FIPS 140-3 Non-Proprietary Security Policy

Oracle Linux 9 OpenSSL FIPS Provider

FIPS 140-3 Level 1 Validation

Software Version: 3.0.7-b27cdeb3ba51be46

Last Updated: 2024-08-06

Prepared by:

atsec information security corporation

4516 Seton Center Pkwy, Suite 250

Austin, TX 78759

www.atsec.com

Title: Oracle Linux 9 OpenSSL FIPS Provider Security Policy

Date: August 8th, 2024

Contributing Authors:
Oracle Linux Engineering
Security Evaluations – Global Product Security
atsec information security

Oracle Corporation
World Headquarters
2300 Oracle Way
Austin, TX 78741
U.S.A.
Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
www.oracle.com

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software, Engineered to Work Together

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 software version 3.0.7-b27cdeb3ba51be46 of the Oracle Linux 9 OpenSSL FIPS Provider. It contains the security rules under which the module must operate and describes how this module meets the requirements as specified in FIPS PUB 140-3 (Federal Information Processing Standards Publication 140-3) for an overall Security Level 1 module.

This Non-Proprietary Security Policy may be reproduced and distributed, but only whole and intact and including this notice. Other documentation is proprietary to their authors.

1.1.1 How this Security Policy Was Prepared

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

Table 1 describes the individual security areas of FIPS 140-3, as well as the security levels of those individual areas.

ISO/IEC 24759 Section 6 Subsections	FIPS 140-3 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	Not Applicable
8	Non-invasive Security	Not Applicable
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 Oracle Linux 9 OpenSSL FIPS Provider (hereafter referred to as “the module”) is defined as a software module in a multi-chip standalone embodiment. It provides a C language application program interface (API) for use by other applications that require cryptographic functionality. The module consists of one software component, the “FIPS provider”, which implements the FIPS requirements, and the cryptographic functionality provided to the operator.

Module Type: Software

Module Embodiment: Multi-chip standalone

Module Characteristics: N/A

Cryptographic Boundary: Figure 1 shows the cryptographic boundary of the module, its interfaces with the operational environment and the flow of information between the module and operator (depicted through the arrows).

Tested Operational Environment’s Physical Perimeter (TOEPP): The TOEPP of the module is defined as the general-purpose computer on which the module is installed.

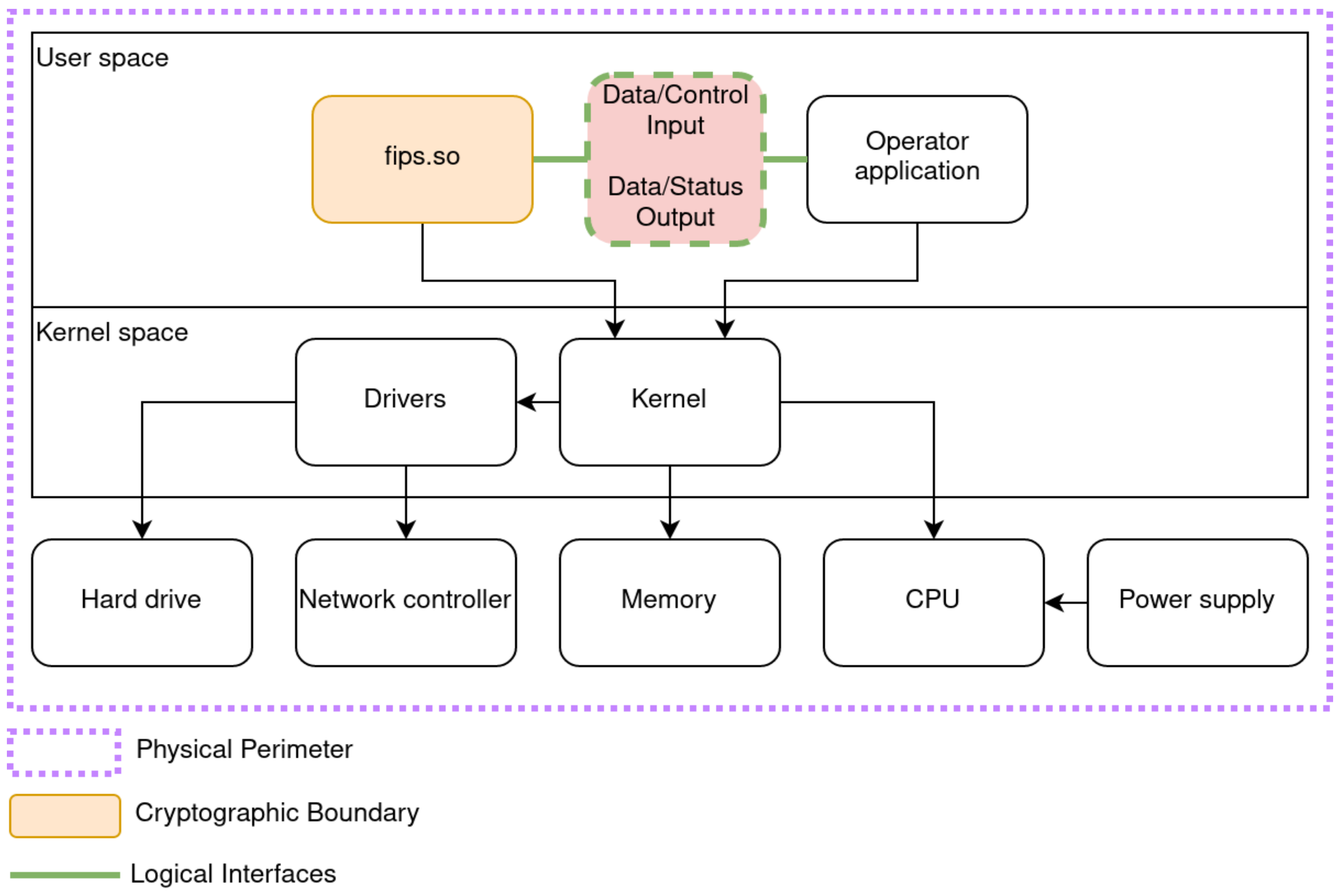

Figure 1 – Block Diagram

2.2 Operating Environments

Hardware Operating Environments: N/A

Software, Firmware, Hybrid Tested Operating Environments:

Operating System	Hardware Platform	Processor(s)	PAA/PAI	Hypervisor and Host OS
Oracle Linux 9	ORACLE SERVER X9-2c	Intel(R) Xeon(R) Platinum 8358	AES-NI and SHA Extensions	KVM on Oracle Linux 8
	ORACLE SERVER E4-2c	AMD EPYC 7J13	AES-NI and SHA Extensions	
	ORACLE SERVER A1-2c	Ampere(R) Altra(R) Q80-30	NEON and Cryptography Extensions (CE)	

Table 2 - Software, Firmware, Hybrid Tested Operating Environments

Executable Code Sets:

Package or File Names	Software/ Firmware Versions	Features	Hybrid Hardware Version	Integrity Test
fips.so	3.0.7-b27cdeb3ba51be46	N/A	N/A	HMAC-SHA-256

Table 3 - Executable Code Sets

Vendor Affirmed Operating Environments:

Operating Systems	Hardware Platforms	Virtual Platforms
Oracle Linux 9	Oracle X Series Servers Oracle E Series Servers Oracle A Series Servers Marvell T93 LiquidIO III (ARM v8.x) SmartNIC Pensando DSC-200-R (ARM v8.x) SmartNIC	Oracle Linux KVM VmWare ESXi

Table 4 - Vendor Affirmed Operational Environments

Note: the CMVP makes no statement as to the correct operation of the module or the security strengths of the generated SSPs 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:

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

Table 5 - 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 indicators:

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.

Degraded Mode Description:

The module does not implement a degraded mode of operation.

2.5 Algorithms

Approved Algorithms:

Algorithm Name	CAVP Numbers	Algorithms Capabilities	OE (Implementation)	Reference
AES-CBC	#A4313, #A4314, #A4315, #A4327, #A4328, #A4329	Encryption, Decryption using 128, 192, 256-bit keys	**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC^{TM} 7001 Series AMD EPYC 7J13:** AESNI, BAES_CTASM, AESASM **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** CE, VPAES, AES_C	FIPS 197, SP 800-38A, SP 800-38A Addendum
AES-CBC-CTS-CS1				
AES-CBC-CTS-CS2				
AES-CBC-CTS-CS3				

Algorithm Name	CAVP Numbers	Algorithms Capabilities	OE (Implementation)	Reference
AES-CCM		Authenticated Encryption, Authenticated Decryption, Key Wrapping, Key Unwrapping (compliant to IG D.G) using 128, 192, 256-bit keys	**Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** AESNI, BAES_CTASM, AESASM	
AES-CFB1		Encryption, Decryption using 128, 192, 256-bit keys		
AES-CFB8				
AES-CFB128				
AES-CMAC		Message Authentication Code Generation, Message Authentication Code Verification using 128, 192, 256-bit keys		FIPS 197, SP 800-38B
AES-CTR		Encryption, Decryption using 128, 192, 256-bit keys		FIPS 197, SP 800-38A, SP 800-38A Addendum
AES-ECB	#A4320, #A4327, #A4328, #A4329, #A4331, #A4332, #A4333, #A4334	Encryption, Decryption using 128, 192, 256-bit keys	**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC™ 7001 Series AMD EPYC 7J13:** SSH_ASM, AESNI, BAES_CTASM, AESASM, SSH_SHANI, SSH_AVX2, SSH_AVX, SSH_SSSE3 **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** SSH_ASM, CE, VPAES, AES_C **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SSH_ASM, AESNI, BAES_CTASM, AESASM, SSH_SHANI, SSH_AVX2, SSH_AVX, SSH_SSSE3	
AES-GCM (internal IV)	#A4316, #A4321, #A4322, #A4323, #A4335, #A4336, #A4337, #A4338, #A4339, #A4340, #A4341, #A4342, #A4343	Authenticated Encryption, Key Wrapping using 128, 192, 256-bit keys	**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC™ 7001 Series AMD EPYC 7J13:** AESNI_AVX, AESNI_CLMULNI, AESNI_ASM, BAES_CTASM_AVX, BAES_CTASM_CLMULNI, BAES_CTASM_ASM, AESASM_AVX, AESASM_CLMULNI, AESASM_ASM **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:**	FIPS 197 SP 800-38D FIPS 140-3 IG D.G Additional comment 8
AES-GCM (external IV)		Authenticated Decryption, Key Unwrapping using		

Algorithm Name	CAVP Numbers	Algorithms Capabilities	OE (Implementation)	Reference
		128, 192, 256-bit keys	CE_GCM_UNROLL8_EOR3, CE_GCM, VPAES_GCM, AES_C_GCM	
AES-GMAC		Message Authentication Code Generation, Message Authentication Code Verification using 128, 192, 256-bit keys	**Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** AESNI_AVX, AESNI_CLMULNI, AESNI_ASM, BAES_CTASM_AVX, BAES_CTASM_CLMULNI, BAES_CTASM_ASM, AESASM_AVX, AESASM_CLMULNI, AESASM_ASM	
AES-OFB	#A4313, #A4314, #A4315, #A4327, #A4328, #A4329	Encryption, Decryption using 128, 192, 256-bit keys	**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC™ 7001 Series AMD EPYC 7J13:** AESNI, BAES_CTASM, AESASM **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** CE, VPAES, AES_C **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** AESNI, BAES_CTASM, AESASM	FIPS 197, SP 800-38A, SP 800-38A Addendum
AES-KW		Key Wrapping, Key Unwrapping (compliant to IG D.G) using 128, 192, 256-bit keys		FIPS 197, SP 800-38F
AES-KWP				
AES-XTS		Encryption, Decryption using 128 and 256-bit keys		FIPS 197, SP 800-38E
ANS X9.42 KDF (CVL) with AES KW-128, AES KW-192, AES KW-256 and SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256,	#A4326, #A4330, #A4344, #A4345, #A4346	Key Derivation	**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC™ 7001 Series AMD EPYC 7J13:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3 **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** SHA_ASM, SHA_CE **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3	SP 800-135r1
ANS X9.42 KDF (CVL) with AES KW-128, AES KW-192, AES KW-256 with SHA3-224, SHA3-256, SHA3-384, SHA3-512	#A4312, #A4319,		**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC™ 7001 Series AMD EPYC 7J13:** SHA3_ASM **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** SHA3_CE **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SHA3_ASM	
ANS X9.63 KDF (CVL) with SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256,	#A4317, #A4326, #A4330, #A4344, #A4345, #A4346		**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC™ 7001 Series AMD EPYC 7J13:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3 **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** SHA_ASM, SHA_CE **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SHA_ASM,	

Algorithm Name	CAVP Numbers	Algorithms Capabilities	OE (Implementation)	Reference
			SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3	
ANS X9.63 KDF (CVL) with SHA3-224, SHA3-256, SHA3-384, SHA3-512	#A4312, #A4319		**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC™ 7001 Series AMD EPYC 7J13:** SHA3_ASM **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** SHA3_CE **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SHA3_ASM	
CTR_DRBG	#A4311	Random Number Generation using 128, 192, 256-bit keys, with/without PR, with/without DF	**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC™ 7001 Series AMD EPYC 7J13:** DRBG_3 **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** DRBG_3 **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** DRBG_3	SP 800-90Ar1
ECDSA with SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	#A4317, #A4326, #A4330, #A4344, #A4345, #A4346	Signature Generation, Signature Verification using P-224, P-256, P-384, P-521 elliptic curves	**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC™ 7001 Series AMD EPYC 7J13:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3, **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** SHA_ASM, SHA_CE **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3,	FIPS 186-4
ECDSA with SHA3-224, SHA3-256, SHA3-384, SHA3-512	#A4312, #A4319	Signature Generation, Signature Verification P-224, P-256, P-384, P-521 elliptic curves	**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC™ 7001 Series AMD EPYC 7J13:** SHA3_ASM **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** SHA3_CE **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SHA3_ASM	
ECDSA	#A4317, #A4326, #A4330, #A4344, #A4345, #A4346	Key Pair Generation using P-224, P-256, P-384, P-521 elliptic curves	**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC™ 7001 Series AMD EPYC 7J13:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3, **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** SHA_ASM, SHA_CE **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3,	FIPS 186-4 Appendix B.4.2 Testing Candidates
		Key Pair Verification using P-224, P-256, P-384, P-521 elliptic curves		FIPS 186-4
HKDF with SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224,	#A4310	Key Derivation	**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC™ 7001 Series AMD EPYC 7J13:** TLS v1.3 **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** TLS v1.3	SP800-56Cr1

Algorithm Name	CAVP Numbers	Algorithms Capabilities	OE (Implementation)	Reference
SHA3-256, SHA3-384, SHA3-512			**Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** TLS v1.3	
HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-384, HMAC-SHA2-512, HMAC-SHA2-512/224, HMAC-SHA2-512/256	#A4317, #A4326, #A4330, #A4344, #A4345, #A4346	Message Authentication Code Generation, Message Authentication Code Verification using 112-524288-bit keys	**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC^{TM} 7001 Series AMD EPYC 7J13:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3, **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** SHA_ASM, SHA_CE **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3	FIPS 198-1
HMAC-SHA2-256	#A4317, #A4318, #A4326, #A4330, #A4344, #A4345, #A4346		**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC^{TM} 7001 Series AMD EPYC 7J13:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3, **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** SHA_ASM, SHA_CE, NEON **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3	
HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512	#A4312, #A4319		**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC^{TM} 7001 Series AMD EPYC 7J13:** SHA3_ASM **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** SHA3_CE **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SHA3_ASM	FIPS 202
HMAC_DRBG	#A4311	Random Number Generation using 112-524288-bit keys, with/without PR	**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC^{TM} 7001 Series AMD EPYC 7J13:** DRBG_3 **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** DRBG_3 **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** DRBG_3	SP 800-90Ar1
Hash_DRBG		Random Number Generation, with/without PR		
KAS-ECC-SSC	#A4317, #A4326, #A4330, #A4344, #A4345, #A4346	Shared Secret Computation with P-256, P-384, P-521 elliptic curves	**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC^{TM} 7001 Series AMD EPYC 7J13:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3, **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** SHA_ASM, SHA_CE **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3,	SP 800-56Ar3 FIPS 140-3 IG D.F scenario 2 path (1)

Algorithm Name	CAVP Numbers	Algorithms Capabilities	OE (Implementation)	Reference
KAS-FFC-SSC	#A4325	Shared Secret Computation with MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192	**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC™ 7001 Series AMD EPYC 7J13:** FFC_DH **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** FFC_DH **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** FFC_DH	
KBKDF with CMAC-AES128, CMAC-AES192, CMAC-AES256 and HMAC SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512	#A4324	Key Derivation	**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC™ 7001 Series AMD EPYC 7J13:** KBKDF **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** KBKDF **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** KBKDF	SP 800-108r1
KDA OneStep[1] with HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512, HMAC-SHA2-512/224, HMAC-SHA2-512/256, HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512	#A4309		**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC™ 7001 Series AMD EPYC 7J13:** KDA **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** KDA **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** KDA	SP 800-56Cr2
PBKDF2 with SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256,	#A4317, #A4326, #A4330, #A4344, #A4345, #A4346		**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC™ 7001 Series AMD EPYC 7J13:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3 **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** SHA_ASM, SHA_CE **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3	Option 1a SP 800-132
PBKDF2 with SHA3-224, SHA3-256, SHA3-384, SHA3-512	#A4312, #A4319		**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC™ 7001 Series AMD EPYC 7J13:** SHA3_ASM	

[1] This algorithm is referred to as "Single Step KDF" or "SSKDF" by OpenSSL.

Algorithm Name	CAVP Numbers	Algorithms Capabilities	OE (Implementation)	Reference
			Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30: SHA3_CE **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SHA3_ASM	
RSA PKCS#1 v1.5 and PSS with SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	#A4317, #A4326, #A4330, #A4344, #A4345, #A4346	Signature Generation using 2048, 3072, 4096-bit keys	**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC™ 7001 Series AMD EPYC 7J13:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_SSSE3 **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** SHA_ASM, SHA_CE **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_SSSE3	FIPS 186-4
		Signature Verification using 1024, 2048, 3072, 4096-bit keys	**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC™ 7001 Series AMD EPYC 7J13:** SHA_ASM, SHA_SHANI **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** SHA_ASM **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SHA_ASM, SHA_SHANI	
RSA	#A4317, #A4326, #A4330, #A4344, #A4345, #A4346	Key Pair Generation using 2048-15360-bit keys	**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC™ 7001 Series AMD EPYC 7J13:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_SSSE3 **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** SHA_ASM, SHA_CE **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_SSSE3	FIPS 186-4 Appendix B.3.6 Probable Primes with Conditions Based on Auxiliary Probable Primes
Safe Primes	#A4325	Key Generation using MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192	**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC™ 7001 Series AMD EPYC 7J13:** FFC_DH **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** FFC_DH **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** FFC_DH	SP 800-56Ar3 Section 5.6.1.1.4 Testing Candidates
Safe Primes		Key Verification using MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe2048, ffdhe3072, ffdhe4096,		SP 800-56Ar3 Sections 5.6.2.1.2 and 5.6.2.1.4

Algorithm Name	CAVP Numbers	Algorithms Capabilities	OE (Implementation)	Reference
		ffdhe6144, ffdhe8192		
SHA-1, SHA2-224, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	#A4317, #A4326, #A4330, #A4344, #A4345, #A4346	Hashing	**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC^{TM} 7001 Series AMD EPYC 7J13:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_SSSE3 **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** SHA_ASM, SHA_CE **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_SSSE3	FIPS 180-4
SHA2-256	#A4317, #A4318, #A4326, #A4330, #A4344, #A4345, #A4346		**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC^{TM} 7001 Series AMD EPYC 7J13:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_SSSE3 **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** SHA_ASM, SHA_CE, NEON **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_SSSE3	
SHA3-224, SHA3-256, SHA3-384, SHA3-512	#A4312, #A4319		**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC^{TM} 7001 Series AMD EPYC 7J13:** SHA3_ASM **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** SHA3_CE **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SHA3_ASM	FIPS 202
SHAKE128, SHAKE256		XOFs		
SSH KDF (CVL) with AES-128, AES-192, AES-256 and SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	#A4320, #A4331, #A4332, #A4333, #A4334	Key Derivation	**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC^{TM} 7001 Series AMD EPYC 7J13:** SSH_ASM, SSH_SHANI, SSH_AVX2, SSH_AVX, SSH_SSSE3 **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** SSH_ASM **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SSH_ASM, SSH_SHANI, SSH_AVX2, SSH_AVX, SSH_SSSE3	SP 800-135r1
TLS 1.2 KDF (RFC 7627) (CVL) with SHA2-256, SHA2-384, SHA2-512 using RFC 7627 Extended Master Secret	#A4317, #A4326, #A4330, #A4344, #A4345, #A4346		**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC^{TM} 7001 Series AMD EPYC 7J13:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_SSSE3 **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** SHA_ASM, SHA_CE **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_SSSE3	

Algorithm Name	CAVP Numbers	Algorithms Capabilities	OE (Implementation)	Reference
TLS 1.3 KDF (CVL) with SHA2-256, SHA2-384	#A4310		**Oracle Linux 9 on KVM on Oracle Linux 8 on AMD EPYC^{TM} 7001 Series AMD EPYC 7J13:** TLS v1.3 **Oracle Linux 9 on KVM on Oracle Linux 8 on Ampere® Altra® Q80-30:** TLS v1.3 **Oracle Linux 9 on KVM on Oracle Linux 8 on Intel® Xeon® Platinum 8358:** TLS v1.3	RFC 8446

Table 6 - Approved Algorithms

Vendor Affirmed Algorithms:

Algorithm Name	Algorithm Capabilities	OE (Implementation)	References
Cryptographic Key Generation (CKG)	FIPS 186-4 Key generation RSA KeyGen: 2048, 3072, 4096 bits with 112, 128, 149 bits of key strength. ECDSA KeyGen: P-224, P-256, P 384, P-521 elliptic curves with 112-256 bits of key strength Safe Primes Key Generation: MODP-2048, MODP-3072, MODP-4096, MODP-6144, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 with 112-200 bits of key strength	Same as in Table 6	SP 800-133Rev2 Section 4, 5.1, 5.2 FIPS 140-3 IG D.H

Table 7 - Vendor Affirmed Algorithms

Non-Approved, Allowed Algorithms:

The module does not implement non-approved algorithms allowed in the approved mode of operation.

Non-Approved, Allowed Algorithms with No Security Claimed:

The module does not implement non-approved algorithms allowed in the approved mode of operation with no security claimed.

Non-Approved, Not Allowed Algorithms:

Name	Use and Function
AES GCM with external IV	Encryption
ANS X9.42 KDF (SHAKE128, SHAKE256)	Key Derivation
ANS X9.63 KDF (SHA-1, SHAKE128, SHAKE256)	
Hash_DRBG (SHA-224, SHA-384)	Random Number Generation
HMAC_DRBG (SHA-224, SHA-384)	
ECDSA with curve P-192	Key Pair Generation, Key Pair Verification
HMAC (< 112-bit keys)	Message Authentication Code Generation, Message Authentication Code Verification
KAS1, KAS2	Shared Secret Computation
KBKDF, KDA OneStep, HKDF, ANS X9.42 KDF, ANS X9.63 KDF (< 112-bit keys)	Key Derivation
KDA OneStep, HKDF (SHAKE128, SHAKE256)	
PBKDF2 (short password; short salt; insufficient iterations; < 112-bit keys)	Password-Based Key Derivation
RSA and ECDSA (pre-hashed message), ECDSA with curve P-192	Signature Generation, Signature Verification
RSA-PSS (invalid salt length)	
RSA-OAEP	Asymmetric Encryption, Asymmetric Decryption
SSH KDF (SHA-512/224, SHA-512/256, SHA-3, SHAKE128, SHAKE256)	Key Derivation
TLS 1.2 KDF (SHA-1, SHA-224, SHA-512/224, SHA-512/256, SHA-3)	

Name	Use and Function
TLS 1.2 KDF using master secret non-compliant with RFC 7627	
TLS 1.3 KDF (SHA-1, SHA-224, SHA-512, SHA-512/224, SHA-512/256, SHA-3)	

Table 8 - Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	SF Capabilities	Algorithms
KAS-ECC-SSC	KAS	SP 800-56Arev3. KAS-ECC-SSC per IG D.F scenario 2(1)	Ephemeral Unified scheme Curves: P-224, P-256, P-384, P-521 elliptic curves with 112-256 bits of key strength	KAS-ECC-SSC: #A4317, #A4326, #A4330, #A4344, #A4345, #A4346
KAS-FFC-SSC		SP 800-56Arev3. KAS-FFC-SSC per IG D.F scenario 2(1)	Ephemeral Unified scheme Keys: 2048, 3072, 4096, 6144, 8192-bit keys with 112-200 bits of key strength	KAS-FFC-SSC: #A4325
AES-CCM	KTS	SP 800-38C and SP 800-38F. KTS (Key wrapping and unwrapping) per IG D.G	128, 192, 256 bits with 128-256 bits of key strength	AES: #A4313, #A4314, #A4315, #A4327, #A4328, #A4329
AES-GCM		SP 800-38D and SP 800-38F. KTS (Key wrapping and unwrapping) per IG D.G, Additional Comment 8	128, 192, 256 bits with 128-256 bits of key strength	AES: #A4316, #A4321, #A4322, #A4323, #A4335, #A4336, #A4337, #A4338, #A4339, #A4340, #A4341, #A4342, #A4343
AES-KW		SP 800-38F. KTS (Key wrapping and unwrapping) per IG D.G	128, 192, 256 bits with 128-256 bits of key strength	AES: #A4313, #A4314, #A4315, #A4327, #A4328, #A4329
AES-KWP				

Table 9 - Security Function Implementation

2.7 Algorithm Specific Information

2.7.1 AES GCM IV

The Crypto Officer shall consider the following requirements and restrictions when using the module.

For TLS 1.2, the module offers the AES GCM implementation and uses the context of Scenario 1 of FIPS 140-3 IG C.H. The module is compliant with SP 800-52r2 Section 3.3.1 and the mechanism for IV generation is compliant with RFC 5288 and 8446.

The module does not implement the TLS protocol. The module's implementation of AES GCM is used together with an application that runs outside the module's cryptographic boundary. The design of the TLS protocol implicitly ensures that the counter (the nonce_explicit part of the IV) does not exhaust the maximum number of possible values for a given session key.

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.

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.

Finally, 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. The module's implementation of AES GCM is used together with an application that runs outside the module's cryptographic boundary. The design of the TLS protocol implicitly ensures that the counter (the nonce_explicit part of the IV) does not exhaust the maximum number of possible values for a given session key.

2.7.2 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 with 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, and shall consist of lowercase, uppercase, and numeric characters. The probability of guessing the value 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 (this is verified by the module to determine the service is approved), 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 module only allows minimum iteration count to be 1000.

2.7.3 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. 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.

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.

2.7.4 SP 800-56Ar3 Assurances

The module offers DH and ECDH shared secret computation services compliant to the SP 800-56Ar3 and meeting IG D.F scenario 2 path (1). To meet the required assurances listed in section 5.6 of SP 800-56Ar3, the module shall be used together with an application that implements the "TLS protocol" and the following steps shall be performed.

- The entity using the module, must use module's "Key pair generation" service for generating DH/ECDH ephemeral keys. This meets the assurances required by key pair owner defined in the section 5.6.2.1 of SP 800-56Ar3.
- As part of the module's shared secret computation (SSC) service, the module internally performs the public key validation 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 800-56Ar3.
- The module does not support static keys. Therefore, the "assurance of peer's possession of private key" is not applicable.

2.7.5 Legacy Algorithms

The module provides the following legacy uses as defined in SP 800-131rev2:

- RSA Signature Verification, under FIPS 186-4, allows verifying signatures with a bit size of 1024, along with the approved modulus sizes of 2048, 3072, and 4096 bits.

2.8 RNG and Entropy

Entropy Information:

Name	Type	Operational Environment	Sample Size	Entropy Per Sample	Conditioning Component
Oracle OpenSSL CPU Time Jitter RNG Entropy Source (Cert. #E90)	Non-physical	See Table 2	64 bits	Full entropy	Linear-Feedback Shift Register (LFSR); HMAC-SHA-512 DRBG (AVP cert A3862, A4162); AES-256 CTR DRBG (CAVP cert A4311)

Table 10 - Entropy

RNG Information:

The module employs two Deterministic Random Bit Generator (DRBG) implementations based on SP 800-90Ar1. These DRBGs are used internally by the module (e.g. to generate seeds for asymmetric key pairs and random numbers for security functions). They can also be accessed using the specified API functions. The following parameters are used:

1. Private DRBG: AES-256 CTR_DRBG with derivation function. This DRBG is used to generate secret random values (e.g. during asymmetric key pair generation). It can be accessed using RAND_priv_bytes.
2. Public DRBG: AES-256 CTR_DRBG with derivation function. This DRBG is used to generate general purpose random values that do not need to remain secret (e.g. initialization vectors). It can be accessed using RAND_bytes.

For seeding, the DRBG that is seeded with 384 bits of seed material, corresponding to 384 bits of entropy, obtained from the SP800-90B compliant entropy source above in Table 10. During reseeding, the DRBG obtains 256 bits of seed material, corresponding to 256 bits of entropy. These DRBGs will always employ prediction resistance. More information regarding the configuration and design of these DRBGs can be found in the module's manual pages.

2.9 Key Generation

Name	Type	Properties
Safe primes key pair generation	CKG	Key type: DH key pair Groups: MODP-2048, MODP-3072, MODP-4096, MODP-6144, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 Security strength: 112-200 bits Method: SP 800-56Ar3 (safe primes) Section 5.6.1.1.4 Testing Candidates Compliant to FIPS 140-3 IG D.H, SP 800-133r2, Section 4, 5.2
ECDSA key pair generation		Key type: EC key pair Curves: P-224, P-256, P-384, P-521 Security strength: 112-256 bits Method: FIPS 186-4 Appendix B.4.2 Testing Candidates Compliant to FIPS 140-3 IG D.H, SP 800-133r2, Section 4, 5.1
RSA key pair generation		Key type: RSA key pair Modulus: 2048-15360 bits Security strength: 112-256 bits Method: FIPS 186-4 Appendix B.3.6 Probable Primes with Conditions Based on Auxiliary Probable Primes Compliant to FIPS 140-3 IG D.H, SP 800-133r2, Section 4, 5.1
KBKDF key derivation	Key Derivation	Key type: Symmetric key Security strength: 112-256 bits

		Method: Counter and feedback mode, using CMAC and HMAC SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512 Compliant to SP 800-108r1
KDA OneStep		Key type: Symmetric key Security strength: 112-256 bits Method: (HMAC) SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512 Compliant to SP 800-56Cr2
HKDF		Key type: Symmetric key Security strength: 112-256 bits Method: (HMAC) SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512 Compliant to SP 800-56Cr1
ANS X9.42 KDF (CVL)		Key type: Symmetric key Security strength: 112-256 bits Method: AES KW with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512 Compliant to SP 800-135r1
ANS X9.63 KDF (CVL)		Key type: Symmetric key Security strength: 112-256 bits Method: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512 Compliant to SP 800-135r1
SSH KDF (CVL)		Key type: Symmetric key Security strength: 112-256 bits Method: AES-128, AES-192, AES-256 with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 Compliant to SP 800-135r1
TLS 1.2 KDF (RFC 7627) (CVL)		Key type: Symmetric key Security strength: 112-256 bits Method: SHA-256, SHA-384, SHA-512 Compliant to SP 800-135r1
TLS 1.3 KDF (CVL)		Key type: Symmetric key Security strength: 112-256 bits Method: SHA-256, SHA-384 Compliant to SP 800-135r1
PBKDF2		Key type: Symmetric key Security strength: 112-256 bits Method: Option 1a with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512 Compliant to option 1a of SP 800-132

Table 11 - Key Generation

2.10 Key Establishment

Name	Type	Properties
KAS-FFC-SSC [SP800-56Arev3]	KAS (Shared Secret Computation)	Groups: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 Security strength: 112-200 bits Compliant with: Scenario 2 (1) of FIPS 140-3 IG D.F: Shared secret computation
KAS-ECC-SSC [SP800-56Arev3]		Curves: P-224, P-256, P-384, P-521 Security strength: 112-256 bits Compliant with: Scenario 2 (1) of FIPS 140-3 IG D.F: Shared secret computation
AES CCM [SP 800-38C]	KTS-Wrap (Key Wrapping, Key Unwrapping)	Keys: 128, 192, or 256 bits Security strength: 128, 192, or 256 bits Compliant with IG D.G
AES GCM [SP 800-38D]	KTS-Wrap (Key Wrapping)	Keys: 128, 192, or 256 bits Security strength: 128, 192, or 256 bits IV generated internally Compliant with IG D.G Additional comment 8
	KTS-Wrap (Key Unwrapping)	Keys: 128, 192, or 256 bits Security strength: 128, 192, or 256 bits IV provided externally Compliant with IG D.G Additional comment 8
AES KW [SP 800-38F]	KTS-Wrap (Key Wrapping, Key Unwrapping)	Keys: 128, 192, or 256 bits Security strength: 128, 192, or 256 bits Compliant with IG D.G
AES KWP [SP 800-38F]		

Table 12 - Key Establishment

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 Table 12. Note that the module only implements key pair generation, key pair verification, and shared secret computation.

SSH 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 1.2, 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.

No other part of the IKE, SSH or TLS protocols, other than the approved cryptographic algorithms and the KDFs listed above, have been tested by the CAVP and CMVP.

3 Cryptographic Module Interfaces

3.1 Description

Physical Port	Logical Interface	Data That Passes Over the Port/Interface
As a software-only module, the module does not have physical ports. Physical Ports are interpreted to be the physical ports of the hardware platform on which it runs.	Data Input	API data input parameters
	Data Output	API output parameters
	Control Input	API function calls, API control input parameters
	Status Output	API return code, error queue

Table 13 - 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.

3.2 Trusted Channel Specification

The module does not implement a trusted channel.

3.3 Control Interface Not Inhibited

The module does not implement a control output interface.

4 Roles, Services, and Authentication

4.1 Authentication Methods

The module does not implement authentication.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	N/A (Implicitly assumed)

Table 14 - 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.

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	Roles	SSP Access
Message Digest	Compute a message digest	EVP_DigestFinal_ex returns 1	Message	Digest value	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512	CO	N/A
XOF	Compute the output of an XOF	EVP_DigestFinalXOF returns 1	Message	Digest value	SHAKE128, SHAKE256		N/A
Encryption	Encrypt a plaintext	EVP_EncryptFinal_ex returns 1	AES Key, plaintext	Ciphertext	AES ECB, CBC, CBC-CTS-CS1, CBC-CTS-CS2, CBC-CTS-CS3, CFB1, CFB8, CFB128, CTR, OFB, XTS		AES Key: W, E
Decryption	Decrypt a ciphertext	EVP_DecryptFinal_ex returns 1	AES Key, ciphertext	Plaintext			
Authenticated Encryption	Encrypt a plaintext	AES GCM: EVP_CIPHER_*_FIPS_INDICATOR_APPROVED Others: EVP_EncryptFinal_ex returns 1	AES Key, plaintext, IV (for CCM and GCM only)	Ciphertext, MAC tag	AES CCM, GCM (internal IV)		AES Key: W, E
Authenticated Decryption	Decrypt a ciphertext	AES GCM: EVP_CIPHER_*_FIPS_INDICATOR_APPROVED Others: EVP_DecryptFinal_ex returns 1	AES Key, ciphertext, MAC tag	Plaintext or failure	AES CCM, GCM (external IV)		
Message Authentication Code	Compute a MAC tag	HMAC: EVP_MAC_*_FIPS_INDICATOR_APPROVED	AES Key, message	MAC tag	AES CMAC, AES GMAC		AES Key: W, E

Name	Description	Indicator	Inputs	Outputs	Security Functions	Roles	SSP Access
Generation		Others: EVP_MAC_final returns 1	HMAC Key, message		HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512, HMAC-SHA2-512/224, HMAC-SHA2-512/256, HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512		HMAC Key: W, E
Message Authentication Code Verification	Verify a MAC tag		AES Key, message, MAC tag	Pass/fail	AES CMAC, AES GMAC		AES Key: W, E
			HMAC Key, message, MAC tag		HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512, HMAC-SHA2-512/224, HMAC-SHA2-512/256, HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512		HMAC Key: W, E

Shared Secret Computation	Compute a Shared Secret	EVP_PKEY_derive returns 1	DH Private Key, DH Public Key	Shared Secret	KAS-FFC-SSC		DH Private Key: W, E;
							DH Public Key: W, E;
							Shared Secret: G, R

			EC Private Key, EC Public Key		KAS-ECC-SSC		EC Private Key: W, E;
							EC Public Key: W, E;
							Shared Secret: G, R

Key Derivation	Derive a key	EVP_KDF_*_FIPS_INDICATOR_APPROVED	TLS Pre-Master Secret	TLS Master Secret	TLS 1.2 KDF (RFC 7627) (CVL), TLS 1.3 KDF (CVL)		TLS Pre-Master Secret: G; TLS Master Secret: G
			TLS Master Secret	TLS Derived Key	TLS 1.2 KDF (RFC 7627) (CVL), TLS 1.3 KDF (CVL)		TLS Master Secret: E; TLS Derived Key: G, R

			Shared Secret	Derived Key	KDA OneStep, KDA HKDF, KDF, ANS X9.42 KDF (CVL), ANS X9.63 KDF (CVL), SSH KDF (CVL)		Shared Secret: W, E;
							KDA HKDF Derived key: G, R;
							KDA OneStep Derived key: G, R;
							SSH KDF Derived Key: G, R
							ANS X9.42 KDF Derived key: G, R;
							ANS X9.63 KDF Derived key: G, R;

Key-Based Key Derivation	Derive a key from a key		Key-Derivation Key		KBKDF		Key-Derivation Key: W, E; KBKDF Derived Key: G, R
Password-Based Key Derivation	Derive a key from a password		Password		PBKDF2		Password: W, E; PBKDF2 Derived Key: G, R
Key Pair	Generate a	EVP_PKEY_generate	DH Group	Module	Safe Primes Key		Module Generated DH Private

Name	Description	Indicator	Inputs	Outputs	Security Functions	Roles	SSP Access

Generation	key pair	returns 1		Generated DH Private Key, Module Generated DH Public Key	Generation CKG		Key: G, R;
							Module Generated DH Public Key: G, R;
							Intermediate Key Generation Value: G, E, Z

			Curve	Module Generated EC Private Key, Module Generated EC Public Key	ECDSA KeyGen CKG		Module Generated EC Private Key: G, R;
							Module Generated EC Public Key: G, R;
							Intermediate Key Generation Value: G, E, Z

			Modulus	Module Generated RSA Private Key, Module Generated RSA Public Key	RSA KeyGen CKG		Module Generated RSA Private Key: G, R; Module Generated RSA Public Key: G, R Intermediate Key Generation Value: G, E, Z

Key Pair Verification	Verify a key pair. Safe primes key pair verification, ECDSA key pair verification	EVP_PKEY_public_check or EVP_PKEY_private_check or EVP_PKEY_check returns 1	DH Private Key, DH Public Key EC Private Key, EC Public Key	Pass/fail	Safe Prime KeyVer, ECDSA KeyVer		DH Public Key: W, E;
							DH Private Key: W, E;
							EC Private Key: W, E;
							EC Public Key: W, E

Key Wrapping	Wrap a key	EVP_EncryptFinal_ex returns 1	AES Key, key to be wrapped	Wrapped key	AES CCM, GCM (internal IV), AES KW, AES KWP		AES Key: W, E
Key Unwrapping	Unwrap a key	EVP_DecryptFinal_ex returns 1	AES Key, key to be unwrapped	Unwrapped key	AES CCM, GCM (external IV), AES KW, AES KWP		AES Key: W, E

Random Number Generation	Generate random bytes	EVP_RAND_generate returns 1	Output length	Random bytes	CTR_DRBG		Entropy Input: W, E;
							DRBG Seed: E, G;
							Internal State (V, Key): E, G

					HMAC_DRBG		

					Hash_DRBG		Entropy Input: W, E;
							DRBG Seed: E, G;
							Internal State (V, C): E, G

Signature Verification	Verify a digital signature	RSA: OSSL_*_FIPSINDICATOR_APPROVED and EVP_SIGNATURE_*_FIPS_INDICATOR_APPROVED ECDSA: OSSL_*_FIPSINDICATOR_APPROVED	Message, EC public key or RSA Public Key, signature, hash algorithm	Pass/fail	RSA PKCS#1 v1.5 and PSS with SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256 ECDSA (P-224, P-256, P-384, P-521) with SHA-		RSA Public Key: W, E; ECDSA Public Key: W, E
Signature Generation	Generate a digital signature		Message, EC Private Key or RSA Private	Signature			RSA Private Key: W, E; ECDSA Private Key: W, E

Name	Description	Indicator	Inputs	Outputs	Security Functions	Roles	SSP Access
			Key, hash algorithm		224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512		
Show Version	Return the module name and version information	None	N/A	Module name and version	N/A		N/A
Show Status	Return the module status	None	N/A	Module status	N/A		N/A
Self-Test	Perform the CASTs and integrity tests	None	N/A	Pass/fail	SHA-1, SHA-512, SHA3-256, AES ECB, AES GCM, KBKDF, KDA OneStep, HKDF, ANS X9.42 KDF (CVL), ANS X9.63 KDF (CVL), SSH KDF (CVL), TLS 1.2 KDF (RFC 7627) (CVL), TLS 1.3 KDF (CVL), PBKDF2, CTR_DRBG, Hash_DRBG, HMAC_DRBG, KAS-FFC-SSC, KAS-ECC-SSC, RSA PKCS#1 v1.5, ECDSA See Table 23 for specifics		N/A
Zeroization	Zeroize all SSPs	None	Any SSP	N/A	N/A		All SSPs: Z

Table 15 - Approved Services

Table 15 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 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. Additionally, this EVP API layer can be used to retrieve the approved service indicator for the module. The redhat_ossl_query_fipsindicator() function indicates 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.

- OSSL_CIPHER_PARAM_*_FIPS_INDICATOR
- OSSL_MAC_PARAM_*_FIPS_INDICATOR
- OSSL_KDF_PARAM_*_FIPS_INDICATOR
- OSSL_SIGNATURE_PARAM_*_FIPS_INDICATOR
- OSSL_ASYM_CIPHER_PARAM_*_FIPS_INDICATOR

4.4 Non-Approved Services

Name	Description	Security Functions	Role
Encryption	Encrypt a plaintext	AES GCM with external IV	CO
Message Authentication Code Generation	Compute a MAC tag	HMAC with < 112-bit keys	
Message Authentication Code Verification	Verify a MAC tag		
Key Derivation	Derive a key	KDA OneStep with < 112-bit keys	
		HKDF with < 112-bit keys	
		ANS X9.42 KDF with < 112-bit keys	
		ANS X9.63 KDF with < 112-bit keys	
		SSH KDF with < 112-bit keys	
		TLS 1.2 KDF < 112-bit keys	
		TLS 1.3 KDF < 112-bit keys	
		KDA OneStep with SHAKE128, SHAKE256	
		ANS X9.42 KDF with SHAKE128, SHAKE256	
		ANS X9.63 KDF with SHA-1, SHAKE128, SHAKE256	
		SSH KDF with SHA-512/224, SHA-512/256, SHA-3, SHAKE128, SHAKE256	
		TLS 1.2 KDF using master secret non-compliant with RFC 7627	
		TLS 1.2 KDF with SHA-1, SHA-224, SHA-512/224, SHA-512/256, SHA-3	
		TLS 1.3 KDF with SHA-1, SHA-224, SHA-512, SHA-512/224, SHA-512/256, SHA-3	
Key-Based Key Derivation	Derive a key from a derivation key	KBKDF with < 112-bit keys	
Password-Based Key Derivation	Derive a key from a password	PBKDF2 with a short password; short salt; insufficient iterations; < 112-bit keys	
Key Pair Generation	Generate a key pair	ECDSA with curve P-192	
Key Pair Verification	Verify a key		
Shared Secret Computation	Compute a shared secret	KAS1, KAS2	
Signature Generation	Generate a signature	ECDSA signature generation with a pre-hashed message, ECDSA with curve P-192	
		RSA signature generation with a pre-hashed message	
Signature Verification	Verify a signature	ECDSA signature verification with a pre-hashed message, ECDSA with curve P-192	
		RSA signature verification with a pre-hashed message	
Asymmetric Encryption	Encrypt a plaintext	RSA-OAEP encryption	
Asymmetric Decryption	Decrypt a plaintext	RSA-OAEP decryption	

Table 16 - Non-Approved Services

4.5 External Software/Firmware Loaded

The module does not load external software or firmware.

4.6 Bypass Actions and Status

The module does not implement a bypass capability.

4.7 Cryptographic Output Actions and Status

The module does not implement a self-initiated cryptographic output capability.

5 Software/Firmware Security

5.1 Integrity Techniques

The integrity of the module is verified by comparing a HMAC SHA-256 value calculated at run time with the HMAC SHA-256 value embedded in the fips.so file that was computed at build time. If the integrity test fails, the module enters the error state.

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 test may be invoked on-demand by unloading and subsequently re-initializing the module, or by calling the OSSL_PROVIDER_self_test function. This will perform (among others) the software integrity test.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operating Environment: modifiable: the module executes on a general purpose operating system (Oracle Linux 9), which allows modification, loading, and execution of software that is not part of the validated module.

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 Configurable Settings and Restrictions

The module shall be installed as stated in Section 11.1.

There are no concurrent operators.

The module does not have the capability of loading software or firmware from an external source.

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

7 Physical Security

The module is comprised of software only and therefore this section is not applicable.

8 Non-Invasive Security

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 17 - Storage Areas

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	Related SFI
API input parameters	Operator calling application (TOEPP)	Cryptographic module	Plaintext (P)	Manual (MD)	Electronic (EE)	N/A
API output parameters	Cryptographic module	Operator calling application (TOEPP)				

Table 18 - SSP Input-Output

The module does not support entry and output of SSPs beyond the physical perimeter of the operational environment. The SSPs are provided to the module via API input parameters in the plaintext form and output via API output parameters in the plaintext form to and from the calling application running on the same operational environment.

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
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.	By calling the appropriate zeroization functions: **AES key:** EVP_CIPHER_CTX_free, EVP_MAC_CTX_free **HMAC key:** EVP_MAC_CTX_free **Key-derivation key**: EVP_KDF_CTX_free **Shared secret:** EVP_KDF_CTX_free **PBKDF Password:** EVP_KDF_CTX_free **Derived key:** EVP_KDF_CTX_free **Entropy input:** EVP_RAND_CTX_free **DRBG seed:** EVP_RAND_CTX_free **DRBG Internal state:** EVP_RAND_CTX_free **DH private key:** EVP_PKEY_free **DH public key:** EVP_PKEY_free **EC private key:** EVP_PKEY_free **EC public key:** EVP_PKEY_free **RSA private key:** EVP_PKEY_free **RSA public key:** EVP_PKEY_free **TLS pre-master secret:** EVP_KDF_CTX_free **TLS master secret:** EVP_KDF_CTX_free **TLS derived secret:** EVP_KDF_CTX_free
Automatic			**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	By removing power

Table 19 - SSP Zeroization Methods

9.4 SSPs

Name	Description	Size	Strength	Type	Generated By	Established By
AES Key	AES key used for encryption, decryption, authenticated encryption, authenticated decryption, key wrapping, key unwrapping, and computing MAC tags	128, 192, 256 bits	128-256 bits	Symmetric key	N/A	N/A
HMAC Key	HMAC key	112-524288 bits	112-256 bits	Authentication key	N/A	N/A
Shared Secret	Shared secret established by DH/ECDH	ECDH: 128-256 bits	128-256 bits	Shared secret	N/A	KAS-ECC-SSC
		DH: 112-200 bits	112-200 bits			KAS-FFC-SSC
Key-Derivation Key	Key-derivation key for KBKDF	112-256 bits	112-256 bits	Key-derivation key	N/A	N/A
Password	PBKDF2 password	At least 14 characters	N/A	Password	N/A	N/A
KBKDF Derived Key	KBKDF derived key	112-4096 bits	112-256 bits	Derived key	KBKDF	N/A
PBKDF2 Derived key	PBKDF2 derived key	112-4096 bits			PBKDF2	
KDA OneStep Derived key	KDA OneStep derived key	112-2048 bits			OneStep KDA	
KDA HKDF Derived key	KDA HKDF derived key	2048 bits			KDA HKDF	
ANS X9.42 KDF Derived key	ANS X9.42 KDF derived key	112-4096 bits			ANS X9.42 KDF (CVL)	
ANS X9.63 KDF Derived key	ANS X9.63 KDF derived key	128-4096 bits			ANS X9.63 KDF (CVL)	
SSH KDF Derived key	SSH KDF derived key	112-1024 bits			SSH KDF (CVL)	
Entropy Input	Entropy input used to seed the DRBGs	128-448 bits	128-256 bits	Entropy	Entropy Source (ESV cert. E90) See Table 10	N/A
DRBG Seed	DRBG seed derived from entropy input as defined in SP 800-	CTR_DRBG: 128, 192, 256 bits	128-256 bits	Seed	CTR_DRBG, Hash_DRBG, HMAC_DRBG	N/A
		Hash_DRBG: 128, 256 bits				

Name	Description	Size	Strength	Type	Generated By	Established By
	90Ar1	HMAC_DRBG: 128, 256 bits				
DRBG Internal State (V, Key)	Internal state of CTR_DRBG and HMAC_DRBG	CTR_DRBG: 128, 192, 256 bits HMAC_DRBG: 128, 256 bits	128-256 bits	DRBG Internal state	CTR_DRBG, HMAC_DRBG (derived from DRBG seed as defined in SP800-90Ar1)	N/A
DRBG Internal State (V, C)	Internal state of Hash_DRBG	Hash_DRBG: 128, 256 bits	128-256 bits		Hash_DRBG (derived from DRBG seed as defined in SP800-90Ar1)	N/A
DH Public Key	Public key used for Shared Secret Computation	ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	112-200 bits	Public key	N/A	KAS-FFC-SSC
DH Private Key	Private key used for Shared Secret Computation			Private key		
Module Generated DH Public Key	DH public key generated by the module			Public key	Safe primes (SP 800-56Ar3 section 5.6.1.1.4 Testing Candidates) CTR_DRBG (for generation of random values per SP 800-90Ar1)	N/A
Module Generated DH Private Key	DH private key generated by the module			Private key		
EC Private Key	Private key used for ECDSA signature generation and Shared Secret Computation	P-224, P-256, P-384, P-521	112, 128, 192, 256 bits	Private key	N/A	KAS-ECC-SSC
EC Public Key	Public key used for ECDSA signature verification and Shared Secret Computation			Public key		
Module Generated EC Private Key	EC private key generated by the module			Private key	ECDSA (FIPS 186-4 Appendix B.4.2 Testing Candidates) CTR_DRBG (for generation of random values per SP 800-90Ar1)	N/A
Module Generated EC Public Key	EC public key generated by the module			Public key		
RSA Private Key	Private key used for RSA signature generation	2048, 3072, 4096 bits	112, 128, 150 bits	Private key	N/A	N/A
RSA Public Key	Public key used for RSA signature verification	1024, 2048, 3072, 4096 bits	80, 112, 128, 150 bits	Public key	N/A	N/A
Module Generated RSA	RSA private key generated by the	2048, 3072, 4096	112, 128, 150 bits	Private key	RSA (FIPS 186-4 Appendix B.3.6	N/A

Name	Description	Size	Strength	Type	Generated By	Established By
Private Key	module	bits			Probable Primes with Conditions Based on Auxiliary Probable Primes) CTR_DRBG (random values generation per 800-90Ar1)	
Module Generated RSA Public Key	RSA public key generated by the module			Public key		N/A
Intermediate Key Generation Value	Intermediate key generation value	224-4096 bits	112-256 bits	Intermediate key generation value	CKG	N/A
TLS Pre-Master Secret	TLS pre-master secret used for deriving the TLS master secret	112-256 bits	112-256 bits	TLS pre-master secret	N/A	KAS-FFC-SSC, KAS-ECC-SSC
TLS Master Secret	TLS master secret used for deriving the TLS derived secret	112-256 bits	112-256 bits	TLS master secret	TLS 1.2 KDF (RFC 7627) (CVL), TLS 1.3 KDF (CVL)	N/A
TLS Derived Key	TLS derived key, derived from TLS master secret	112-256 bits	112-256 bits	Shared secret		N/A

Table 20 - SSP Information First

Name	Used By	Inputs/Outputs	Storage	Storage Duration	Zeroization	Type	Related SSPs
AES Key	Encryption, Decryption, Authenticated Encryption, Authenticated Decryption, Key Wrapping, Key Unwrapping Message Authentication Code Generation, Message Authentication Code Verification	API input parameters (input)	RAM	For the duration of the service	Free Cipher Handle, Module Reset	CSP	None
HMAC Key	Message Authentication Code Generation, Message Authentication Code Verification						
Shared Secret	Key Derivation	API output parameters (output)					DH Public Key, DH Private Key, EC Public Key, EC Private Key, KDA OneStep Derived key, KDA HKDF

Name	Used By	Inputs/Outputs	Storage	Storage Duration	Zeroization	Type	Related SSPs
							Derived key, ANS X9.63 KDF Derived key, SSH KDF Derived key
Key-Derivation Key	Key-Based Key Derivation	API input parameters (input)		For the duration of the service			KBKDF Derived Key
Password	Password-Based Key Derivation	API input parameters (input)					PBKDF2 Derived Key
KBKDF Derived Key	Key-based Key Derivation	API output parameters (output)					Key-derivation Key
PBKDF2 Derived key	Password-based Key Derivation						Password
KDA OneStep Derived key	Key Derivation						Shared secret
KDA HKDF Derived key							Shared secret
ANS X9.42 KDF Derived key							Shared secret
ANS X9.63 KDF Derived key							Shared secret
SSH KDF Derived key							Shared secret
Entropy Input	Random Number Generation	N/A		From generation until DRBG seed is created		CSP (Compliant with IG D.L)	DRBG Seed
DRBG Seed		N/A		While the DRBG is being instantiated			Entropy Input DRBG Internal State (V, C) DRBG Internal State (V, Key)
DRBG Internal State (V, Key)		N/A		From DRBG instantiation until DRBG termination			DRBG Seed
DRBG Internal State (V, C)		N/A					DRBG Seed
DH Public Key	Shared Secret Computation, Key Pair Verification	API input parameters (input)		For the duration of the service		PSP	DH Private Key Shared Secret
DH Private Key						CSP	DH Public Key Shared Secret
Module Generated DH Public Key	Shared Secret Computation	API output parameters (output)				PSP	Module Generated DH Private Key Intermediate key generation value
Module Generated DH Private Key						CSP	Module Generated DH Public Key Intermediate key generation value
EC Public Key	Shared Secret Computation, Signature Verification,	API input parameters (input)				PSP	EC Private Key Shared Secret

Name	Used By	Inputs/Outputs	Storage	Storage Duration	Zeroization	Type	Related SSPs
	Key Pair Verification						
EC Private Key	Shared Secret Computation, Signature Generation, Key Pair Verification					CSP	EC Public Key Shared Secret
Module Generated EC Public Key	Shared Secret Computation	API output parameters (output)				PSP	Module Generated EC Private Key Intermediate key generation value
Module Generated EC Private Key						CSP	Module Generated EC Public Key Intermediate key generation value
RSA Private Key	Digital Signature Generation	API input parameters (input)				CSP	RSA Public Key
RSA Public Key	Digital Signature Verification					PSP	RSA Private Key
Module Generated RSA Private Key	N/A	API output parameters (output)				CSP	Module Generated RSA Public Key Intermediate key generation value
Module Generated RSA Public Key						PSP	Module Generated RSA Private Key Intermediate key generation value
Intermediate Key Generation Value	Key Pair Generation	N/A			Automatically	CSP	Module Generated RSA Public Key, Module Generated RSA Private Key, Module Generated DH Public Key, Module Generated DH Private Key, Module Generated EC Public Key, Module Generated EC Private Key
TLS Pre-Master Secret	Key Derivation	N/A			Free Cipher Handle, Module	CSP	TLS Master

Name	Used By	Inputs/Outputs	Storage	Storage Duration	Zeroization	Type	Related SSPs
					Reset		Secret DH Public Key DH Private Key EC Public Key EC Private Key
TLS Master Secret		N/A				CSP	TLS Pre-Master Secret TLS Derived Secret
TLS Derived Key		API output parameters (output)				CSP	TLS Master Secret

Table 21 - SSP Information Second

9.5 Transitions

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

The RSA, ECDSA algorithm as implemented by the module conforms to FIPS 186-4, which has been superseded by FIPS 186-5. FIPS 186-4 will be withdrawn on February 3, 2024.

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm	Implementation	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256	SHA_CE, SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3, NEON	256-bit key	Message Authentication	Software integrity	Module becomes operational	Integrity test for fips.so

Table 22 - Pre-Operational Self-Tests

The pre-operational software integrity test is performed automatically (after the CASTs) 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 software 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-test has passed successfully. If the pre-operational self-test fails, the module transitions to the error state.

10.2 Conditional Self-Tests

Algorithm	Implementation	Test Properties	Test Method	Type	Indicator	Details	Conditions
ECDSA	SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3, SHA_ASM, SHA_CE	SHA2-256	PCT	Pair-Wise Consistency Test	Successful key generation	Signature generation and verification	EC key pair generation
RSA	SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3, SHA_ASM, SHA_CE	PKCS#1 v1.5 with SHA2-256					RSA key pair generation
Safe Primes	C	N/A				Public key re-computation and comparison with the existing public key (per SP 800-56Ar3 Section 5.6.2.1.4)	Safe Primes key pair generation
SHA-1, SHA2-512	SHA_CE, SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3	24-bit message	KAT	CAST	Module is operational	Message Digest	Module initialization
SHA3-256	SHA3_ASM, SHA3_CE	32-bit message					
AES-ECB	SSH_ASM, AESNI, BAES_CTASM, AESASM, SSH_SHANI, SSH_AVX2, SSH_AVX, SSH_SSSE3, CE, VPAES, AES_C	128-bit keys, 128-bit ciphertext				Decryption	

Algorithm	Implementation	Test Properties	Test Method	Type	Indicator	Details	Conditions
AES-GCM	AESNI_AVX, AESNI_CLMULNI, AESNI_ASM, BAES_CTASM_AVX, BAES_CTASM_CLMULNI, BAES_CTASM_ASM, AESASM_AVX, AESASM_CLMULNI, AESASM_ASM, CE_GCM_UNROLL8_EOR3, CE_GCM, VPAES_GCM, AES_C_GCM	256-bit keys, 96-bit IVs, 128-bit plaintext, 128-bit additional data				Encryption	
						Decryption	
KBKDF	KBKDF	Counter mode, HMAC-SHA2-256, 128-bit input key				Key Derivation	
KDA OneStep	KDA	SHA-224, 392-bit input secret					
HKDF	TLS v1.3	SHA-256, 48-bit input secret					
ANS X9.42 KDF (CVL)	SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3, SHA_CE	SHA-1 with AES-128, KW, 160-bit input secret					
ANS X9.63 KDF (CVL)	SHA3_ASM, SHA_CE, SHA3_CE, SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3, SHA_CE	SHA-256, 192-bit input secret					
SSH KDF (CVL)	SSH_ASM, SSH_SHANI, SSH_AVX2, SSH_AVX, SSH_SSSE3	SHA-1, 1056-bit input secret					
TLS 1.2 KDF (CVL)	SHA_CE, SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3, SHA_CE	SHA-256, 84-bit input secret					
TLS 1.3 KDF (CVL)	TLS v1.3	Extract and expand modes, SHA-256					
PBKDF2	SHA_CE, SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3, SHA_CE	SHA-256, 24-character password, 288-bit salt, Iteration count: 4096					
CTR_DRBG	DRBG_3	AES-128 with prediction resistance and derivation function				Instantiate, Generate, Reseed, Generate (compliant	

Algorithm	Implementation	Test Properties	Test Method	Type	Indicator	Details	Conditions
Hash_DRBG	DRBG_3	SHA-256 with prediction resistance				with SP 800-90Ar1 Section 11.3)	
HMAC_DRBG	DRBG_3	SHA-1 with prediction resistance					
KAS-FFC-SSC	C	ffdhe2048				Shared Secret Computation	
KAS-ECC-SSC	C	P-256					
RSA	SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_SSSE3, SHA_CE, NEON	PKCS#1 v1.5 with SHA-256 and 2048-bit key				Signature Generation	
						Signature Verification	
ECDSA	SHA_ASM, SHA_SHANI, SHA_AVX2, SHA_AVX, SHA_SSSE3, SHA_CE	SHA-256 and P-224, P-256, P-384, and P-521				Signature Generation	
						Signature Verification	

Table 23 - Conditional Self-Tests

10.2.1 Conditional Cryptographic Algorithm 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 Table 23. 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.

10.2.2 Conditional Cryptographic Algorithm Tests

Upon generation of a DH, EC or RSA key pair, the module will perform a pair-wise consistency test (PCT) as shown in Table 23, which provides some assurance that the generated key pair is well formed. The test for DH consists of the PCT described in Section 5.6.2.1.4 of SP 800-56Ar3. For EC or RSA key pairs, the tests consist of performing signature generation and verification using the generated key pairs. Services are not available, and data output (via the data output interface) is inhibited during execution of the PCT. If a PCT test fails, the module transitions to the error state.

10.3 Periodic Self-Tests

The module does not implement any periodic self-tests.

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Error State	The module immediately stops functioning due to a self-test failure	Software integrity test failure	Restart of the module	Module will not load
		CAST failure		
		PCT failure		Module stops functioning

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

The software integrity tests and CASTs can be invoked on demand by unloading and subsequently re-initializing the module. Additionally, the integrity test may be invoked on-demand by calling the OSSL_PROVIDER_self_test function. The PCTs can be invoked on demand by requesting the Key Pair Generation service.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

The module is distributed as a part of the Oracle Linux 9 (OL9) RPM package in the form of openssl-libs-3.0.7-24.0.3.el9_fips RPM package that is located in the "Oracle Linux 9 Security Validation (Update 3)" yum repository (ol9_u3_security_validation).

The module can achieve the approved mode by:

- For installation add the fips=1 option to the kernel command line during the system installation. During the software selection stage, do not install any third-party software.
- Switching the system into the approved mode after the installation. Execute the fips-mode-setup –enable command. Restart the system.

In both cases, the Crypto Officer must verify the Oracle Linux 9 system operates in approved mode by executing the fips-mode-setup –check command, which should output "FIPS mode is enabled."

For more information on Oracle Linux 9 system FIPS validated environment, please see Oracle Linux 9 product documentation.

After installation of the openssl-libs-3.0.7-24.0.3.el9_fips RPM package, the Crypto Officer must execute the openssl list -providers command. The Crypto Officer must ensure that the FIPS provider is listed in the output as follows:

```
fips
  name: Oracle Linux Linux 9 OpenSSL FIPS Provider
  version: 3.0.7-b27cdeb3ba51be46
  status: active
```

The cryptographic boundary consists only of the FIPS provider as listed. If any other OpenSSL or third-party provider is invoked, the user is not interacting with the module specified in this Security Policy.

11.2 Administrator Guidance

The Approved and Non-Approved modes of operation are specified in Section 2.4. The administrative functions are specified in the Approved Services table. All the logical interfaces are specified in Section 3.1.

11.3 Non-Administrator Guidance

The approved and non-approved security functions available to users are listed in Section 2. The physical ports and logical interfaces available to users are specified in Section 3.1. The approved and non-approved modes of operation are specified in Section 2.4. All the algorithm-specific information is listed in Section 2.7.

11.4 Maintenance Requirements

There are no maintenance requirements.

11.5 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. Then, if desired, the openssl-libs-3.0.7-24.0.3.el9_fips RPM package can be uninstalled from the Oracle Linux 9 system.

12 Mitigation of Other Attacks

Certain cryptographic subroutines and algorithms are vulnerable to timing analysis. The module 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.

Glossary and Abbreviations

AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
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
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
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, Adleman
SHA	Secure Hash Algorithm
SSC	Shared Secret Computation
SSP	Sensitive Security Parameter
TOEPP	Tested Operational Environment’s Physical Perimeter
XTS	XEX-based Tweaked-codebook mode with cipher text Stealing

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 186-4	Digital Signature Standard (DSS) July 2013 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
FIPS 186-5	Digital Signature Standard (DSS) February 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 800-56Cr1	Recommendation for Key-Derivation Methods in Key-Establishment Schemes August 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr1.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