS1	A5002, A5003, A5004	Direction - decrypt, encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC-CS2	A5002, A5003, A5004	Direction - decrypt, encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC-CS3	A4714, A4718, A5002, A5003, A5004	Direction - decrypt, encrypt Key Length - 128, 192, 256	SP 800-38A

Algorithm	CAVP Cert	Properties	Reference
AES-CCM	A4712, A4716, A5002, A5003, A5004	Key Length - 128, 192, 256	SP 800-38C
AES-CFB1	A5002, A5003, A5004	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB128	A5002, A5003, A5004	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A5002, A5003, A5004	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A4712, A4716, A5002, A5003, A5004	Direction - Generation, Verification Key Length - 128, 192, 256	SP 800-38B
AES-CTR	A4712, A4716, A5002, A5003, A5004	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A4711, A4712, A4715, A4716, A4717, A5002, A5003, A5004, A5019	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A4712, A4715, A4717	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 192, 256	SP 800-38D
AES-GCM	A5005, A5006, A5007, A5008	Direction - Decrypt, Encrypt IV Generation - External, Internal IV Generation Mode - 8.2.1, 8.2.2 Key Length - 128, 192, 256	SP 800-38D
AES-GMAC	A4712, A5005, A5006, A5007, A5008	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 192, 256	SP 800-38D
AES-KW	A5002, A5003, A5004	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
AES-KWP	A5002, A5003, A5004	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
AES-OFB	A5002, A5003, A5004	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-XTS Testing Revision 2.0	A4712, A4716, A5002, A5003, A5004	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38E
Counter DRBG	A4711, A4712, A4715, A4717	Prediction Resistance - No, Yes Mode - AES-128, AES-192, AES-256 Derivation Function Enabled - Yes	SP 800-90A Rev. 1
Counter DRBG	A5015	Prediction Resistance - No, Yes Mode - AES-128, AES-192, AES-256 Derivation Function Enabled - No, Yes	SP 800-90A Rev. 1
ECDSA KeyGen (FIPS186-5)	A4711	Curve - P-256, P-384 Secret Generation Mode - testing candidates	FIPS 186-5
ECDSA KeyGen (FIPS186-5)	A5009, A5018	Curve - P-224, P-256, P-384, P-521 Secret Generation Mode - testing candidates	FIPS 186-5

Algorithm	CAVP Cert	Properties	Reference
ECDSA KeyVer (FIPS186-5)	A5009, A5018	Curve - P-224, P-256, P-384, P-521	FIPS 186-5
ECDSA SigGen (FIPS186-5)	A5009, A5018	Curve - P-224, P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 Component - No	FIPS 186-5
ECDSA SigGen (FIPS186-5)	A5011, A5020	Curve - P-224, P-256, P-384, P-521 Hash Algorithm - SHA3-224, SHA3-256, SHA3-384, SHA3-512 Component - No	FIPS 186-5
ECDSA SigVer (FIPS186-5)	A5009, A5018	Curve - P-224, P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	FIPS 186-5
ECDSA SigVer (FIPS186-5)	A5011, A5020	Curve - P-224, P-256, P-384, P-521 Hash Algorithm - SHA3-224, SHA3-256, SHA3-384, SHA3-512	FIPS 186-5
EDDSA KeyGen	A5016	Curve - ED-25519, ED-448	FIPS 186-5
EDDSA SigGen	A5016	Curve - ED-25519, ED-448	FIPS 186-5
EDDSA SigVer	A5016	Curve - ED-25519, ED-448	FIPS 186-5
Hash DRBG	A5015	Prediction Resistance - No, Yes Mode - SHA-1, SHA2-256, SHA2-512	SP 800-90A Rev. 1
HMAC DRBG	A5015	Prediction Resistance - No, Yes Mode - SHA-1, SHA2-256, SHA2-512	SP 800-90A Rev. 1
HMAC-SHA-1	A5009, A5018	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A5009, A5018	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A5009, A5010, A5018	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A5009, A5018	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A5009, A5018	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512/224	A5009, A5018	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512/256	A5009, A5018	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA3-224	A5011, A5020	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA3-256	A5011, A5020	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA3-384	A5011, A5020	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA3-512	A5011, A5020	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A4711	Domain Parameter Generation Methods - P-256, P-384	SP 800-56A Rev. 3

Algorithm	CAVP Cert	Properties	Reference
		Scheme - ephemeralUnified - KAS Role - initiator, responder	
KAS-ECC-SSC Sp800-56Ar3	A5009, A5018	Domain Parameter Generation Methods - P-224, P-256, P-384, P-521 Scheme - ephemeralUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3
KAS-FFC-SSC Sp800-56Ar3	A5014	Domain Parameter Generation Methods - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 Scheme - dhEphem - KAS Role - initiator, responder	SP 800-56A Rev. 3
KAS-IFC-SSC	A5009, A5018	Modulo - 2048, 3072, 4096, 6144, 8192 Key Generation Methods - rsakpg1-basic, rsakpg1-crt, rsakpg1-prime-factor, rsakpg2-basic, rsakpg2-crt, rsakpg2-prime-factor Scheme - KAS1 - KAS Role - initiator, responder	SP 800-56A Rev. 3
KDA HKDF Sp800-56Cr1	A5013	Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-2048 Increment 8 HMAC Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512	SP 800-56C Rev. 2
KDA OneStep SP800-56Cr2	A5012	Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-2048 Increment 8	SP 800-56C Rev. 2
KDA TwoStep SP800-56Cr2	A5012	MAC Salting Methods - default, random KDF Mode - feedback Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-2048 Increment 8	SP 800-56C Rev. 2
KDF ANS 9.42 (CVL)	A5009, A5018	KDF Type - DER Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256 Key Data Length - Key Data Length: 8-4096 Increment 8	SP 800-135 Rev. 1
KDF ANS 9.42 (CVL)	A5011, A5020	KDF Type - DER Hash Algorithm - SHA3-224, SHA3-256, SHA3-384, SHA3-512 Key Data Length - Key Data Length: 8-4096 Increment 8	SP 800-135 Rev. 1
KDF ANS 9.63 (CVL)	A5009, A5018	Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	SP 800-135 Rev. 1

Algorithm	CAVP Cert	Properties	Reference
		Key Data Length - Key Data Length: 128-4096 Increment 8	
KDF KMAC Sp800-108r1	A5017	Derived Key Length - Derived Key Length: 112-4096 Increment 8	SP 800-108 Rev. 1
KDF SP800-108	A5017	KDF Mode - Counter, Feedback Supported Lengths - Supported Lengths: 112, 128, 776, 3456, 4096	SP 800-108 Rev. 1
KDF SSH (CVL)	A5019	Cipher - AES-128, AES-192, AES-256, TDES Hash Algorithm - SHA-1, SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1
KDF TLS (CVL)	A5009, A5018	TLS Version - v1.0/1.1	SP 800-135 Rev. 1
KMAC-128	A5011, A5020	Message Length - Message Length: 0-65536 Increment 8 Key Data Length - Key Data Length: 128-1024 Increment 8	SP 800-185
KMAC-256	A5011, A5020	Message Length - Message Length: 0-65536 Increment 8 Key Data Length - Key Data Length: 128-1024 Increment 8	SP 800-185
KTS-IFC	A5009, A5018	Modulo - 2048, 3072, 4096, 6144, 8192 Key Generation Methods - rsakpg1-basic, rsakpg1-crt, rsakpg1-prime-factor, rsakpg2-basic, rsakpg2-crt, rsakpg2-prime-factor Scheme - KTS-OAEP-basic - KAS Role - initiator, responder Key Transport Method - Key Length - 768	SP 800-56B Rev. 2
PBKDF	A5009, A5011, A5018, A5020	Iteration Count - Iteration Count: 1000-10000 Increment 1 Password Length - Password Length: 14-128 Increment 1	SP 800-132
RSA KeyGen (FIPS186-5)	A5009, A5018	Key Generation Mode - probableWithProbableAux Modulo - 2048, 3072, 4096 Primality Tests - 2powSecStr Private Key Format - standard	FIPS 186-5
RSA SigGen (FIPS186-5)	A5009, A5018	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
RSA SigVer (FIPS186-5)	A5009, A5018	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
Safe Primes Key Generation	A5014	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	SP 800-56A Rev. 3
Safe Primes Key Verification	A5014	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	SP 800-56A Rev. 3
SHA-1	A5009, A5018	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4

Algorithm	CAVP Cert	Properties	Reference
SHA2-224	A4711, A4712, A4716, A5009, A5018	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-256	A4711, A4712, A4716, A5009, A5010, A5018	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A4711, A4712, A4716, A5009, A5018	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A4711, A4712, A4716, A5009, A5018	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512/224	A5009, A5018	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512/256	A5009, A5018	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA3-224	A4713, A5011, A5020	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-256	A4713, A5011, A5020	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-384	A4713, A5011, A5020	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-512	A4713, A5011, A5020	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHAKE-128	A5011, A5020	Output Length - Output Length: 16-65536 Increment 8	FIPS 202
SHAKE-256	A5011, A5020	Output Length - Output Length: 16-65536 Increment 8	FIPS 202
TLS v1.2 KDF RFC7627 (CVL)	A5009, A5018	Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1
TLS v1.3 KDF (CVL)	A5013	HMAC Algorithm - SHA2-256, SHA2-384 KDF Running Modes - DHE, PSK, PSK-DHE	SP 800-135 Rev. 1

Table 5: Approved Algorithms

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
Kernel ECDSA Key Generation	Curves:P-256, P-384 (128, 192 bits)	Summit Linux (ECDH_C)	FIPS 186-5, SP 800-133rev2 Section 5.1, 5.2
FIPS provider Safe Primes Key Generation	Safe Primes:MODP-2048, MODP-3072, MODP-4096, MODP-6144, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 with 112-200 bits of key strength	FIPS provider (FFC_DH)	FIPS 186-5, SP 800-133rev2 Section 5.1, 5.2
FIPS provider EDDSA Key Generation	Curves:ED-25519, ED-448 (128, 224 bits)	FIPS provider (EDDSA_3_2)	FIPS 186-5, SP 800-133rev2 Section 5.1
FIPS provider RSA Key Generation	Keys:2048, 3072, 4096 (112, 128, 149 bits)	FIPS provider (SHA_ASM)	FIPS 186-5, SP 800-133rev2 Section 5.1, 5.2

Name	Properties	Implementation	Reference
FIPS provider ECDSA Key Generation	Curves:P-224, P-256, P-384, P-512 (112, 128, 192, 256 bits)	FIPS provider (SSH_ASM)	FIPS 186-5, SP 800-133rev2 Section 5.1, 5.2

Table 6: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms with No Security Claimed:

N/A for this module.

Non-Approved, Not Allowed Algorithms:

Name	Use and Function
FIPS provider PBKDF with salt length less than 128 bits	Key derivation
FIPS provider TLSv1.0 and TLSv1.1 KDF using EMS	Key derivation
FIPS provider TLSv1.2 KDF without using EMS	Key derivation

Table 7: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
Kernel AES-CCM (KTS-Wrap)	KTS-Wrap	Key Unwrapping, Key Unwrapping	Keys: 128, 192, 256-bit keys with 128, 192, 256 bits of key strength, respectively Compliance:Compliant with IG D.G	AES-CCM: (A4712, A4716)
Kernel AES-GCM (KTS-Wrap)	KTS-Wrap	Key Wrapping, Key Unwrapping	Keys:128, 192, 256-bit keys with 128, 192, 256 bits of key strength, respectively Compliance:Compliant with IG D.G	AES-GCM: (A4712, A4715, A4717)
Kernel AES CBC with HMAC	KTS-Wrap	Key Wrapping, Key Unwrapping	Keys:128, 192, 256-bit keys with 128, 192, 256 bits of key strength, respectively Compliance:Compliant with IG D.G	AES-CBC: (A4712, A4716) HMAC-SHA2-256: (A4711, A4712, A4716) HMAC-SHA2-384: (A4711, A4712, A4716)

Name	Type	Description	Properties	Algorithms
				HMAC-SHA2-512: (A4711, A4712, A4716)
Kernel AES CTR with HMAC	KTS-Wrap	Key Wrapping, Key Unwrapping	Keys:128, 192, 256-bit keys with 128, 192, 256 bits of key strength, respectively Compliance:Compliant with IG D.G	AES-CTR: (A4712, A4716) HMAC-SHA2-256: (A4711, A4712, A4716) HMAC-SHA2-384: (A4711, A4712, A4716) HMAC-SHA2-512: (A4711, A4712, A4716)
Kernel KAS-ECC-SSC	KAS-SSC	Shared Secret Computation	Curves:Curves : P-256, P-384 elliptic curves with 128 and 192 bits of key strength Compliance : Compliant with IG D.F scenario 2(1)	KAS-ECC-SSC Sp800-56Ar3: (A4711)
Kernel AES-ECB	BC-UnAuth	Encryption/Decryption	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-ECB: (A4711, A4712, A4715, A4716, A4717)
Kernel AES-CTR	BC-UnAuth	Encryption/Decryption	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CTR: (A4712, A4716)
Kernel AES-CBC	BC-UnAuth	Encryption/Decryption	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CBC: (A4712, A4716)
Kernel AES-CBC-CS3	BC-UnAuth	Encryption/Decryption	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CBC-CS3: (A4714, A4718)
Kernel AES-XTS	BC-UnAuth	Encryption/Decryption	Keys:128, 256 bits with 128 and 256 bits of key strength	AES-XTS Testing Revision 2.0: (A4712, A4716)

Name	Type	Description	Properties	Algorithms
Kernel AES-CCM (BC-Auth)	BC-Auth	Authenticated Encryption/Decryption	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CCM: (A4712, A4716)
Kernel AES-GCM (BC-Auth)	BC-Auth	Authenticated Encryption/Decryption	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-GCM: (A4712, A4715, A4717)
Kernel AES-CMAC	MAC	Message authentication code (MAC)	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CMAC: (A4712, A4716)
Kernel AES-GMAC	MAC	Message authentication code (MAC)	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-GMAC: (A4712)
Kernel Counter DRBG	DRBG	Random Number Generation	Compliance:Compliant with SP800-90ARev1	Counter DRBG: (A4711, A4712, A4715, A4717)
Kernel ECDSA Key Generation	CKG	Key Generation	Curves:P-256, P-384	ECDSA KeyGen (FIPS186-5): (A4711)
Kernel HMAC	MAC	Message authentication code (MAC)	Keys:112-256 bits with 112-256 bits of key strength	HMAC-SHA2-224: (A4711, A4712, A4716) HMAC-SHA2-256: (A4711, A4712, A4716) HMAC-SHA2-384: (A4711, A4712, A4716) HMAC-SHA2-512: (A4711, A4712, A4716) HMAC-SHA3-224: (A4713) HMAC-SHA3-256: (A4713) HMAC-SHA3-384: (A4713) HMAC-

Name	Type	Description	Properties	Algorithms
				SHA3-512: (A4713)
Kernel Hashes	SHA	Hashing		SHA2-224: (A4711, A4712, A4716) SHA2-256: (A4711, A4712, A4716) SHA2-384: (A4711, A4712, A4716) SHA2-512: (A4711, A4712, A4716) SHA3-224: (A4713) SHA3-256: (A4713) SHA3-384: (A4713) SHA3-512: (A4713)
FIPS provider AES-CCM (KTS-Wrap)	KTS-Wrap	Key Unwrapping, Key Unwrapping	Keys: 128, 192, 256-bit keys with 128, 192, 256 bits of key strength, respectively Compliance:Compliant with IG D.G	AES-CCM: (A5002, A5003, A5004)
FIPS provider AES-GCM (KTS-Wrap)	KTS-Wrap	Key Wrapping, Key Unwrapping	Keys:128, 192, 256-bit keys with 128, 192, 256 bits of key strength, respectively Compliance:Compliant with IG D.G	AES-GCM: (A5005, A5006, A5007, A5008)
FIPS provider KAS-IFC-SSC	KAS-SSC	Shared Secret Computation	Keys:2048, 3072, 4096, 6144, 8192-bit keys with 112-200 bits of key strength Compliance : Compliant with IG D.F scenario 1(1)	KAS-IFC-SSC: (A5009, A5018)
FIPS provider KTS-IFC	KTS-Encap	Key encapsulation, Key unencapsulation	Keys:2048, 3072, 4096, 6144, 8192-bit keys with 112-200 bits of key strength respectively Compliance:Compliant with IG D.G	KTS-IFC: (A5009, A5018)

Name	Type	Description	Properties	Algorithms
FIPS provider Safe Primes Key Generation	CKG	Key Generation	Groups:MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192	Safe Primes Key Generation: (A5014)
FIPS provider Safe Primes Key Verification	AsymKeyPair-KeyVer	Key Verification	Groups:MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192	Safe Primes Key Verification: (A5014)
FIPS provider KAS-FFC-SSC	KAS-SSC	Shared Secret Computation	Keys:2048, 3072, 4096, 6144, 8192-bit keys with 112-200 bits of key strength Compliance : Compliant with IG D.F scenario 2(1)	KAS-FFC-SSC Sp800-56Ar3: (A5014)
FIPS provider KAS-ECC-SSC	KAS-SSC	Shared Secret Computation	Curves:P-224, P-256, P-384, P-521 elliptic curves with 112-256 bits of key strength Compliance : Compliant with IG D.F scenario 2(1)	KAS-ECC-SSC Sp800-56Ar3: (A5009, A5018)
FIPS provider AES KW	KTS-Wrap	Key Wrapping, Key Unwrapping	Keys:128, 192, 256-bit keys with 128, 192, 256 bits of key strength, respectively Compliance:Compliant with IG D.G	AES-KW: (A5002, A5003, A5004)
FIPS provider AES KWP	KTS-Wrap	Key Wrapping, Key Unwrapping	Keys:128, 192, 256-bit keys with 128, 192, 256 bits of key strength, respectively Compliance:Compliant with IG D.G	AES-KWP: (A5002, A5003, A5004)
FIPS provider AES-ECB	BC-UnAuth	Encryption/Decryption	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-ECB: (A5002, A5003, A5004, A5019)
FIPS provider AES-CTR	BC-UnAuth	Encryption/Decryption	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CTR: (A5002, A5003, A5004)
FIPS provider AES-CBC	BC-UnAuth	Encryption/Decryption	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CBC: (A5002, A5003, A5004)

Name	Type	Description	Properties	Algorithms
FIPS provider AES-CBC-CS1	BC-UnAuth	Encryption/Decryption	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CBC-CS1: (A5002, A5003, A5004)
FIPS provider AES-CBC-CS2	BC-UnAuth	Encryption/Decryption	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CBC-CS2: (A5002, A5003, A5004)
FIPS provider AES-CBC-CS3	BC-UnAuth	Encryption/Decryption	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CBC-CS3: (A5002, A5003, A5004)
FIPS provider AES-CFB1	BC-UnAuth	Encryption/Decryption	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CFB1: (A5002, A5003, A5004)
FIPS provider AES-CFB8	BC-UnAuth	Encryption/Decryption	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CFB8: (A5002, A5003, A5004)
FIPS provider AES-CFB128	BC-UnAuth	Encryption/Decryption	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CFB128: (A5002, A5003, A5004)
FIPS provider AES-XTS	BC-UnAuth	Encryption/Decryption	Keys:128, 256 bits with 128 and 256 bits of key strength	AES-XTS Testing Revision 2.0: (A5002, A5003, A5004)
FIPS provider AES-CCM (BC-Auth)	BC-Auth	Authenticated Encryption/Decryption	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CCM: (A5002, A5003, A5004)
FIPS provider AES-GCM (BC-Auth)	BC-Auth	Authenticated Encryption/Decryption	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-GCM: (A5005, A5006, A5007, A5008)
FIPS provider AES-OFB	BC-UnAuth	Encryption/Decryption	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-OFB: (A5002, A5003, A5004)
FIPS provider AES-CMAC	MAC	Message authentication code (MAC)	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CMAC: (A5002, A5003, A5004)

Name	Type	Description	Properties	Algorithms
FIPS provider AES-GMAC	MAC	Message authentication code (MAC)	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-GMAC: (A5005, A5006, A5007, A5008)
FIPS provider Counter DRBG	DRBG	Random Number Generation	Compliance:Compliant with SP800-90ARev1	Counter DRBG: (A5015)
FIPS provider Hash DRBG	DRBG	Random Number Generation	Compliance:Compliant with SP800-90ARev1	Hash DRBG: (A5015)
FIPS provider HMAC DRBG	DRBG	Random Number Generation	Compliance:Compliant with SP800-90ARev1	HMAC DRBG: (A5015)
FIPS provider ECDSA Key Generation	CKG	Key Generation	Curves:P-224, P-256, P-384, P-521	ECDSA KeyGen (FIPS186-5): (A5009, A5018)
FIPS provider ECDSA Key Verification	AsymKeyPair-KeyVer	Key Verification	Curves:P-224, P-256, P-384, P-521	ECDSA KeyVer (FIPS186-5): (A5009, A5018)
FIPS provider ECDSA Signature Generation	DigSig-SigGen	Signature Generation	Curves:P-224, P-256, P-384, P-521	ECDSA SigGen (FIPS186-5): (A5009, A5011, A5018, A5020)
FIPS provider ECDSA Signature Verification	DigSig-SigVer	Signature Verification	Curves:P-224, P-256, P-384, P-521	ECDSA SigVer (FIPS186-5): (A5009, A5011, A5018, A5020)
FIPS provider EDDSA Key Generation	CKG	Key Generation	Curves:Ed25519, Ed448	EDDSA KeyGen: (A5016)
FIPS provider EDDSA Signature Generation	DigSig-SigGen	Signature Generation	Curves:Ed25519, Ed448	EDDSA SigGen: (A5016)
FIPS provider EDDSA	DigSig-SigVer	Signature Verification	Curves:Ed25519, Ed448	EDDSA SigVer: (A5016)

Name	Type	Description	Properties	Algorithms
Signature Verification				
FIPS provider RSA Key Generation	CKG	Key Generation	Keys:2048, 3072, 4096 keys with 112-150 bits of key strength respectively	RSA KeyGen (FIPS186-5): (A5009, A5018)
FIPS provider RSA Signature Generation	DigSig-SigGen	Signature Generation	Keys:2048, 3072, 4096 keys with 112-150 bits of key strength respectively	RSA SigGen (FIPS186-5): (A5009, A5018)
FIPS provider RSA Signature Verification	DigSig-SigVer	Signature Verification	Keys:2048, 3072, 4096 keys with 112-150 bits of key strength respectively	RSA SigVer (FIPS186-5): (A5009, A5018)
FIPS provider HMAC	MAC	Message authentication code (MAC)	Keys:112-256 bits with 112-256 bits of key strength	HMAC-SHA-1: (A5009, A5018) HMAC-SHA2-224: (A5009, A5018) HMAC-SHA2-256: (A5009, A5010, A5018) HMAC-SHA2-384: (A5009, A5018) HMAC-SHA2-512: (A5009, A5018) HMAC-SHA2-512/224: (A5009, A5018) HMAC-SHA2-512/256: (A5009, A5018) HMAC-SHA3-224: (A5011, A5020) HMAC-SHA3-256: (A5011,

Name	Type	Description	Properties	Algorithms
				A5020) HMAC-SHA3-384: (A5011, A5020) HMAC-SHA3-512: (A5011, A5020)
FIPS provider KMAC	MAC	Message authentication code (MAC)	Keys:112-256 bits with 112-256 bits of key strength	KMAC-128: (A5011, A5020) KMAC-256: (A5011, A5020)
FIPS provider Hashes	SHA	Hashing		SHA-1: (A5009, A5018) SHA2-224: (A5009, A5018) SHA2-256: (A5009, A5010, A5018) SHA2-384: (A5009, A5018) SHA2-512: (A5009, A5018) SHA2-512/224: (A5009, A5018) SHA2-512/256: (A5009, A5018) SHA3-224: (A5011, A5020) SHA3-256: (A5011, A5020) SHA3-384: (A5011, A5020) SHA3-512: (A5011, A5020) SHAKE-128: (A5011, A5020)

Name	Type	Description	Properties	Algorithms
				SHAKE-256: (A5011, A5020)
FIPS provider ANS 9.42 Key Derivation (CVL)	KAS-135KDF	Key Derivation	OID:AES-128-KW, AES-192-KW, AES-256-KW with 128, 192, 256 bits of key strength, respectively	KDF ANS 9.42: (A5009, A5011, A5018, A5020)
FIPS provider ANS 9.63 Key Derivation (CVL)	KAS-135KDF	Key Derivation	Key data length:128-4096 bits	KDF ANS 9.63: (A5009, A5018)
FIPS provider TLS 1.0 and 1.1 Key Derivation (CVL)	KAS-135KDF	Key Derivation	Derived key:112-256 bits with 112-256 bits of key strength	KDF TLS: (A5009, A5018)
FIPS provider TLS 1.2 Key Derivation (CVL)	KAS-135KDF	Key Derivation	Derived key:112-256 bits with 112-256 bits of key strength	TLS v1.2 KDF RFC7627: (A5009, A5018)
FIPS provider TLS 1.3 Key Derivation (CVL)	KAS-135KDF	Key Derivation	Derived key:112-256 bits with 112-256 bits of key strength	TLS v1.3 KDF: (A5013)
FIPS provider HKDF Key Derivation	KAS-56CKDF	Key Derivation	Derived key:112-256 bits with 112-256 bits of key strength	KDA HKDF Sp800-56Cr1: (A5013)
FIPS provider Password-based Key Derivation	PBKDF	Key Derivation	Derived key:112-4096 bits with 112-150 bits of key strength	PBKDF: (A5009, A5011, A5018, A5020)
FIPS provider OneStep Key Derivation	KAS-56CKDF	Key Derivation	Derived key:2048 bits with 112 bits of key strength	KDA OneStep SP800-56Cr2: (A5012)
FIPS provider TwoStep Key Derivation	KAS-56CKDF	Key Derivation	Derived key:2048 bits with 112 bits of key strength	KDA TwoStep SP800-56Cr2: (A5012)

Name	Type	Description	Properties	Algorithms
FIPS provider KMAC Key Derivation	KBKDF	Key Derivation	Derived key:112-4096 bits with 112-150 bits of key strength	KDF KMAC Sp800-108r1: (A5017)
FIPS provider KBKDF Key Derivation	KBKDF	Key Derivation.	Derived key:112-4096 bits with 112-150 bits of key strength	KDF SP800-108: (A5017)
FIPS provider SSH Key Derivation	KAS-135KDF	Key Derivation	Keys:128, 192, 256 bits with 128-256 bits of key strength	KDF SSH: (A5019)

Table 8: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 AES GCM IV

AES-GCM encryption and decryption are used in the context of the TLS protocol version 1.2 and 1.3 using the FIPS provider component (corresponding to Scenario 1 and 5 of IG C.H), and in the context of IEEE 802.11 GCMP using the kernel/hardware components (corresponding to Scenario 5 of IG C.H). For IPsec, the module offers the AES GCM implementation and uses the context of Scenario 1 of FIPS 140-3 IG C.H. The mechanism for IV generation is compliant with RFC 4106. IVs generated using this mechanism may only be used in the context of AES GCM encryption within the IPsec protocol.

Alternatively, the Crypto Officer can use the module's API to perform AES GCM encryption using internal IV generation. These IVs are always 96 bits and generated using the approved DRBG internal to the module's boundary, compliant with Scenario 2 of FIPS 140-3 IG C.H.

The module also provides a non-approved AES GCM encryption service which accepts arbitrary external IVs from the operator. This service can be requested by invoking the EVP_EncryptInit_ex2 API function with a non-NULL iv value. When this is the case, the API will set a non-approved service indicator.

2.7.1.1 TLS version 1.2

For TLS v1.2, the module uses the context of Scenario 1 of IG C.H. The module is compliant with SP 800-52 section 3.3.1 and the mechanism for IV generation is compliant with RFC5288. For this compliance, the module's implementation of the AES-GCM shall be used together with an application that negotiates the protocol session's keys and the 32-bit nonce value of the IV. The setting of the counter portion of the IV is performed within the cryptographic boundary.

The nonce explicit part of the IV does not exhaust the maximum number of possible values for a given session key. This condition is implicitly ensured by the design of the TLS protocol, in which the nonce_explicit is denied exhaustion by the control exerted by the protocol's management logic (wherein the nonce_explicit is incremented per each TLS record). This management logic also implies that the probability of an exhaustion of all $2^{64} – 1$ values of the nonce_explicit for the same TLS session in a realistic time frame is not significant.

In the event the module’s power is lost and restored, the consuming application must ensure that a new key for use with the AES GCM key encryption or decryption under this scenario shall be established.

2.7.1.2 TLS version 1.3

For TLS 1.3, the AES GCM implementation uses the context of Scenario 5 of FIPS 140-3 IG C.H. The protocol that provides this compliance is TLS 1.3, defined in RFC8446 of August 2018, using the cipher-suites that explicitly select AES GCM as the encryption/decryption cipher (Appendix B.4 of RFC8446). The module supports acceptable AES GCM cipher suites from Section 3.3.1 of SP800-52r2. TLS 1.3 employs separate 64-bit sequence numbers, one for protocol records that are received, and one for protocol records that are sent to a peer. These sequence numbers are set at zero at the beginning of a TLS 1.3 connection and each time when the AES-GCM key is changed. After reading or writing a record, the respective sequence number is incremented by one. The protocol specification determines that the sequence number should not wrap, and if this condition is observed, then the protocol implementation must either trigger a re-key of the session (i.e., a new key for AES-GCM), or terminate the connection.

In the event the module’s power is lost and restored, the consuming application must ensure that a new key for use with the AES GCM key encryption or decryption under this scenario shall be established.

2.7.1.3 IEEE 802.11 GCMP

The kernel component is in compliance with FIPS 140-3 IG C.H scenario 5 for the WPA2 protocol. Specifically, GCMP is defined in IEEE 802.11ac-2013. For IEEE 802.11 GCMP, the module implements an internal production unit logic that constructs the IV deterministically upon the initialization of a GCMP connection, and therefore the initialization of a GCM encryption context." and "In case the module's power is lost and then restored, the key used for AES GCM encryption or decryption shall be re-distributed.

2.7.2 AES XTS

The length of a single data unit encrypted or decrypted with AES XTS shall not exceed 2^{20} AES blocks, that is 16MB, of data per XTS instance. An XTS instance is defined in Section 4 of SP 800-38E. To meet the requirement stated in IG C.I, the module implements a check to ensure that the two AES keys used in AES XTS mode are not identical.

The XTS mode shall only be used for the cryptographic protection of data on storage devices. It shall not be used for other purposes, such as the encryption of data in transit.

2.7.3 Key derivation using SP 800-132 PBKDF2

The module provides password-based key derivation (PBKDF2), compliant with SP 800-132. The module supports option 1a from Section 5.4 of SP 800-132, in which the Master Key (MK) or a segment of it is used directly as the Data Protection Key (DPK). In accordance to SP 800-132 and FIPS 140-3 IG D.N, the following requirements shall be met:

- Derived keys shall only be used in storage applications. The MK shall not be used for other purposes. The length of the MK or DPK shall be of 112 bits or more.
- Passwords or passphrases, used as an input for the PBKDF2, shall not be used as cryptographic keys.

- The length of the password or passphrase shall be at least 8 characters. The probability of guessing the value, assuming a worst-case scenario of all digits, is estimated to be at most 10^{-8}. Combined with the minimum iteration count as described below, this provides an acceptable trade-off between user experience and security against brute-force attacks.
- A portion of the salt, with a length of at least 128 bits, shall be generated randomly using the SP 800-90Ar1 DRBG provided by the module.
- The iteration count shall be selected as large as possible, as long as the time required to generate the key using the entered password is acceptable for the users. The minimum value is 1000.

2.7.4 SP 800-56Ar3 Assurances

Kernel Component: The module offers ECDH shared secret computation services compliant to the SP 800-56ARev3. In order to meet the required assurances listed in section 5.6 of SP 800-56ARev3, the module shall be used together with an application that implements the "IPsec protocol" and the following steps shall be performed.

1. The entity using the module, must use the module's "key pair generation" service for generating ECDH ephemeral keys. The key generation service performs full public key validation. This meets the assurances required by key pair owner defined in the section 5.6.2.1 of SP 800-56ARev3.
2. The consumer using the module doesn't need to obtain assurance of the peer's possession of private key as the module only makes use of ephemeral keys.
3. As part of the module’s shared secret computation service, the module internally performs the public key validation on the peer's public key passed in as input to the SSC function. This meets the public key validity assurance required by the sections 5.6.2.2.1/5.6.2.2.2 of SP 800-56ARev3.

FIPS provider Component: The module offers DH and ECDH shared secret computation services compliant to the SP 800-56ARev3. To comply with the assurances found in Section 5.6.2 of SP 800-56Ar3, the operator must use the module together with an application that implements the TLS protocol. Additionally, the module’s approved key pair generation service must be used to generate ephemeral Diffie-Hellman or EC Diffie-Hellman key pairs, or the key pairs must be obtained from another FIPS-validated module. As part of this service, the module will internally perform the full public key validation of the generated public key. The module’s shared secret computation service will internally perform the full public key validation of the peer public key, complying with Sections 5.6.2.2.1 and 5.6.2.2.2 of SP 800-56Ar3.

2.7.5 RSA Key Encapsulation

To comply with SP800-56Br2 assurances found in its Section 6 (specifically SP800-56Br2 Section 6.4 Required Assurances) the entity using the module must obtain required assurances listed in section 6.4 of SP 800-56Br2 by performing the following steps:

1. The entity requesting the RSA key un-encapsulation service from the module, shall only use an RSA private key that was generated by an active FIPS validated module that implements FIPS 186-5 compliant RSA key generation service and performs the key pair validity and the pairwise consistency as stated in section 6.4.1.1 of the SP 800-56Br2. Additionally, the entity shall renew these assurances over time by using any method described in section 6.4.1.5 of the SP 800-56Br2.
2. For use of an RSA key encapsulation service in the context of key transport per IG D.G the entity using the module shall:

a. verify the validity of the peer's public key using the public key validation service of the module (EVP_PKEY_check() API).
b. confirm the peer's possession of private key by using any method specified in section 6.4.2.3 of the SP 800-56Br2.

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

2.7.6 RSA Key Agreement

To comply with the assurances found in Section 6.4 of SP 800-56Br2, the module's approved RSA key pair generation service must be used to generate the RSA key pairs, or the key pairs must be obtained from another FIPS-validated module. As part of this service, the module will internally perform the key pair validity and the pairwise consistency according to section 6.4.1.1 of SP 800-56Br2.

Additionally, the entity requesting the shared secret computation service shall verify the validity of the peer's public key using the public key validation service of the module (EVP_PKEY_check() API). This service will perform the full public key validation of the peer's public key, complying with Section 6.4.2.1 of SP 800-56Br2.

2.7.7 RSA SigGen and SigVer compliance

The module provides RSA signature generation and signature verification compliant with IG C.F.
The module supports RSA modulus lengths of 2048, 3072, and 4096 bits for signature generation and 1024, 2048, 3072, and 4096 for signature verification.
The RSA signature generation and signature verification implementations have been tested for all implemented RSA modulus lengths.
The number of Miller-Rabin tests is consistent with the bit sizes of p and q from Table B.1 of FIPS 186-4.

2.7.8 SHA-3 compliance

The module provides SHA-3 hash functions compliant with IG C.C.
Every implementation of each SHA-3 function was tested and validated on all of the module's operating environments.
SHAKE functions are not implemented.
SHA-3 hash functions are also used as part of a higher-level algorithm for HMAC.

2.8 RBG and Entropy

Cert Number	Vendor Name
E119	Ezurio

Table 9: Entropy Certificates

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
Summit CPU Time Jitter RNG Entropy Source	Non-Physical	Summit Linux 11.0	64 bits	64 bits	A4713 (SHA3-256)

Table 10: Entropy Sources

The module provides DRBG's compliant with SP800-90A for random number generation and the creation of key components of asymmetric keys. The kernel component DRBG implements a CTR_DRBG mechanism with AES-128, AES-192 or AES-256, with selectable enabling of derivation function and prediction resistance. The FIPS provider component DRBG's implements a CTR_DRBG mechanism with AES-128, AES-192 or AES-256 with selectable enabling of derivation function and prediction resistance. Additionally, a Hash_DRBG and HMAC_DRBG mechanism with SHA-1, SHA-256 or SHA-512 with selectable enabling of derivation function and prediction resistance is implemented. Lastly the kernel and FIPS Provider DRBG's are seeded with 320 bits and assessed to be full entropy in the ESV certificate #E119 at 1 bit/bit or 256 bits of entropy.

2.9 Key Generation

For generating RSA, ECDSA, Diffie-Hellman, EC Diffie-Hellman keys for the FIPS Provider component and ECDSA keys for Kernel component, the module implements asymmetric key generation services compliant with FIPS186-5 or SP800-56Arev3 as applicable and using a DRBG compliant with SP800-90A. The random value used in asymmetric key generation is obtained from the DRBG. In accordance with FIPS 140-3 IG D.H, the cryptographic module performs Cryptographic Key Generation (CKG) for asymmetric keys as per Section 4 of SP800-133rev2 (vendor affirmed).

Additionally, the module implements the following key derivation methods according to section 6.2 of SP 800-133r2:

- HKDF compliant with SP 800-56Cr1: derivation of secret keys in the context of an SP 800-56Ar3 and SP 800-56Br2 key agreement schemes.
- TLS KDF compliant with SP 800-135r1: derivation of secret keys in the context of the TLS 1.0/1.1, TLS 1.2 and TLS 1.3 protocols.
- PBKDF2: compliant with option 1a of SP 800-132: derivation of keys for use in storage applications.
- ANS 9.42 KDF compliant with SP 800-135r1: derivation of secret keys in the context of ANS X9.42-2001 key agreement schemes.
- ANS 9.63 KDF compliant with SP 800-135r1: derivation of secret keys in the context ANS X9.63-2001 key agreement schemes.
- KBKDF compliant with Sp800-108r1: derivation of secret keys.
- KDA OneStep and KDA TwoStep compliant with SP800-56Cr2: derivation of secret keys in the context of SP 800-56Ar3 and SP 800-56Br2 key agreement schemes.

2.10 Key Establishment

The module implements following key establishments methods that are listed in the Security Function Implementations table:
- shared secret computation for KAS-IFC-SSC, KAS-FFC-SSC KAS-ECC-SSC

- key transport for KTS-IFC and KTS-Wrap

2.11 Industry Protocols

For DH, the module supports the use of the safe primes defined in RFC 3526 (IKE) and RFC 7919 (TLS) as listed in Approved Services table. Note that the module only implements key pair generation, key pair verification, and shared secret computation. No other part of the IKE or TLS protocols is implemented (with the exception of the TLS 1.2 and 1.3 KDFs).

SSH KDF, TLS 1.0/1.1 KDF, TLS 1.2 KDF (RFC 7627), TLS 1.3 KDF implementations shall only be used to generate secret keys in the context of the SSH, TLS 1.0/1.1, TLS 1.2, or TLS 1.3 protocols, respectively. Note that TLS 1.2 KDF must be compliant with RFC 7627 to be considered approved.

ANS X9.42 KDF and ANS X9.63 KDF implementations shall only be used to generate secret keys in the context of an ANS X9.42-2001 resp. ANS X9.63-2001 key agreement scheme.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	API data input parameters, AF_ALG type sockets (kernel component)
N/A	Data Output	API output parameters, AF_ALG type sockets (kernel component)
N/A	Control Input	API function calls, API control input parameters, AF_ALG type sockets (kernel component), kernel command line (kernel component)
N/A	Status Output	API return values, error queue, AF_ALG type sockets (kernel component), kernel logs (kernel component)
N/A	Power	The hardware on which the module runs receives power from the circuit board on which the hardware resides.

Table 11: Ports and Interfaces

The logical interfaces are the APIs through which the applications request services. These logical interfaces are logically separated from each other by the API design.

4 Roles, Services, and Authentication

4.1 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	Crypto Officer	None

Table 12: Roles

The module supports the Crypto Officer role only. This sole role is implicitly and always assumed by the operator of the module. No support is provided for multiple concurrent operators.

4.2 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Kernel Encryption	Encryption	crypto_skcipher_setkey returns 0	AES key, plaintext	ciphertext	Kernel AES-ECB Kernel AES-CTR Kernel AES-CBC Kernel AES-CBC-CS3 Kernel AES-XTS	Crypto Officer - Kernel AES key: W,E
Kernel Decryption	Decryption	crypto_skcipher_setkey returns 0	AES key, ciphertext	plaintext	Kernel AES-ECB Kernel AES-CTR Kernel AES-CBC Kernel AES-CBC-CS3 Kernel AES-XTS	Crypto Officer - Kernel AES key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Kernel Authenticated Encryption	Encryption	crypto_aead_setkey returns 0	AES key, IV, plaintext	ciphertext, MAC tag	Kernel AES-CCM (BC-Auth) Kernel AES-GCM (BC-Auth)	Crypto Officer - Kernel AES key: W,E
Kernel Authenticated Decryption	Decryption	crypto_aead_setkey returns 0	AES key, IV, MAC tag, ciphertext	plaintext or fail	Kernel AES-CCM (BC-Auth) Kernel AES-GCM (BC-Auth)	Crypto Officer - Kernel AES key: W,E
Kernel key wrapping	Wrap a key	crypto_skcipher_setkey returns 0; crypto_aead_setkey returns 0; crypto_shash_init returns 0	AES key, key to be wrapped	wrapped key	Kernel AES-CCM (KTS-Wrap) Kernel AES-GCM (KTS-Wrap) Kernel AES CBC with HMAC Kernel AES CTR with HMAC	Crypto Officer - Kernel AES key: W,E
Kernel key unwrapping	unwrap a key	crypto_skcipher_setkey returns 0; crypto_aead_setkey returns 0; crypto_shash_init returns 0	AES key, key to be unwrapped	unwrapped key	Kernel AES-CCM (KTS-Wrap) Kernel AES-GCM (KTS-Wrap) Kernel	Crypto Officer - Kernel AES key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					AES CBC with HMAC Kernel AES CTR with HMAC	
Kernel AES Message Authentication	compute a MAC tag	crypto_shash_init returns 0	AES key, message	MAC tag	Kernel AES-CMAC Kernel AES-GMAC	Crypto Officer - Kernel AES key: W,E
Kernel HMAC Message Authentication	compute a MAC tag	crypto_shash_init returns 0	HMAC key, message	MAC tag	Kernel HMAC	Crypto Officer - Kernel HMAC key: W,E
Kernel Message Digest	compute a message digest	crypto_shash_init returns 0	message	digest value	Kernel Hashes	Crypto Officer
Kernel ECC Shared Secret Computation	compute a shared secret	crypto_kpp_compute_shared_secret returns 0	EC public key, EC private key	Shared Secret	Kernel KAS-ECC-SSC	Crypto Officer - Kernel EC public key: W,E - Kernel EC private key: W,E - Kernel shared secret: W,E
Kernel Random Number Generation	generate random bytes	crypto_rng_get_bytes returns 0	output length	random data	Kernel Counter DRBG	Crypto Officer - Entropy input: W,E - Kernel DRBG seed: G,E - DRBG Internal State (V, Key): W,E - DRBG Internal state (V, C): W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Kernel EC Key generation	generate key pair	crypto_kpp_set_secret and crypto_kpp_generate_public_key return 0	Curve	EC keys	Kernel ECDSA Key Generation	Crypto Officer - Kernel EC public key: G,R - Kernel EC private key: G,R - Kernel Intermediate Key Generation Value: G,E,Z
FIPS provider Message Digest	compute a mesage digest	_SUMMIT_FIPS_INDICATOR_APPROVED	message	digest value	FIPS provider Hashes	Crypto Officer
FIPS provider Encryption	Encrypt plaintext	_SUMMIT_FIPS_INDICATOR_APPROVED	AES key, plaintext	ciphertext	FIPS provider AES-CTR FIPS provider AES-CBC FIPS provider AES-ECB FIPS provider AES-CBC-CS1 FIPS provider AES-CBC-CS2 FIPS provider AES-CBC-CS3 FIPS provider AES-CFB1 FIPS provid	Crypto Officer - FIPS provider AES Key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					er AES-CFB8 FIPS provider AES-CFB128 FIPS provider AES-XTS FIPS provider AES-OFB	
FIPS provider Decryption	Decrypt ciphertext	_SUMMIT_FIPS_INDICATOR_APPROVED	AES key, ciphertext	plaintext	FIPS provider AES-CTR FIPS provider AES-CBC FIPS provider AES-ECB FIPS provider AES-CBC-CS1 FIPS provider AES-CBC-CS2 FIPS provider AES-CBC-CS3 FIPS provider AES-CFB1 FIPS provider AES-CFB8 FIPS	Crypto Officer - FIPS provider AES Key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					provider AES-CFB128 FIPS provider AES-XTS FIPS provider AES-OFB	
FIPS provider Authenticated Encryption	Encrypt plaintext	_SUMMIT_FIPS_INDICATOR_APPROVED	AES key, IV, plaintext	ciphertext, MAC tag	FIPS provider AES-CCM (BC-Auth) FIPS provider AES-GCM (BC-Auth)	Crypto Officer - FIPS provider AES Key: W,E
FIPS provider Authenticated Decryption	Decrypt ciphertext	_SUMMIT_FIPS_INDICATOR_APPROVED	AES key, IV, MAC tag, ciphertext	plaintext	FIPS provider AES-CCM (BC-Auth) FIPS provider AES-GCM (BC-Auth)	Crypto Officer - FIPS provider AES Key: W,E
FIPS provider AES Message Authentication	compute a MAC tag	_SUMMIT_FIPS_INDICATOR_APPROVED	AES key, message	MAC tag	FIPS provider AES-CMAC FIPS provider AES-GMAC	Crypto Officer - FIPS provider AES Key: W,E
FIPS provider HMAC Message Authentication	compute a MAC tag	_SUMMIT_FIPS_INDICATOR_APPROVED	HMAC key, message	MAC tag	FIPS provider HMAC	Crypto Officer - FIPS provider HMAC key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
FIPS provider FFC Shared Secret Computation	compute a shared secret	_SUMMIT_FIPS_INDICATOR_APPROVED	DH private key, DH public key	Shared Secret	FIPS provider KAS-FFC-SSC	Crypto Officer - FIPS provider DH public key: W,E - FIPS provider DH private key: W,E
FIPS provider ECC Shared Secret Computation	compute a shared secret	_SUMMIT_FIPS_INDICATOR_APPROVED	EC public key, EC private key	Shared Secret	FIPS provider KAS-ECC-SSC	Crypto Officer - FIPS provider EC public key: W,E - FIPS provider EC private key: W,E - FIPS provider shared secret: W,E
FIPS provider IFC Shared Secret Computation	compute a shared secret	_SUMMIT_FIPS_INDICATOR_APPROVED	RSA public key, RSA private key	Shared Secret	FIPS provider KAS-IFC-SSC	Crypto Officer - FIPS provider RSA public key: W,E - FIPS provider RSA private key: W,E - FIPS provider shared secret: W,E
FIPS provider Key Derivation	derive a key	_SUMMIT_FIPS_INDICATOR_APPROVED	Shared secret	derived key	FIPS provider ANS 9.42 Key Derivation (CVL) FIPS provid	Crypto Officer - FIPS provider shared secret: W,E - FIPS provider derived

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					er ANS 9.63 Key Derivation (CVL) FIPS provider TLS 1.0 and 1.1 Key Derivation (CVL) FIPS provider TLS 1.2 Key Derivation (CVL) FIPS provider TLS 1.3 Key Derivation (CVL) FIPS provider HKDF Key Derivation FIPS provider OneStep Key Derivation FIPS provider TwoStep Key	key: G,R - FIPS provider key-derivation key: W,E - FIPS provider AES Derived Key: G,R - FIPS provider HMAC Derived Key : G,R - FIPS provider 802.11 Pre-shared key (PSK): W,E - FIPS provider 802.11 Pairwise Master Key (PMK): W,E - FIPS provider 802.11 KDF Internal State: R - FIPS provider 802.11 Temporal Keys: W,E - FIPS provider 802.11 MIC keys (KCK): W,E - FIPS provider 802.11 Key

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					Derivation FIPS provider KMAC Key Derivation FIPS provider KBKDF Key Derivation FIPS provider SSH Key Derivation	Encryption Key (KEK): W,E - FIPS provider 802.11 Group Temporal Key (GTK): W,E
FIPS provider Password-based key derivation	derive a key from a password	_SUMMIT_FIPS_INDICATOR_APPROVED	password	derived key	FIPS provider Password-based Key Derivation	Crypto Officer - FIPS provider derived key: G,R - FIPS provider Password: W,E
FIPS provider SafePrime key generation	generate a key pair	_SUMMIT_FIPS_INDICATOR_APPROVED	DH-Group	Module generated DH private key, Module generated DH public key	FIPS provider Safe Primes Key Generation	Crypto Officer - FIPS provider module generated DH public key: G,R - FIPS provider module generated DH private key: G,R - FIPS provider Intermediate Key Generatio

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						n Value: G,E,Z
FIPS provider EC Key generation	generate a key pair	_SUMMIT_FIPS_INDICATOR_APPROVED	Curve	Module Generated EC Private Key, Module Generated EC Public Key	FIPS provider ECDSA Key Generation FIPS provider EDDSA Key Generation	Crypto Officer - FIPS provider module generated EC public key: G,R - FIPS provider module generated EC private key: G,R - FIPS provider Intermediate Key Generation Value: G,E,Z
FIPS provider RSA key generation	generate a key pair	_SUMMIT_FIPS_INDICATOR_APPROVED	Modulus	Module Generated RSA Private Key, Module Generated RSA Public Key	FIPS provider RSA Key Generation	Crypto Officer - FIPS provider module generated RSA private key: G,R - FIPS provider module generated RSA public key: G,R - FIPS provider Intermediate Key Generation Value: G,E,Z
FIPS provider SafePrime Key Verification	verify key pair	_SUMMIT_FIPS_INDICATOR_APPROVED	DH Private key, DH public key	Pass/fail	FIPS provider Safe Primes Key	Crypto Officer - FIPS provider DH public key: W

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					Verification	- FIPS provider DH private key: W
FIPS provider EC Key Verification	verify key pair	_SUMMIT_FIPS_INDICATOR_APPROVED	EC public key, EC private key	Pass/fail	FIPS provider ECDSA Key Verification	Crypto Officer - FIPS provider EC public key: W - FIPS provider EC private key: W
FIPS provider Key wrapping	wrap a key	_SUMMIT_FIPS_INDICATOR_APPROVED	AES key, key to be wrapped	wrapped key	FIPS provider AES-CCM (KTS-Wrap) FIPS provider AES-GCM (KTS-Wrap) FIPS provider AES KW FIPS provider AES KWP	Crypto Officer - FIPS provider AES Key: W,E
FIPS provider Key unwrapping	unwrap a key	_SUMMIT_FIPS_INDICATOR_APPROVED	AES key, key to be unwrapped	unwrapped key	FIPS provider AES-CCM (KTS-Wrap) FIPS provider AES-GCM (KTS-Wrap) FIPS provider AES KW FIPS	Crypto Officer - FIPS provider AES Key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					provider AES KWP	
FIPS provider RSA Signature Verification	verify digital signature	_SUMMIT_FIPS_INDICATOR_APPROVED	RSA public key, signature, hash algorithm	Pass/fail	FIPS provider RSA Signature Verification	Crypto Officer - FIPS provider RSA public key: W,E
FIPS provider EC Signature Verification	verify digital signature	_SUMMIT_FIPS_INDICATOR_APPROVED	Message, EC public key, signature, hash algorithm (ECDSA only)	Pass/fail	FIPS provider ECDSA Signature Verification FIPS provider EDDSA Signature Verification	Crypto Officer - FIPS provider EC public key: W,E
FIPS provider EC Signature Generation	generate digital signature	_SUMMIT_FIPS_INDICATOR_APPROVED	Message, EC public key, signature, hash algorithm (ECDSA only)	signature	FIPS provider ECDSA Signature Generation FIPS provider EDDSA Signature Generation	Crypto Officer - FIPS provider EC private key: W,E
FIPS provider RSA Signature Generation	generate digital signature	_SUMMIT_FIPS_INDICATOR_APPROVED	Message,RSA public key, signature, hash algorithm	signature	FIPS provider RSA Signature Generation	Crypto Officer - FIPS provider RSA private key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
FIPS provider Random Number Generation	generate random bytes	_SUMMIT_FIPS_INDICATOR_APPROVED	output length	random bytes	FIPS provider Counter DRBG FIPS provider Hash DRBG FIPS provider HMAC DRBG	Crypto Officer - Entropy input: W,E - FIPS provider DRBG seed: G,E - DRBG Internal State (V, Key): G,W,E - DRBG Internal state (V, C): G,W,E
FIPS provider key encapsulation	KTS	_SUMMIT_FIPS_INDICATOR_APPROVED	RSA public key	Encapsulated key	FIPS provider KTS-IFC	Crypto Officer - FIPS provider RSA public key: W,E
FIPS provider key un-encapsulation	KTS	_SUMMIT_FIPS_INDICATOR_APPROVED	RSA private key	Decapsulated key	FIPS provider KTS-IFC	Crypto Officer - FIPS provider RSA private key: W,E
FIPS provider KMAC Message Authentication	MAC	_SUMMIT_FIPS_INDICATOR_APPROVED	KMAC key	Mac tag	FIPS provider KMAC	Crypto Officer - FIPS Provider KMAC Key: W,E
Show version	Return the module name and version information	None	N/A	module name and version	None	Unauthenticated
Show status	return module status	None	N/A	module status	None	Unauthenticated
Self-Tests	perform CASTs and	None	N/A	Pass/Fail	None	Unauthenticated

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	integrity test					
Zeroization	zeroize all SSPs	None	Any SSP	N/A	None	Crypto Officer - Kernel AES key: Z - FIPS provider AES Key: Z - Kernel shared secret: Z - Kernel HMAC key: Z - FIPS provider shared secret: Z - Entropy input: Z - Kernel DRBG seed: Z - DRBG Internal State (V, Key): Z - DRBG Internal state (V, C): Z - FIPS provider DH public key: Z - FIPS provider DH private key: Z - Kernel EC public key: Z - Kernel EC private key: Z - FIPS provider EC public key: Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- FIPS provider EC private key: Z - FIPS provider module generated DH public key: Z - FIPS provider module generated DH private key: Z - FIPS provider module generated EC public key: Z - FIPS provider module generated RSA public key: Z - FIPS provider RSA public key: Z - FIPS provider RSA private key: Z - FIPS provider module generated RSA public key: Z - FIPS provider module generated RSA private key: Z - FIPS

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						provider Intermediate Key Generation Value: Z - FIPS provider derived key: Z - FIPS provider DRBG seed: Z - Kernel Intermediate Key Generation Value: Z - FIPS provider key-derivation key: Z - FIPS provider Password: Z - FIPS provider HMAC key: Z - FIPS Provider KMAC Key: Z - FIPS provider 802.11 Pairwise Master Key (PMK): Z - FIPS provider 802.11 Group Temporal Key (GTK): Z - FIPS provider

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						802.11 KDF Internal State: Z - FIPS provider 802.11 Temporal Keys: Z - FIPS provider 802.11 MIC keys (KCK): Z - FIPS provider 802.11 Key Encryption Key (KEK): Z

Table 13: Approved Services

The table above lists the approved services. The following convention is used to specify access rights to SSPs:

- **Generate (G):** The module generates or derives the SSP.
- **Read (R):** The SSP is read from the module (e.g. the SSP is output).
- **Write (W):** The SSP is updated, imported, or written to the module.
- **Execute (E):** The module uses the SSP in performing a cryptographic operation.
- **Zeroize (Z):** The module zeroizes the SSP.

To interact with the FIPS provider component of the module, a calling application must use the EVP API layer provided by OpenSSL. This layer will delegate the request to the FIPS provider, which will in turn perform the requested service. The EVP_KDF_CTX_get_params() function can be used to determine whether an EVP API function is approved. After a cryptographic service was performed by the module, the API context associated with this request can contain a parameter (listed below) which represents the approved service indicator.

- _SUMMIT_FIPS_INDICATOR_APPROVED (approved services)
- _SUMMIT_FIPS_INDICATOR_NOT_APPROVED (non-approved services)

The security function implementation “FIPS provider KBKDF Key Derivation” listed for the “FIPS provider Key Derivation” approved service in the table above is intended to derive keys for use of 802.11 protocols.

The Kernel does not provide any non-approved services. The API for the Kernel services listed in the Indicator column, return the specified values that indicate that the services are approved.

4.3 Non-Approved Services

Name	Description	Algorithms	Role
FIPS provider PBKDF with salt length less than 128 bits	Key derivation	FIPS provider PBKDF with salt length less than 128 bits	CO
FIPS provider TLSv1.0 and TLSv1.1 KDF using EMS	Key derivation	FIPS provider TLSv1.0 and TLSv1.1 KDF using EMS	CO
FIPS provider TLSv1.2 KDF without using EMS	Key derivation	FIPS provider TLSv1.2 KDF without using EMS	CO

Table 14: Non-Approved Services

4.4 External Software/Firmware Loaded

The module does not load external software or firmware.

5 Software/Firmware Security

5.1 Integrity Techniques

The integrity of the module’s firmware components (the kernel, the FIPS Provider components and fipscheck application and library) is individually verified by the fipscheck integrity test tool using HMAC-SHA2-256 implemented in the FIPS Provider. The FIPS Provider component executes its CASTs (which include the KAT for the integrity technique using HMAC-SHA2-256) before the fipscheck integrity test tool executes to verify the integrity of the module. The HMAC value of each firmware component is computed at build time and stored in the .hmac file for each component. The value is recalculated at runtime for the image of the kernel, FIPS Provider binary and fipscheck application and library, and then compared against the stored value in the file. If the comparison succeeds, then the remaining CASTs (consisting of the Known Answer Tests) for the kernel are performed.

5.2 Initiate on Demand

Integrity tests are performed as part of the pre-operational self-tests, which are executed when the module is initialized. The integrity tests can be invoked on demand by unloading and subsequently re-initializing the module, which will perform (among others) the firmware integrity tests.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Limited

How Requirements are Satisfied:

The operating system provides process isolation and memory protection mechanisms that ensure appropriate separation for memory access among the processes on the system. Each process has control over its own data and uncontrolled access to the data of other processes is prevented.

6.2 Configuration Settings and Restrictions

The module shall be installed as stated in Section 11.1.

Instrumentation tools like the ptrace system call, gdb and strace, as well as other tracing mechanisms offered by the Linux environment such as ftrace or systemtap, shall not be used in the operational environments. The use of any of these tools implies that the cryptographic module is running in a non-validated operational environment.

7 Physical Security

7.1 Mechanisms and Actions Required

N/A for this module.

The module is enclosed within a production-grade enclosure with components that include standard passivation techniques (e.g., a conformal coating applied over the module's circuitry to protect against environmental or other physical damage) conformant to the Level 1 requirements for physical security.

8 Non-Invasive Security

8.1 Mitigation Techniques

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

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
RAM	Temporary storage for SSPs used by the module as part of service execution. The module does not perform persistent storage of SSPs.	Dynamic

Table 15: Storage Areas

The module does not perform persistent storage of SSPs. The SSPs are temporarily stored in the RAM in plaintext form. SSPs are provided to the module by the calling process and are destroyed when released by the appropriate zeroization function calls.

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
API input parameters	Operating calling application (TOEPP)	Cryptographic module	Plaintext	Manual	Electronic	
Kernel AF_ALG_type sockets (input)	Operating calling application (TOEPP)	Cryptographic module	Plaintext	Manual	Electronic	
API output parameters	Cryptographic module	Operator calling application (TOEPP)	Plaintext	Manual	Electronic	
Kernel AF_ALG type sockets (output)	Cryptographic module	Operator calling application (TOEPP)	Plaintext	Manual	Electronic	

Table 16: SSP Input-Output Methods

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Kernel free cipher handle	Zeroizes the SSPs contained within the cipher handle	Memory occupied by SSPs is overwritten with zeroes, which renders the SSP values irretrievable. The completion of	By calling the appropriate zeroization functions:- AES key: crypto_free_skcipher and crypto_free_aead; - DRBG Internal state: crypto_free_rng; - EC public

Zeroization Method	Description	Rationale	Operator Initiation
		the zeroization routine(s) indicate that the zeroization procedure succeeded	& private key: crypto_free_kpp and crypto_free_akcipher
FIPS provider calling the zeroization API	Zeroizes the SSPs	Memory occupied by SSPs is overwritten with zeroes, which renders the SSP values irretrievable. All data output is inhibited during zeroization. The completion of the zeroization routine(s) indicate that the zeroization procedure succeeded	By calling the appropriate zeroization functions: - EVP_CIPHER_CTX_free(): clears and frees symmetric cipher context; - EVP_MAC_CTX_free(): clears and frees MAC context; - EVP_KDF_CTX_free(): clears and frees KDF context; - EVP_RAND_CTX_free(): clears and frees DRBG context; - EVP_PKEY_free(): clears and frees asymmetric key pair structures
FIPS provider Automatic	Zeroizes the SSPs	Memory occupied by SSPs is overwritten with zeroes, which renders the SSP values irretrievable. All data output is inhibited during zeroization.	Intermediate key generation value: zeroized automatically by the module (after the requested service completed)
Remove power from the module	De-allocates the volatile memory used to store SSPs	Volatile memory used by the module is overwritten within nanoseconds when power is removed. Module power off indicates that the zeroization procedure succeeded.	By removing power

Table 17: SSP Zeroization Methods

All data output is inhibited during zeroization.

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Kernel AES key	AES key used for encryption, decryption, and computing MAC tags	128, 192, 256 bits - 128,	Symmetric Key - CSP			Kernel AES-CCM (KTS-Wrap) Kernel

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		192, 256 bits				AES-GCM (KTS-Wrap) Kernel AES CBC with HMAC Kernel AES CTR with HMAC Kernel AES-ECB Kernel AES-CTR Kernel AES-CBC Kernel AES-CBC-CS3 Kernel AES-XTS Kernel AES-CCM (BC-Auth) Kernel AES-GCM (BC-Auth) Kernel AES-CMAC Kernel AES-GMAC
Kernel HMAC key	HMAC key	112-256 bits - 112-256 bits	Authentication key - CSP			Kernel AES CBC with HMAC Kernel AES CTR with HMAC Kernel HMAC
Kernel Intermediate Key	Intermediate key generation value	P-256, P-384 - 128, 192 bits	Intermediate value - CSP	Kernel ECDSA Key		Kernel ECDSA Key

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Generation Value				Generation		Generation
Kernel shared secret	Shared secret generated by ECDH	P-256, P-384 - 128 and 192 bits	Shared secret - CSP		Kernel KAS-ECC-SSC	
Kernel DRBG seed	DRBG seed derived from entropy input	128, 192, 256 bits - 128, 192, 256 bits	Seed - CSP	Kernel Counter DRBG		Kernel Counter DRBG
Kernel EC public key	Public key used for ECDH	P-256, P-384 - 128, 192 bits	Public key - PSP	Kernel ECDSA Key Generation		Kernel KAS-ECC-SSC
Kernel EC private key	Private key used for ECDH	P-256, P-384 - 128, 192 bits	Private key - CSP	Kernel ECDSA Key Generation		Kernel KAS-ECC-SSC
Entropy input	Entropy input used to seed the DRBGs	320 bits - 256 bits	Entropy input - CSP			Kernel Counter DRBG FIPS provider Counter DRBG FIPS provider Hash DRBG FIPS provider HMAC DRBG
DRBG Internal State (V, Key)	Internal state of Counter DRBG	Counter DRBG: 128, 192, 256 bits; HMAC DRBG: 160-512 bits - 128-256 bits	DRBG Internal state - CSP	Kernel Counter DRBG FIPS provider Counter DRBG FIPS provider HMAC DRBG		Kernel Counter DRBG FIPS provider Counter DRBG FIPS provider HMAC DRBG

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
DRBG Internal state (V, C)	Internal state of Hash and HMAC DRBG	440, 888 bits - 256 bits	DRBG Internal state - CSP	FIPS provider Hash DRBG		FIPS provider Hash DRBG
FIPS provider AES Key	AES key used for encryption, decryption, and computing MAC tags	128, 192, 256 bits - 128, 192, 256 bits	Symmetric Key - CSP			FIPS provider AES-CCM (KTS-Wrap) FIPS provider AES-GCM (KTS-Wrap) FIPS provider AES KWP FIPS provider AES-ECB FIPS provider AES-CTR FIPS provider AES-CBC FIPS provider AES-CBC-CS1 FIPS provider AES-CBC-CS2 FIPS provider AES-CBC-CS3 FIPS provider AES-CFB1 FIPS provider AES-CFB8 FIPS provider AES-XTS FIPS provider

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
						AES-CCM (BC-Auth) FIPS provider AES-GCM (BC-Auth) FIPS provider AES-OFB FIPS provider AES-CMAC FIPS provider AES-GMAC FIPS provider ANS 9.42 Key Derivation (CVL) FIPS provider KBKDF Key Derivation
FIPS provider HMAC key	HMAC key	112-256 bits - 112-256 bits	Authentication key - CSP			FIPS provider HMAC
FIPS provider shared secret	Shared secret generated by KAS-FFC-SSC, KAS-ECC-SSC, or KAS-IFC-SSC	224-8192 bits - 112-256 bits	Shared secret - CSP		FIPS provider KTS-IFC FIPS provider KAS-FFC-SSC FIPS provider KAS-ECC-SSC	FIPS provider ANS 9.42 Key Derivation (CVL) FIPS provider ANS 9.63 Key Derivation (CVL) FIPS provider TLS 1.0 and 1.1

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
						Key Derivation (CVL) FIPS provider TLS 1.2 Key Derivation (CVL) FIPS provider TLS 1.3 Key Derivation (CVL) FIPS provider HKDF Key Derivation FIPS provider OneStep Key Derivation FIPS provider TwoStep Key Derivation FIPS provider SSH Key Derivation
FIPS provider DRBG seed	DRBG seed derived from entropy input	128,192, 256 bits - 128, 192, 256 bits	Seed - CSP			FIPS provider Counter DRBG FIPS provider Hash DRBG FIPS provider HMAC DRBG
FIPS provider	Public key used for DH	2048, 3072,	Public key - PSP			FIPS provider

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
DH public key		4096, 6144, 8192 bits - 112-200 bits				KAS-FFC-SSC
FIPS provider DH private key	Private key used for DH	2048, 3072, 4096, 6144, 8192 bits - 112-200 bits	Private key - CSP			FIPS provider KAS-FFC-SSC
FIPS provider EC public key	Public key used for ECDH and ECDSA	P-224, P-256, P-384, P-521; Ed25519, Ed448 - 112, 128, 192, 256 bits	Public key - PSP			FIPS provider KAS-ECC-SSC FIPS provider ECDSA Key Verification FIPS provider ECDSA Signature Verification FIPS provider EDDSA Signature Verification
FIPS provider EC private key	Private key used for ECDH and ECDSA	P-224, P-256, P-384, P-521; Ed25519, Ed448 - 112, 128, 192, 256 bits	Private key - CSP			FIPS provider KAS-ECC-SSC FIPS provider ECDSA Signature Generation FIPS provider EDDSA

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
						Signature Generation
FIPS provider module generated DH public key	DH public key generated by the module	2048, 3072, 4096, 6144, 8192 bits - 112-200 bits	Public key - PSP	FIPS provider Safe Primes Key Generation		
FIPS provider module generated DH private key	DH private key generated by the module	2048, 3072, 4096, 6144, 8192 bits - 112-200 bits	Private key - CSP	FIPS provider Safe Primes Key Generation		
FIPS provider module generated EC public key	EC public key generated by the module	P-224, P-256, P-384, P-521; Ed25519, Ed448 - 128-256 bits	Public key - PSP	FIPS provider ECDSA Key Generation FIPS provider EDDSA Key Generation		
FIPS provider module generated EC private key	EC private key generated by the module	P-224, P-256, P-384, P-521; Ed25519, Ed448 (128, 192 bits) - 128-256 bits	Private key - CSP	FIPS provider ECDSA Key Generation FIPS provider EDDSA Key Generation		
FIPS provider RSA public key	Public key used for RSA signature generation	2048, 3072, 4096 bits - 112, 128, 150 bits	Public key - PSP			FIPS provider KAS-IFC-SSC FIPS provider KTS-IFC FIPS

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
						provider RSA Signature Verification
FIPS provider RSA private key	Private key used for RSA signature generation	2048, 3072, 4096 bits - 112, 128, 150 bits	Private key - CSP			FIPS provider KAS-IFC-SSC FIPS provider KTS-IFC FIPS provider RSA Signature Generation
FIPS provider module generated RSA public key	RSA public key generated by the module	2048, 3072, 4096 bits - 112, 128, 150 bits	Public key - PSP	FIPS provider RSA Key Generation		
FIPS provider module generated RSA private key	RSA private key generated by the module	2048, 3072, 4096 bits - 112, 128, 150 bits	Private key - CSP	FIPS provider RSA Key Generation		
FIPS provider Intermediate Key Generation Value	Intermediate key generation value	224-4096 bits - 112-256 bits	Intermediate value - CSP	FIPS provider Safe Primes Key Generation FIPS provider ECDSA Key Generation FIPS provider EDDSA Key Generati		FIPS provider Safe Primes Key Generation FIPS provider ECDSA Key Generation FIPS provider EDDSA Key Generati

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
				on FIPS provider RSA Key Generation		on FIPS provider RSA Key Generation
FIPS provider derived key	Symmetric key derived from a key-derivation key , shared secret, or password	112-4096 bits - 112-256 bits	Symmetric key - CSP	FIPS provider ANS 9.42 Key Derivation (CVL) FIPS provider ANS 9.63 Key Derivation (CVL) FIPS provider TLS 1.0 and 1.1 Key Derivation (CVL) FIPS provider TLS 1.2 Key Derivation (CVL) FIPS provider TLS 1.3 Key Derivation (CVL) FIPS provider HKDF Key Derivation FIPS provider Password-based Key Derivation FIPS provider		

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
				OneStep Key Derivation FIPS provider TwoStep Key Derivation FIPS provider KMAC Key Derivation FIPS provider KBKDF Key Derivation FIPS provider SSH Key Derivation		
FIPS provider key-derivation key	Symmetric key used to derive symmetric keys	112-4096 bits - 112-256 bits	Symmetric key - CSP			FIPS provider KMAC Key Derivation FIPS provider KBKDF Key Derivation
FIPS provider Password	Password used to derive symmetric keys	8-128 characters - N/A	Password - CSP			FIPS provider Password-based Key Derivation
FIPS provider AES Derived Key	AES key used for encryption, decryption, and computing MAC tags	128, 192, 256 bits - 128, 192, 256 bits	Symmetric Key - CSP	FIPS provider TLS 1.0 and 1.1 Key Derivatio		

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
				n (CVL) FIPS provider TLS 1.2 Key Derivation (CVL) FIPS provider TLS 1.3 Key Derivation (CVL) FIPS provider KBKDF Key Derivation		
FIPS provider HMAC Derived Key	HMAC key	112-256 bits - 112-256 bits	Authentication Key - CSP	FIPS provider TLS 1.0 and 1.1 Key Derivation (CVL) FIPS provider TLS 1.2 Key Derivation (CVL) FIPS provider TLS 1.3 Key Derivation (CVL) FIPS provider KBKDF Key Derivation		
FIPS provider 802.11 Pre-shared key (PSK)	Used for pre-shared key authentication and session key establishment, as well as for 802.11 KDF	Up to 256 bits of length - Up to 256 bits	Pre-shared key - CSP			FIPS provider KBKDF Key Derivation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
FIPS provider 802.11 Pairwise Master Key (PMK)	Used for pre-shared key authentication and session key establishment, as well as for 802.11 KDF	256 or 384 bits - 256 bits	Pairwise Master Key - CSP			FIPS provider KBKDF Key Derivation
FIPS provider 802.11 KDF Internal State	Used for SP800-108 KDF to calculate the WPA2 session keys	N/A - N/A	Internal state - CSP	FIPS provider KBKDF Key Derivation		FIPS provider KBKDF Key Derivation
FIPS provider 802.11 Temporal Keys	AES-CCM or AES-GCM keys used for session encryption/decryption	128 or 256 bits - 128 or 256 bits	Temporal Keys - CSP	FIPS provider KBKDF Key Derivation		Kernel AES-CCM (BC-Auth) Kernel AES-GCM (BC-Auth)
FIPS provider 802.11 MIC keys (KCK)	Key confirmation keys (KCK) used for message authentication during session establishment	128 or 192 bits - 128 or 192 bits	MIC keys - CSP			FIPS provider KBKDF Key Derivation
FIPS provider 802.11 Key Encryption Key (KEK)	Used for AES Key Wrapping of the 802.11 Group Temporal Key (GTK)	128 or 256 bits - 128 or 256 bits	Key Encryption Key - CSP		Kernel AES-CBC Kernel AES-CCM (BC-Auth) Kernel AES-GCM (BC-Auth)	Kernel AES-CBC Kernel AES-CCM (BC-Auth) Kernel AES-GCM (BC-Auth)
FIPS provider 802.11 Group Temporal Key (GTK)	802.11 session key for broadcast communications	128 to 256 bits - 128 to 256 bits	Group Temporal Key - CSP		Kernel AES-CBC Kernel AES-CCM (BC-Auth) Kernel AES-GCM (BC-Auth)	Kernel AES-CBC Kernel AES-CCM (BC-Auth) Kernel AES-GCM (BC-Auth)

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
FIPS Provider KMAC Key	KMAC key	112-256 bits - 112-256 bits	Authentication key - CSP			FIPS provider KMAC

Table 18: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
Kernel AES key	API input parameters Kernel AF_ALG_type sockets (input)	RAM:Plaintext	For the duration of the service	Kernel free cipher handle Remove power from the module	
Kernel HMAC key	API input parameters Kernel AF_ALG_type sockets (input)	RAM:Plaintext	For the duration of the service	Kernel free cipher handle Remove power from the module	
Kernel Intermediate Key Generation Value		RAM:Plaintext	For the duration of the service	Kernel free cipher handle Remove power from the module	Kernel EC public key:Generates Kernel EC private key:Generates
Kernel shared secret	API output parameters Kernel AF_ALG type sockets (output)	RAM:Plaintext	For the duration of the service	Kernel free cipher handle Remove power from the module	Kernel EC public key:Used With Kernel EC private key:Used With
Kernel DRBG seed		RAM:Plaintext	While the DRBG is being instantiated	Kernel free cipher handle Remove power from the module	Entropy input:Derived From DRBG Internal State (V, Key):Generates
Kernel EC public key	API input parameters Kernel AF_ALG_type sockets (input) API output parameters Kernel AF_ALG type	RAM:Plaintext	For the duration of the service	Kernel free cipher handle Remove power from the module	Kernel EC private key:Paired With Kernel Intermediate Key Generation Value:Generated from

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
	sockets (output)				
Kernel EC private key	API input parameters Kernel AF_ALG_type sockets (input) API output parameters Kernel AF_ALG type sockets (output)	RAM:Plaintext	For the duration of the service	Kernel free cipher handle Remove power from the module	Kernel EC public key:Paired With Kernel Intermediate Key Generation Value:Generated from
Entropy input		RAM:Plaintext	From generation until DRBG seed is created	Kernel free cipher handle FIPS provider calling the zeroization API Remove power from the module	Kernel DRBG seed:Derives FIPS provider DRBG seed:Derives
DRBG Internal State (V, Key)		RAM:Plaintext	From DRBG instantiation until DRBG termination	Kernel free cipher handle FIPS provider calling the zeroization API Remove power from the module	Kernel DRBG seed:Generated from FIPS provider DRBG seed:Generated from
DRBG Internal state (V, C)		RAM:Plaintext	From DRBG instantiation until DRBG termination	FIPS provider calling the zeroization API Remove power from the module	Kernel DRBG seed:Generated from FIPS provider DRBG seed:Generated from
FIPS provider AES Key	API input parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
FIPS provider HMAC key	API input parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	
FIPS provider shared secret	API output parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	FIPS provider DH public key:Established by FIPS provider DH private key:Established by FIPS provider EC public key:Established by FIPS provider EC private key:Established by FIPS provider derived key:Derives FIPS provider RSA public key:Established by FIPS provider RSA private key:Established by
FIPS provider DRBG seed		RAM:Plaintext	While the DRBG is being instantiated	FIPS provider calling the zeroization API Remove power from the module	Entropy input:Derived From DRBG Internal State (V, Key):Generates DRBG Internal state (V, C):Generates
FIPS provider DH public key	API input parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API	FIPS provider DH private key:Paired With FIPS provider Intermediate Key Generation Value:Generated from

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
FIPS provider DH private key	API input parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API	FIPS provider DH public key:Paired With FIPS provider Intermediate Key Generation Value:Generated from
FIPS provider EC public key	API input parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	FIPS provider EC private key:Paired With FIPS provider Intermediate Key Generation Value:Generated from
FIPS provider EC private key	API input parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	FIPS provider EC public key:Paired With FIPS provider Intermediate Key Generation Value:Generated from
FIPS provider module generated DH public key	API output parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	FIPS provider module generated DH private key:Paired With FIPS provider Intermediate Key Generation Value:Generated from
FIPS provider module generated DH private key	API output parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	FIPS provider module generated DH public key:Paired With FIPS provider Intermediate Key Generation Value:Generated from
FIPS provider module generated EC public key	API output parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	FIPS provider module generated EC private key:Paired With FIPS provider Intermediate Key Generation

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
					Value:Generated from
FIPS provider module generated EC private key	API output parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	FIPS provider module generated EC public key:Paired With FIPS provider Intermediate Key Generation Value:Generated from
FIPS provider RSA public key	API input parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	FIPS provider RSA private key:Paired With FIPS provider Intermediate Key Generation Value:Generated from FIPS provider shared secret:Establishes
FIPS provider RSA private key	API input parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	FIPS provider RSA public key:Paired With FIPS provider Intermediate Key Generation Value:Generated from FIPS provider shared secret:Establishes
FIPS provider module generated RSA public key	API output parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	FIPS provider module generated RSA private key:Paired With FIPS provider Intermediate Key Generation Value:Generated from
FIPS provider module generated RSA private key	API output parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove	FIPS provider module generated RSA public key:Paired With FIPS provider Intermediate Key

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
				power from the module	Generation Value:Generated from
FIPS provider Intermediate Key Generation Value		RAM:Plaintext	For the duration of the service	FIPS provider Automatic	FIPS provider module generated DH public key:Generates FIPS provider DH private key:Generates FIPS provider module generated DH public key:Generates FIPS provider module generated DH private key:Generates FIPS provider EC public key:Generates FIPS provider EC private key:Generates FIPS provider module generated EC public key:Generates FIPS provider module generated EC private key:Generates FIPS provider RSA public key:Generates FIPS provider RSA private key:Generates FIPS provider module generated RSA public key:Generates FIPS provider module generated RSA private key:Generates

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
FIPS provider derived key	API output parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	FIPS provider key-derivation key:Derived From FIPS provider shared secret:Derived From FIPS provider password:Derived From
FIPS provider key-derivation key	API input parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	FIPS provider derived key:Derives
FIPS provider Password	API input parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	FIPS provider derived key:Derives
FIPS provider AES Derived Key	API output parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	FIPS provider derived key:Derives
FIPS provider HMAC Derived Key	API output parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	FIPS provider derived key:Derives
FIPS provider 802.11 Pre-shared key (PSK)	API input parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	FIPS provider derived key:Used With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
FIPS provider 802.11 Pairwise Master Key (PMK)	API input parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	FIPS provider derived key:Used With
FIPS provider 802.11 KDF Internal State			For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	
FIPS provider 802.11 Temporal Keys	API input parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	Kernel AES key:Encrypts Kernel AES key:Decrypts
FIPS provider 802.11 MIC keys (KCK)	API input parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	
FIPS provider 802.11 Key Encryption Key (KEK)	API input parameters	RAM:Plaintext	For the duration of the service	Kernel free cipher handle Remove power from the module	Kernel AES key:Encrypts
FIPS provider 802.11 Group Temporal Key (GTK)	API output parameters	RAM:Plaintext	For the duration of the service	Kernel free cipher handle Remove power from the module	Kernel AES key:Encrypts Kernel AES key:Decrypts
FIPS Provider KMAC Key	API input parameters	RAM:Plaintext	For the duration of the service	FIPS provider calling the zeroization API Remove power from the module	

Table 19: SSP Table 2

9.5 Transitions

The SHA-1 algorithm as implemented by the module will be non-approved for all purposes, starting January 1, 2030.

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A5009)	256-bit key	Message Authentication	SW/FW Integrity	Module becomes operational and services are available for use	Integrity test for fips.so; Integrity test for kernel binary; Integrity test for fipsheck binary; Integrity test for fipscheck library

Table 20: Pre-Operational Self-Tests

The pre-operational firmware integrity tests are performed automatically when the module is powered on, before the module transitions into the operational state. The algorithm used for the integrity test (i.e., HMAC-SHA2-256) is self-tested before the firmware integrity test is performed. While the module is executing the self-tests, services are not available, and data output (via the data output interface) is inhibited until the tests are successfully completed. The module transitions to the operational state only after the pre-operational self-tests are passed successfully.

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
ECDSA KeyGen (FIPS186-5) (A4711)	N/A	PCT	PCT	crypto_kpp_generate_public_key returns 0	SP 800-56Ar3 Section 5.6.2.1.4	Key pair generation
SHA2-224 (A4711)	0-8184 bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA2-256 (A4711)	0-8184 bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA2-384 (A4711)	0-8184 bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA2-512 (A4711)	0-8184 bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA3-224 (A4713)	0-8184 bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA3-256 (A4713)	0-8184 bit messages	KAT	CAST	Module is operational	Message digest	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
SHA3-384 (A4713)	0-8184 bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA3-512 (A4713)	0-8184 bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
AES-ECB (A4711)	128, 192, 256 bit keys	KAT	CAST	Module is operational	Encryption, Decryption (Separately)	Module initialization
AES-CBC (A4712)	128, 192, 256 bit keys	KAT	CAST	Module is operational	Encryption, Decryption (Separately)	Module initialization
AES-CTR (A4712)	128, 192, 256 bit keys	KAT	CAST	Module is operational	Encryption, Decryption (Separately)	Module initialization
AES-CCM (A4712)	128, 192, 256 bit keys	KAT	CAST	Module is operational	Encryption, Decryption (Separately)	Module initialization
AES-GCM (A4712)	128, 192, 256 bit keys	KAT	CAST	Module is operational	Message authentication	Module initialization
AES-CMAC (A4712)	128 and 256 bit keys	KAT	CAST	Module is operational	Message authentication	Module initialization
KAS-ECC-SSC Sp800-56Ar3 (A4711)	P-256, P-384	KAT	CAST	Module is operational	Shared secret computation	Module initialization
Counter DRBG (A4711)	128, 192, 256 bit keys with/without PR; Health test per section 11.3 of SP 800-90A	KAT	CAST	Module is operational	Seed Generate	Module initialization
ECDSA KeyGen (FIPS186-5) (A5009)	SHA2-256	PCT	PCT	Successful key generation	Signature generation and verification	EC key pair generation
RSA KeyGen (FIPS186-5) (A5018)	PKCS#1 v1.5 with SHA2-256	PCT	PCT	Successful key generation	Signature generation and verification	RSA key pair generation
Safe Primes Key	N/A	PCT	PCT	Successful key generation	Public key re-computation and	Safe Primes key

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
Generation (A5014)					comparison with the existing public key (per SP 800-56Ar3 Section 5.6.2.1.4)	pair generation
EDDSA KeyGen (A5016)	ED25519 and ED448	PCT	PCT	Successful key generation	Signature generation and verification	EDDSA key pair generation
SHA-1 (A5009)	24-bit message	KAT	CAST	Module is operational	Message Digest	Module initialization
SHA2-512 (A5009)	24-bit message	KAT	CAST	Module is operational	Message Digest	Module initialization
SHA3-256 (A5011)	32-bit message	KAT	CAST	Module is operational	Message Digest	Module initialization
AES-ECB (A5002)	128-bit keys, 128-bit ciphertext	KAT	CAST	Module is operational	Decryption	Module initialization
AES-GCM (A5005)	256-bit keys, 96-bit IVs, 128-bit plaintext, 128-bit additional data	KAT	CAST	Module is operational	Encryption, Decryption (Separately)	Module initialization
KDF SP800-108 (A5017)	Counter mode, HMAC-SHA2-256, 128-bit input key	KAT	CAST	Module is operational	Key Derivation	Module initialization
KDA OneStep SP800-56Cr2 (A5012)	SHA-224, 392-bit input secret	KAT	CAST	Module is operational	Key Derivation	Module initialization
KDA HKDF Sp800-56Cr1 (A5013)	SHA-256, 48-bit input secret	KAT	CAST	Module is operational	Key Derivation	Module initialization
KDF ANS 9.42 (A5009)	SHA-1 with AES-128, KW, 160-	KAT	CAST	Module is operational	Key Derivation	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
	bit input secret					
KDF ANS 9.63 (A5009)	SHA-256, 192-bit input secret	KAT	CAST	Module is operational	Key Derivation	Module initialization
KDF SSH (A5019)	SHA-1, 1056-bit input secret	KAT	CAST	Module is operational	Key Derivation	Module initialization
TLS v1.2 KDF RFC7627 (A5009)	SHA-256, 84-bit input secret	KAT	CAST	Module is operational	Key Derivation	Module initialization
TLS v1.3 KDF (A5013)	Extract and expand modes, SHA-256	KAT	CAST	Module is operational	Key Derivation	Module initialization
PBKDF (A5009)	SHA-256, 24-character password, 288-bit salt, Iteration count: 4096	KAT	CAST	Module is operational	Key Derivation	Module initialization
Counter DRBG (A5015)	AES-128 with prediction resistance	KAT	CAST	Module is operational	Instantiate, Generate, Reseed, Generate (compliant with SP 800-90Ar1 Section 11.3)	Module initialization
HMAC DRBG (A5015)	SHA-1 with prediction resistance	KAT	CAST	Module is operational	Instantiate, Generate, Reseed, Generate (compliant with SP 800-90Ar1 Section 11.3)	Module initialization
Hash DRBG (A5015)	SHA-256 with prediction resistance	KAT	CAST	Module is operational	Instantiate, Generate, Reseed, Generate (compliant with SP 800-90Ar1 Section 11.3)	Module initialization

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
KAS-FFC-SSC Sp800-56Ar3 (A5014)	ffdhe2048	KAT	CAST	Module is operational	Shared Secret Computation	Module initialization
KAS-ECC-SSC Sp800-56Ar3 (A5009)	P-256	KAT	CAST	Module is operational	Shared Secret Computation	Module initialization
RSA SigGen (FIPS186-5) (A5009)	PKCS#1 v1.5 with SHA-256 and 2048-bit key	KAT	CAST	Module is operational	Signature Generation	Module initialization
RSA SigVer (FIPS186-5) (A5009)	PKCS#1 v1.5 with SHA-256 and 2048-bit key	KAT	CAST	Module is operational	Signature Verification	Module initialization
ECDSA SigGen (FIPS186-5) (A5009)	SHA-256 and P-224, P-256, P-384, and P-521	KAT	CAST	Module is operational	Signature Generation	Module initialization
ECDSA SigVer (FIPS186-5) (A5009)	SHA-256 and P-224, P-256, P-384, and P-521	KAT	CAST	Module is operational	Signature Verification	Module initialization
EDDSA SigGen (A5016)	ED25519 and ED448	KAT	CAST	Module is operational	Signature Generation	Module initialization
EDDSA SigVer (A5016)	ED25519 and ED448	KAT	CAST	Module is operational	Signature Verification	Module initialization
KTS-IFC (A5018)	SHA-256, no padding	KAT	CAST	Module is operational	Decryption	Module initialization

Table 21: Conditional Self-Tests

The module performs self-tests on all approved cryptographic algorithms as part of the approved services supported in the approved mode of operation, using the tests shown in in the table above. Services are not available, and data output (via the data output interface) is inhibited during the self-tests. If any of these tests fails, the module transitions to the error state.

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A5009)	Message Authentication	SW/FW Integrity	On-demand	Manually

Table 22: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
ECDSA KeyGen (FIPS186-5) (A4711)	PCT	PCT	On demand	Manually
SHA2-224 (A4711)	KAT	CAST	On demand	Manually
SHA2-256 (A4711)	KAT	CAST	On demand	Manually
SHA2-384 (A4711)	KAT	CAST	On demand	Manually
SHA2-512 (A4711)	KAT	CAST	On demand	Manually
SHA3-224 (A4713)	KAT	CAST	On demand	Manually
SHA3-256 (A4713)	KAT	CAST	On demand	Manually
SHA3-384 (A4713)	KAT	CAST	On demand	Manually
SHA3-512 (A4713)	KAT	CAST	On demand	Manually
AES-ECB (A4711)	KAT	CAST	On demand	Manually
AES-CBC (A4712)	KAT	CAST	On demand	Manually
AES-CTR (A4712)	KAT	CAST	On demand	Manually
AES-CCM (A4712)	KAT	CAST	On demand	Manually
AES-GCM (A4712)	KAT	CAST	On demand	Manually
AES-CMAC (A4712)	KAT	CAST	On demand	Manually
KAS-ECC-SSC Sp800-56Ar3 (A4711)	KAT	CAST	On demand	Manually
Counter DRBG (A4711)	KAT	CAST	On demand	Manually
ECDSA KeyGen (FIPS186-5) (A5009)	PCT	PCT	On demand	Manually
RSA KeyGen (FIPS186-5) (A5018)	PCT	PCT	On demand	Manually
Safe Primes Key Generation (A5014)	PCT	PCT	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
EDDSA KeyGen (A5016)	PCT	PCT	On demand	Manually
SHA-1 (A5009)	KAT	CAST	On demand	Manually
SHA2-512 (A5009)	KAT	CAST	On demand	Manually
SHA3-256 (A5011)	KAT	CAST	On demand	Manually
AES-ECB (A5002)	KAT	CAST	On demand	Manually
AES-GCM (A5005)	KAT	CAST	On demand	Manually
KDF SP800-108 (A5017)	KAT	CAST	On demand	Manually
KDA OneStep SP800-56Cr2 (A5012)	KAT	CAST	On demand	Manually
KDA HKDF Sp800-56Cr1 (A5013)	KAT	CAST	On demand	Manually
KDF ANS 9.42 (A5009)	KAT	CAST	On demand	Manually
KDF ANS 9.63 (A5009)	KAT	CAST	On demand	Manually
KDF SSH (A5019)	KAT	CAST	On demand	Manually
TLS v1.2 KDF RFC7627 (A5009)	KAT	CAST	On demand	Manually
TLS v1.3 KDF (A5013)	KAT	CAST	On demand	Manually
PBKDF (A5009)	KAT	CAST	On demand	Manually
Counter DRBG (A5015)	KAT	CAST	On demand	Manually
HMAC DRBG (A5015)	KAT	CAST	On demand	Manually
Hash DRBG (A5015)	KAT	CAST	On demand	Manually
KAS-FFC-SSC Sp800-56Ar3 (A5014)	KAT	CAST	On demand	Manually
KAS-ECC-SSC Sp800-56Ar3 (A5009)	KAT	CAST	On demand	Manually
RSA SigGen (FIPS186-5) (A5009)	KAT	CAST	On demand	Manually
RSA SigVer (FIPS186-5) (A5009)	KAT	CAST	On demand	Manually
ECDSA SigGen (FIPS186-5) (A5009)	KAT	CAST	On demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
ECDSA SigVer (FIPS186-5) (A5009)	KAT	CAST	On demand	Manually
EDDSA SigGen (A5016)	KAT	CAST	On demand	Manually
EDDSA SigVer (A5016)	KAT	CAST	On demand	Manually
KTS-IFC (A5018)	KAT	CAST	On demand	Manually

Table 23: Conditional Periodic Information

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Error State	The module immediately stops functioning due to a self-test failure	Integrity test failure CAST Failure PCT Failure	Reboot and successful completion of self-tests	Module reboots

Table 24: Error States

In the error state, the output interface is inhibited, and the module accepts no more inputs or requests (as the module is no longer running).

10.5 Operator Initiation of Self-Tests

All self-tests, with the exception of the continuous health tests, can be invoked on demand by unloading and subsequently re-initializing the module.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

Before deploying the module for usage, the Crypto Officer shall employ the following steps:

1. Verify the HMAC values of each component of the module as listed in section 2.2.
2. Verify that the kernel component command line is configured to run `fipsInit.sh` before any user mode application or init system.
3. Verify that `'fips=1'` parameter is present on the kernel command line for approved mode operation.

11.2 Administrator Guidance

The Crypto Officer must execute the “cat /proc/sys/crypto/fips_name” command. The Crypto Officer must ensure that the proper name is listed in the output as follows:

Summit Linux

This output maps to the module name “Summit Linux FIPS Core Crypto Module”.

Next the Crypto Officer must execute “cat /proc/sys/crypto/fips_version”. This command must output the following:

11.0

The following are the HMAC values for each of the module components:

- wb45n/Image.lzma.hmac:3df472065e70a8c176c976c0588170e89b816d3ca1caee5f734ea0b187696e13
- wb45n/libfipscheck.so.1.hmac:d193a7627cb9f9b7bae33a1bba2c57fdb52a42710a914ec57122590a5e63a65c
- wb45n/fipscheck.hmac:1e5bfd65b7a5d351e6cd1445df5cff7dddd62e8d9097ec065bcb877188a5271f
- wb45n/fips.so.hmac:f2eae64d2d5ae6b84995e671d25aba0672408f7b1b247c45a89cb817276c30f9

11.3 Non-Administrator Guidance

There is no non-administrator guidance.

11.4 End of Life

As the module does not persistently store SSPs, secure sanitization of the module consists of unloading the module. This will zeroize all SSPs in volatile memory.

12 Mitigation of Other Attacks

12.1 Attack List

For the FIPS Provider component, certain cryptographic subroutines and algorithms are vulnerable to timing analysis. The FIPS Provider component mitigates this vulnerability by using constant-time implementations. This includes, but is not limited to:

- Big number operations: computing GCDs, modular inversion, multiplication, division, and modular exponentiation (using Montgomery multiplication).
- Elliptic curve point arithmetic: addition and multiplication (using the Montgomery ladder).
- Vector-based AES implementations.
- In addition, RSA, ECDSA, ECDH, and DH employ blinding techniques to further impede timing and power analysis.

No configuration is needed to enable the aforementioned countermeasures.

Appendix A. Glossary and Abbreviations

AES	Advanced Encryption Standard
API	Application Programming Interface
CAST	Cryptographic Algorithm Self-Test
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cipher Feedback
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter
CTS	Ciphertext Stealing
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EMS	Extended Master Secret
ENT (NP)	Non-physical Entropy Source
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standards
GCM	Galois Counter Mode
GMAC	Galois Counter Mode Message Authentication Code
HKDF	HMAC-based Key Derivation Function
HMAC	Keyed-Hash Message Authentication Code
IPsec	Internet Protocol Security
KAT	Known Answer Test
KBKDF	Key-based Key Derivation Function
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
PAA	Processor Algorithm Acceleration
PBKDF2	Password-based Key Derivation Function v2
PKCS	Public-Key Cryptography Standards
RSA	Rivest, Shamir, Addleman
SFI	Security Function Implementation
SHA	Secure Hash Algorithm
SSC	Shared Secret Computation
SSP	Sensitive Security Parameter
TOEPP	Test Operational Environment’s Physical Perimeter
XTS	XEX-based Tweaked-codebook mode with cipher text Stealing

Appendix B. References

ANS X9.42-2001	**Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography** 2001 https://webstore.ansi.org/standards/ascx9/ansix9422001
ANS X9.63-2001	**Public Key Cryptography for the Financial Services Industry, Key Agreement and Key Transport Using Elliptic Curve Cryptography** 2001 https://webstore.ansi.org/standards/ascx9/ansix9632001
FIPS 140-3	FIPS PUB 140-3 - Security Requirements For Cryptographic Modules March 2019 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-3.pdf
FIPS 140-3 IG	Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program https://csrc.nist.gov/Projects/cryptographic-module-validation-program/fips-140-3-ig-announcements
FIPS 180-4	**Secure Hash Standard (SHS)** March 2012 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
FIPS 186-5	**Digital Signature Standard (DSS)** February 3, 2023 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf
FIPS 197	**Advanced Encryption Standard** November 2001 https://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
FIPS 198-1	**The Keyed Hash Message Authentication Code (HMAC)** July 2008 https://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf
FIPS 202	**SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions** August 2015 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf
PKCS#1	**Public Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1** February 2003 https://www.ietf.org/rfc/rfc3447.txt
RFC 3526	**More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)** May 2003 https://www.ietf.org/rfc/rfc3526.txt
RFC 5288	**AES Galois Counter Mode (GCM) Cipher Suites for TLS** August 2008 https://www.ietf.org/rfc/rfc5288.txt

RFC 7919 **Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)**
August 2016
https://www.ietf.org/rfc/rfc7919.txt

RFC 8446 **The Transport Layer Security (TLS) Protocol Version 1.3**
August 2018
https://www.ietf.org/rfc/rfc8446.txt

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

SP 800-38A Addendum **Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode**
October 2010
https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a-add.pdf

SP 800-38B **Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication**
May 2005
https://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf

SP 800-38C **Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality**
May 2004
https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf

SP 800-38D Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC
November 2007
https://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf

SP 800-38E **Recommendation for Block Cipher Modes of Operation: The XTS AES Mode for Confidentiality on Storage Devices**
January 2010
https://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf

SP 800-38F **Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping**
December 2012
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf

SP 800-52r2 **Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations**
August 2019
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r2.pdf

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

SP 800-56Cr2 **Recommendation for Key-Derivation Methods in Key-Establishment Schemes**
August 2020
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr2.pdf

SP 800-90Ar1	**Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf
SP 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
SP 800-108r1	**NIST Special Publication 800-108 - Recommendation for Key Derivation Using Pseudorandom Functions** August 2022 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-108r1.pdf
SP 800-131Ar2	**Transitioning the Use of Cryptographic Algorithms and Key Lengths** March 2019 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf
SP 800-132	**Recommendation for Password-Based Key Derivation - Part 1: Storage Applications** December 2010 https://csrc.nist.gov/publications/nistpubs/800-132/nist-sp800-132.pdf
SP 800-133r2	**Recommendation for Cryptographic Key Generation** June 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r2.pdf
SP 800-135r1	**Recommendation for Existing Application-Specific Key Derivation Functions** December 2011 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-135r1.pdf
SP 800-140B	CMVP Security Policy Requirements March 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-140B.pdf