ORACLE
Linux

FIPS 140-3 Non-Proprietary Security Policy

Oracle Linux 8 Unbreakable Enterprise Kernel (UEK7) Cryptographic Module and Oracle Linux 9 Unbreakable Enterprise Kernel (UEK7) Cryptographic Module

FIPS 140-3 Level 1 Validation

Software Versions: kernel: 5.15.0-101.103.2.1.el8uek and 5.15.0-101.103.2.1.el9uek

libkcapi: 1.2.0-2.0.1.el8 and 1.3.1-3.0.1.el9

Date: July 3rd, 2024

Prepared by:

atsec information security corporation

4516 Seton Center Parkway, Suite 250

Austin, TX 78759

www.atsec.com

Title: Oracle Linux 8 Unbreakable Enterprise Kernel (UEK7) Cryptographic Module and Oracle Linux 9 Unbreakable Enterprise Kernel (UEK7) Cryptographic Module Security Policy

Date: July 3rd, 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®

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 Oracle Linux 8 Unbreakable Enterprise Kernel (UEK7) Cryptographic Module and Oracle Linux 9 Unbreakable Enterprise Kernel (UEK7) Cryptographic Module versions:

- kernel: 5.15.0-101.103.2.1.el8uek and 5.15.0-101.103.2.1.el9uek;
- libkcapi: 1.2.0-2.0.1.el8 and 1.3.1-3.0.1.el9

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. [Number Below]	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	Not Applicable

Table 1 - Security Levels

2 Cryptographic Module Specification

2.1 Description

Purpose and Use: The Oracle Linux 8 Unbreakable Enterprise Kernel (UEK7) Cryptographic Module and Oracle Linux 9 Unbreakable Enterprise Kernel (UEK7) Cryptographic Module (hereafter referred to as "the module") provides a C language application program interface (API) for use by other (kernel space and user space) processes that require cryptographic functionality. The module operates on a general-purpose computer as part of the Linux kernel. Its cryptographic functionality can be accessed using the Linux Kernel Crypto API.

Module Type: Software

Module Embodiment: Multi-chip standalone

Module Characteristics: N/A

Cryptographic Boundary: The cryptographic boundary of the module is defined as the kernel binary and the loadable kernel crypto object files, the libkcapi library, and the sha512hmac binary, which is used to verify the integrity of the software components. In addition, the cryptographic boundary contains the .hmac files which store the expected integrity values for each of the software components.

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.

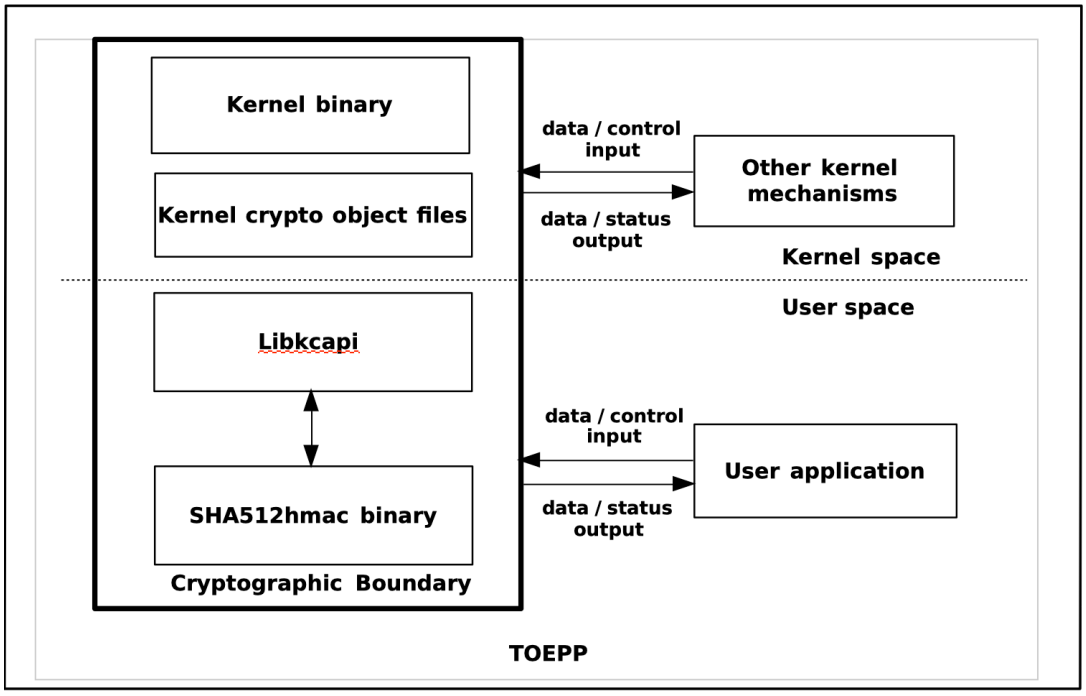

Figure 1 – Block Diagram

2.2 Version Information

Hardware Versions: N/A

Software Versions: kernel: 5.15.0-101.103.2.1.el8uek and 5.15.0-101.103.2.1.el9uek; libkcapi: 1.2.0-2.0.1.el8 and 1.3.1-3.0.1.el9

Firmware Versions: N/A

2.3 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 8	ORACLE SERVER X9-2c	Intel(R) Xeon(R) Platinum 8358	Yes: AES-NI and SHA Extensions	KVM on Oracle Linux 8
	ORACLE SERVER E4-2c	AMD EPYC 7J13	Yes: AES-NI and SHA Extensions	
	ORACLE SERVER A1-2c	Ampere(R) Altra(R) Q80-30	Yes: NEON and Cryptography Extensions	
Oracle Linux 9	ORACLE SERVER X9-2c	Intel(R) Xeon(R) Platinum 8358	Yes: AES-NI and SHA Extensions	
	ORACLE SERVER E4-2c	AMD EPYC 7J13	Yes: AES-NI and SHA Extensions	
	ORACLE SERVER A1-2c	Ampere(R) Altra(R) Q80-30	Yes: NEON and Cryptography Extensions	

Table 2 - Software, Firmware, Hybrid Tested Operating Environments

Executable Code Sets:

Package or File Names	Software/ Firmware Versions	Features	Hybrid Hardware Version	Integrity Test
Oracle Linux 8 Intel and AMD Platforms: /boot/vmlinuz-5.15.0-101.103.2.1.el8uek.x86_64 *.ko and *.ko.xz files in /usr/lib/modules/5.15.0-101.103.2.1.el8uek.x86_64/kernel/crypto *.ko and *.ko.xz files in /usr/lib/modules/5.15.0-101.103.2.1.el8uek.x86_64/kernel/arch/x86/crypto **Oracle Linux 8 Ampere Platform:** /boot/vmlinuz-5.15.0-101.103.2.1.el8uek.	5.15.0-101.103.2.1.el8uek	N/A	N/A	HMAC-SHA-512[1] RSA 4096 bit signature verification[2]

[1] HMAC-SHA-512 integrity test is used for the kernel binary.

[2] RSA 4096 bit signature verification integrity test is used for kernel object files.

Package or File Names	Software/ Firmware Versions	Features	Hybrid Hardware Version	Integrity Test
aarch64 *.ko and *.ko.xz files in /usr/lib/modules/5.15.0-101.103.2.1.el8uek.aarch64/kernel/crypto *.ko and *.ko.xz files in /usr/lib/modules/5.15.0-101.103.2.1.el8uek.aarch64/kernel/arch/arm64/crypto **Oracle Linux 9 Intel and AMD Platforms:** /boot/vmlinuz-5.15.0-101.103.2.1.el9uek.x86_64 *.ko and *.ko.xz files in /usr/lib/modules/5.15.0-101.103.2.1.el9uek.x86_64/kernel/crypto *.ko and *.ko.xz files in /usr/lib/modules/5.15.0-101.103.2.1.el9uek.x86_64/kernel/arch/x86/crypto **Oracle Linux 9 Ampere Platform:** /boot/vmlinuz-5.15.0-101.103.2.1.el9uek.aarch64 *.ko and *.ko.xz files in /usr/lib/modules/5.15.0-101.103.2.1.el9uek.aarch64/kernel/crypto *.ko and *.ko.xz files in /usr/lib/modules/5.15.0-101.103.2.1.el9uek.aarch64/kernel/arch/arm64/crypto	5.15.0-101.103.2.1.el9uek			
Oracle Linux 8 Platforms: /usr/lib64/libkcapi.so.1.2.0 /usr/bin/sha512hmac **Oracle Linux 9 Platforms:** /usr/lib64/libkcapi.so.1.3.1 /usr/bin/sha512hmac	1.2.0-2.0.1.el8 and 1.3.1-3.0.1.el9	N/A	N/A	HMAC-SHA-512

Table 3 - Executable Code Sets

Vendor Affirmed Operating Environments:

Operating Systems	Hardware Platforms	Virtual Platforms
Oracle Linux 8 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

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.

2.4 Excluded Components

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

2.5 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.

2.6 Approved Algorithms

Approved Algorithms:

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strengths	Use / Function
A3860 A3861 A3862 A3885 A3886 A3887 A3888 A3889 A3890 A3894 A3895 A4160 A4161 A4162 A4176 A4177 A4179 A4180 A4189 A4190 A4192 A4193	SHA [FIPS 180-4]	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	N/A	Message digest
A3863 A4163	SHA-3 [FIPS 202]	SHA3-224, SHA3-256, SHA3-384, SHA3-512	N/A	Message digest
A3860 A3861 A3862 A3864 A3865 A3866 A3867 A3868 A3869 A3870 A3871 A3872 A3873 A3874 A3876 A3877 A3878 A3879 A3880 A3881 A3882 A3883 A3884 A3889 A3891 A3892 A3893 A3894 A4160 A4161 A4162 A4164 A4165 A4166 A4167 A4168 A4169 A4170 A4171 A4172 A4173 A4174	AES [FIPS 197, SP 800-38A, SP 800-38A Addendum]	ECB, CBC, CBC-CS3, CFB128, OFB, CTR	128, 192, 256 bits	Encryption Decryption

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strengths	Use / Function
A4176 A4178 A4179 A4181 A4182 A4183 A4184 A4185 A4186 A4187 A4188 A4191 A4196 A4197				
A3862 A3869 A3879 A3889 A3891 A4162 A4169 A4176 A4178 A4183	AES [FIPS 197, SP 800-38C]	CCM	128, 192, 256 bits	Authenticated encryption Authenticated decryption
A3867 A3870 A3877 A3880 A3892 A4167 A4170 A4181 A4184 A4196	AES [FIPS 197, SP 800-38D]	GCM (internal IV)	128, 192, 256 bits	Authenticated encryption
A3862 A3868 A3869 A3871 A3876 A3878 A3879 A3881 A3891 A3893 A4162 A4168 A4169 A4171 A4178 A4182 A4183 A4185 A4191 A4197	AES [FIPS 197, SP 800-38D]	GCM (external IV)	128, 192, 256 bits	Authenticated decryption
A3862 A3869 A3876 A3879 A3889 A3891 A3894 A4162 A4169 A4176 A4178 A4179 A4183 A4191	AES [FIPS 197, SP 800-38E]	XTS	128, 256 bits	Encryption Decryption
A3862 A3869 A3879 A3889 A3891 A4162 A4169 A4176 A4178 A4183	AES [FIPS 197, SP 800-38B, SP 800-38D]	CMAC, GMAC	128, 192, 256 bits	Message authentication
A3860 A3861 A3862 A3885 A3886 A3887 A3888 A3889 A3890 A3894 A3895 A4160 A4161 A4162 A4176 A4177 A4179 A4180 A4189 A4190 A4192 A4193	HMAC [FIPS 198-1]	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	112-256 bits	Message authentication
A3863 A4163		SHA3-224, SHA3-256, SHA3-384, SHA3-512		
A3861 A4161	ECDSA [FIPS 186-4]	B.4.2 Testing candidates	P-256, P-384 (128 and 192 bits)	Key pair generation
A3861 A4161	KAS ECC-SSC [SP 800-56Arev3]	Ephemeral Unified Model (initiator/responder)	P-256, P-384 (128 and 192 bits)	Shared secret computation
A3860 A4160	KAS FFC-SSC [SP 800-56Arev3]	dhEphem (initiator/responder)	ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 (112-200 bits)	Shared secret computation
A3860 A4160	Safe Primes [SP 800-56Arev3]	Section 5.6.1.1.4 Testing Candidates	ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 (112-200 bits)	Key pair generation
A3860 A3861 A3862 A3867 A3868 A3869	CTR_DRBG [SP 800-90Ar1]	AES-128, AES-192, AES-256, with derivation function,	128, 192, 256 bits	Random number generation

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strengths	Use / Function
A3870 A3871 A3876 A3877 A3878 A3879 A3880 A3881 A3891 A3892 A3893 A4160 A4161 A4162 A4167 A4168 A4169 A4170 A4171 A4178 A4181 A4182 A4183 A4184 A4185 A4191 A4196 A4197		with/without prediction resistance		
A3860 A3861 A3862 A3867 A3868 A3869 A3870 A3871 A3876 A3877 A3878 A3879 A3880 A3881 A3885 A3886 A3887 A3892 A3893 A4160 A4161 A4162 A4167 A4168 A4169 A4170 A4171 A4181 A4182 A4183 A4184 A4185 A4189 A4190 A4191 A4192 A4196 A4197	Hash_DRBG [SP 800-90Ar1]	SHA-1, SHA-256, SHA-512 with/without prediction resistance	128, 256 bits	Random number generation
	HMAC_DRBG [SP 800-90Ar1]	SHA-1, SHA-256, SHA-512 with/without prediction resistance	128, 256 bits	Random number generation
A3862 A3885 A3886 A3887 A4162 A4189 A4190 A4192	RSA [FIPS 186-4]	PKCS#1 v1.5 with SHA1, SHA-224, SHA-256, SHA-384, SHA-512	2048, 3072, 4096 bits (112, 128, 150 bits)	Signature verification

Table 6 - Approved Algorithms

Vendor Affirmed Algorithms:

The module implements Cryptographic Key Generation (CKG), with vendor affirmed compliance to SP 800-133r2, Section 5.2.

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:

Name	Use and Function
ECDSA Signature Verification Primitive	Signature Verification Primitive

Table 7 - Non-Approved, Allowed Algorithms with No Security Claimed

Non-Approved, Not Allowed Algorithms:

Name	Use and Function
AES GCM with external IV	Encryption
KBKDF (libkcapi)	Key derivation
HKDF (libkcapi)	Key derivation
PBKDF2 (libkcapi)	Password-based key derivation
RSA	Encryption primitive Decryption primitive
RSA with PKCS#1 v1.5 padding	Signature generation (pre-hashed message) Signature verification (pre-hashed message)
	Key encapsulation Key decapsulation

Table 8 – Non-Approved, Not Allowed Algorithms

2.7 RNG and Entropy

Entropy Information:

Name	Type	Operational Environment	Sample Size	Entropy Per Sample	Conditioning Component
UEK7 CPU Time Jitter RNG Entropy Source (Cert. #E61)	Non-physical	See Table 2	64 bits	59.41 bits	Linear-Feedback Shift Register (LFSR)

Table 9 – Entropy

RNG Information:

The module implements three different Deterministic Random Bit Generator (DRBG) implementations based on SP 800-90Ar1: CTR_DRBG, Hash_DRBG, and HMAC_DRBG. Each of these DRBG implementations can be instantiated by the operator of the module, using the parameters listed in Table 6. When instantiated, these DRBGs can be used to generate random numbers for external usage.

Additionally, the module employs a specific HMAC SHA-512 DRBG implementation for internal purposes (e.g. to generate asymmetric key pairs or initialization vectors). This DRBG is initially seeded with 448 output bits from the entropy source (415 bits of entropy) and reseeded with 320 output bits from the entropy source (297 bits of entropy).

2.8 SSP Generation

Name	Type	Properties
Safe primes key pair generation	Asymmetric	Key type: Diffie-Hellman key pair Groups: 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 SP 800-133r2, Section 5.2
EC key pair generation		Key type: EC key pair Curves: P-256, P-384 Security strength: 128, 192 bits Method: FIPS 186-4 Appendix B.4.2 Testing Candidates Compliant to SP 800-133r2, Section 5.2

Table 10 - Key Generation

2.9 SSP Establishment

Name	Type	Properties
Diffie-Hellman	Shared secret computation	Groups: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 Security strength: 112-200 bits Compliant with Scenario 2 (1) of FIPS 140-3 IG D.F
EC Diffie-Hellman	Shared secret computation	Curves: P-256, P-384 Security strength: 128, 192 bits Compliant with Scenario 2 (1) of FIPS 140-3 IG D.F

Table 11 - Key Agreement

Name	Type	Properties
AES CCM	Key wrapping Key unwrapping	Security strength: 128, 192, or 256 bits SP 800-38F
AES GCM	Key wrapping	IV generated internally Security strength: 128, 192, or 256 bits SP 800-38F
	Key unwrapping	IV provided externally Security strength: 128, 192, or 256 bits SP 800-38F
AES CBC or CTR with HMAC SHA-1, HMAC SHA-256, HMAC SHA-384, or HMAC SHA-512	Key wrapping Key unwrapping	Security strength: 112-256 bits SP 800-38F

Table 12 - Key Transport

As permitted by IG D.G, the module provides key transport methods either by using an approved authenticated encryption mode or by a combination of any approved symmetric encryption mode and an approved authentication method.

2.10 Industry Protocols

AES GCM with internal IV generation in the approved mode is compliant with RFC 4106 and shall only be used in conjunction with the IPsec protocol. No parts of this protocol, other than the AES GCM implementation, have been tested by the CAVP and CMVP.

2.11 Design and Rules

Upon start-up, the module immediately performs the pre-operational integrity test using the integrity values stored in the .hmac files associated with the module's software components (Table 23). When all those self-tests pass successfully, the module automatically performs all cryptographic algorithm self-tests (CASTs) as specified in Table 23. Only if these CASTs also passed successfully, the module transitions to the operational state. No operator intervention is required to reach this point.

In the operational state, the module accepts service requests from calling applications through its logical interfaces. If the Linux kernel which contains the module is shut down, the module will end its operation.

2.12 Initialization

There are no specific initialization requirements.

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, AF_ALG type sockets
	Data Output	API output parameters, AF_ALG type sockets
	Control Input	API function calls, API control input parameters, AF_ALG type sockets, kernel command line
	Status Output	API return values, AF_ALG type sockets, kernel logs

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

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	SSP Access
Message digest	Compute a message digest	crypto_shash_init returns 0	Message	Digest value	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512	N/A
Encryption	Encrypt a plaintext	crypto_skcipher_setkey returns 0	AES key, plaintext	Ciphertext	AES ECB, CBC, CBC-CS3, OFB, CFB128, CTR, XTS	AES key: W, E
Decryption	Decrypt a ciphertext		AES key, ciphertext	Plaintext		
Authenticated encryption	Encrypt a plaintext	For all modes except AES GCM: crypto_aead_setkey returns 0 For AES GCM: crypto_aead_get_flags(tfm) has the CRYPTO_TFM_FIPS_COMPLIANCE flag set	AES key, plaintext	Ciphertext, MAC tag	AES CCM, GCM (internal IV) AES CBC or CTR with HMAC SHA-1, HMAC SHA-256, HMAC SHA-384, or HMAC SHA-512	AES key: W, E HMAC key: W, E
Authenticated decryption	Decrypt a ciphertext		AES key, ciphertext, MAC tag	Plaintext	AES CCM, GCM (external IV) AES CBC or CTR with HMAC SHA-1, HMAC SHA-256, HMAC SHA-384, or HMAC SHA-512	
Message authentication	Compute a MAC tag	crypto_shash_init returns 0	AES key, message	MAC tag	AES CMAC, GMAC	AES key: W, E
			HMAC key, message		HMAC SHA-1, HMAC SHA-224, HMAC SHA-256, HMAC SHA-384, HMAC SHA-512, HMAC SHA3-224, HMAC SHA3-256, HMAC SHA3-384, HMAC SHA3-512	HMAC key: W, E
Signature verification primitive	Perform ECDSA signature verification primitive	crypto_akcipher_verify returns 0	Hashed message	Pass/fail	ECDSA signature verification primitive with P-256 and P-384	EC public key: W, E
Shared secret computation	Compute a shared secret	crypto_kpp_compute_shared_secre	DH private key, DH	Shared secret	KAS-FFC-SSC	DH private key: W, E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		t returns 0	public key			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 pair generation	Generate a key pair	crypto_kpp_set_secret and crypto_kpp_generate_public_key return 0	Group	DH private key, DH public key	Safe primes key pair generation	DH private key: G, R DH public key: G, R Intermediate key generation value: G, Z
			Curve	EC private key, EC public key	EC key pair generation	EC private key: G, R EC public key: G, R Intermediate key generation value: G, Z
Random number generation	Generate random bytes	crypto_rng_get_bytes returns 0	Output length	Random bytes	CTR_DRBG Hash_DRBG HMAC_DRBG	Entropy input: W, E DRBG seed: E, G DRBG internal state: E, G
Error detection code	Compute an EDC (crc32, crct10dif)	None	Message	EDC	N/A	N/A
Compression	Compress data (deflate, lz4, lz4hc, lzo, zlib-deflate, zstd)	None	Data	Compressed data	N/A	N/A
Generic system call	Use the kernel to perform various non-cryptographic operations	None	Identifier, various arguments	Various return values	N/A	N/A
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 SHA-3 AES HMAC KAS-FFC-SSC KAS-ECC-SSC CTR_DRBG Hash_DRBG HMAC_DRBG RSA 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 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.

4.4 Non-Approved Services

Name	Description	Security Functions	Role
AES GCM external IV encryption	Encrypt a plaintext using AES GCM with an external IV	AES GCM (external IV)	CO
Key derivation	Derive a key from a key-derivation key or a shared secret	KBKDF (libkcapi) HKDF (libkcapi)	
Password-based key derivation	Derive a key from a password	PBKDF2 (libkcapi)	
RSA encryption primitive	Compute the raw RSA encryption of a plaintext/ciphertext	RSA	
RSA decryption primitive	Compute the raw RSA decryption of a plaintext/ciphertext		
RSA signature generation (pre-hashed message)	Generate a digital signature for a pre-hashed message	RSA with PKCS#1 v1.5 padding (pre-hashed message)	
RSA signature verification (pre-hashed message)	Verify a digital signature for a pre-hashed message		
Key encapsulation	Encapsulate a secret key using RSA with PKCS#1 v1.5 padding		
Key decapsulation	Decapsulate a secret key using RSA with PKCS#1 v1.5 padding		

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 Linux kernel binary is integrity tested using an HMAC SHA-512 calculation performed by the sha512hmac utility (which utilizes the module's HMAC and SHA-512 implementations) which compares the computed HMAC value with a precomputed HMAC value. An HMAC SHA-512 calculation is also performed on the sha512hmac utility and the libkcapi library to verify their integrity by comparing the computed HMAC value with a precomputed HMAC value. The kernel crypto object files listed in Table 3 are loaded on start-up by the module and verified using RSA signature verification with PKCS#1 v1.5 padding, SHA-512, and a 4096-bit key.

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 software integrity tests.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operating Environment: modifiable: the module executes as part of a general-purpose operating system (Oracle Linux 8 and 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.

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 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	Dynamic

Table 17 – 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	Related SFI
API input parameters	Operator calling application (TOEPP)	Cryptographic module	Plaintext	Manual	Electronic	N/A
AF_ALG_type sockets (input)						
API output parameters	Cryptographic module	Operator calling application (TOEPP)				
AF_ALG type sockets (output)						

Table 18 – SSP Input-Output

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
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	By calling the appropriate zeroization functions: AES key: crypto_free_skcipher and crypto_free_aead HMAC key: crypto_free_shash and crypto_free_ahash DRBG internal state: crypto_free_rng Intermediate key generation value: automatically zeroized DH public & private key: crypto_free_kpp EC public & private key: crypto_free_kpp and crypto_free_akcipher
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

All data output is inhibited during zeroization.

9.4 SSPs

Name	Description	Size - Strength	Type – Category	Generated By	Established By	Used By
AES key	AES key used for encryption, decryption, and computing MAC tags	128, 192, 256 bits	Symmetric Key	N/A	N/A	Encryption Decryption Authenticated encryption Authenticated decryption Message authentication
HMAC key	HMAC key	112-256 bits	Authentication key	N/A	N/A	Message authentication code (MAC)
Shared secret	Shared secret generated by (EC) Diffie-Hellman	EC Diffie-Hellman 128 and 192 bits Diffie-Hellman 112-200 bits	Shared secret	N/A	SP 800-56Ar3 (DH and ECDH shared secret computation)	Shared secret computation
Entropy input	Entropy input used to seed the DRBGs. Compliant with IG D.L.	128-448 bits (128-256 bits)	Entropy input	ENT (NP) See Table 9	N/A	Random number generation
DRBG seed	DRBG seed derived from entropy input. Compliant with IG D.L.	CTR_DRBG: 128, 192, 256 bits Hash_DRBG: 128, 256 bits HMAC_DRBG: 128, 256 bits	Seed	Derived from the entropy input as defined in SP800-90Arev1	N/A	Random number generation
DRBG internal state (V, Key)	Internal state of CTR_DRBG and HMAC_DRBG instances. Compliant with IG D.L.		DRBG Internal state	Derived from DRBG seed as defined in SP800-90Arev1	N/A	Random number generation
DRBG internal state (V, C)	Internal state of Hash_DRBG instances. Compliant with IG D.L.		DRBG Internal state		N/A	Random number generation
Intermediate Key Generation Value	Temporary value generated during Key Pair Generation services	ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, P-256, P-384 (112-200 bits)	Intermediate value	CKG	N/A	Key pair generation
DH public key	Public key used for Diffie-Hellman	ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 (112-200 bits)	Public key	SP 800-56Ar3 (safe primes) Section 5.6.1.1.4 Testing Candidates random values used are generated by SP 8000-90ADRBG	N/A	Shared secret computation Key pair generation
DH private key	Private key used for Diffie-Hellman		Private key			
EC public key	Public key used for EC Diffie-Hellman	P-256, P-384 (128, 192 bits)	Public key	FIPS 186-4 Appendix B.4.2 Testing Candidates random values	N/A	Shared secret computation Key pair generation
EC private key	Private key used for EC Diffie-		Private key			Shared secret computation

Name	Description	Size - Strength	Type – Category	Generated By	Established By	Used By
	Hellman			used are generated by SP 8000-90ADRBG		Key pair generation

Table 20 - SSP Information First

Name	Input - Output	Storage	Storage Duration	Type	Related SSPs
AES key	API input parameters AF_ALG type sockets (input)	RAM	Until cipher handled is freed or module powered off	CSP	None
HMAC key					
Shared secret	API output parameters AF_ALG type sockets (output)				DH public & private key EC public & private key
Entropy input	N/A		From generation until DRBG seed is created		DRBG seed
DRBG seed	N/A		While the DRBG is being instantiated		Entropy input DRBG internal state
DRBG internal state (V, Key)	N/A		From DRBG instantiation until DRBG termination		DRBG seed
DRBG Internal state (V, C)	N/A				DRBG seed
Intermediate Key Generation Value	No input No output		Until key pair generation service completes.		DH Private Key, DH Public Key, EC Private Key, EC Public Key
DH public key	API input parameters AF_ALG type sockets (input) API output parameters AF_ALG type sockets (output)		Until cipher handled is freed or module powered off	PSP	DH private key, shared secret
DH private key				CSP	DH public key, shared secret
EC public key				PSP	EC private key, shared secret
EC private key				CSP	EC public key, shared 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 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 SHA-512	Generic C, ARM64	128-bit key	Message Authentication	Software integrity	Module becomes operational	Integrity test for vmlinuz, libkcapi components and sha512hmac binary
RSA PKCS#1 v1.5	Generic C	4096-bit key with SHA-512	Signature Verification			Integrity test for kernel object files

Table 22 - Pre-Operational Self-Tests

The pre-operational software integrity tests are performed automatically when the module is powered on, before the module transitions into the operational state. The algorithms used for the integrity test (i.e., HMAC-SHA2-512 and RSA SigVer with 4096 bit key) run their CASTs before the 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 pre-operational software integrity self-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	Implementation	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
Safe primes CKG	Generic C	N/A	PCT	PCT	crypto_kpp_generate_public_key returns 0	SP 800-56Ar3 Section 5.6.2.1.4	Key pair generation
EC CKG							
SHA-1	Generic C, SSSE3, AVX, AVX2, SHA_NI, CE	0-8184 bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA-224	Generic C, SSSE3, AVX, AVX2, SHA_NI, CE, ARM64, ARM64-NEON						
SHA-256							
SHA-384	Generic C, SSSE3, AVX, AVX2, SHA_NI, ARM64,						
SHA-512							
SHA3-224	Generic C						
SHA3-256							
SHA3-384							
SHA3-512							
AES ECB	Generic C, AESNI, CE, ARM64	128, 192, 256 bit keys				Encryption Decryption (separately)	
AES CBC							
AES CBC-CS3	AESNI, CE, ARM64	128 bit keys					
AES OFB	AESNI, ARM64						
AES CFB128	AESNI ARM64	128, 192, 256 bit keys					
AES CTR	Generic C, AESNI, CE,						

Algorithm	Implementation	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
	ARM64						
AES CCM	AESNI, ARM64	128, 192, 256 bit keys 128-bit IVs					
AES GCM (internal IV)	AESNI, CE	128, 192, 256 bit keys 96-bit IVs				Encryption	
AES GCM (external IV)		128, 192, 256 bit keys				Decryption	
AES XTS	AESNI, CE, ARM64	128 and 256 bit keys				Encryption Decryption (separately)	
AES CMAC		128 and 256 bit keys				Message authentication	
HMAC SHA-1	SHA_NI, CE	32-64 bit keys					
HMAC SHA-224		32-1048 bit keys					
HMAC SHA-256	Generic C, SHA_NI, CE	32-64 bit keys					
HMAC SHA-384	AVX2, ARM64	32-1048 bit keys					
HMAC SHA-512	Generic C, ARM64	32-1048 bit keys					
HMAC SHA3-224	Generic C	32-1048 bit keys					
HMAC SHA3-256		32-1048 bit keys					
HMAC SHA3-384		32-1048 bit keys					
HMAC SHA3-512		32-1048 bit keys					
KAS-FFC-SSC		ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192				Shared secret computation	
KAS-ECC-SSC		P-256, P-384					
CTR_DRBG		128, 192, 256 bit keys With/without PR Health test per section 11.3 of SP 800-90A				Seed Generate	
Hash_DRBG		SHA-256 With/without PR Health test per section 11.3 of SP 800-90A					
HMAC_DRBG		SHA-256, SHA-512 With/without PR Health test per section 11.3 of SP 800-90A					
RSA PKCS#1 v1.5		4096-bit key with SHA-256				Verify	
ENT (NP)		1024 samples	RCT			Entropy source start-up test	Entropy source initialization
		1024 samples	APT				
		N/A	RCT		Entropy source is operational	Entropy source continuous test	Continuously
		N/A	APT				

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 conditional 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 or EC 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. This test consists of the PCT described in Section 5.6.2.1.4 of SP 800-56Ar3. Services are not available, and data output (via the data output interface) is inhibited during execution of the PCT. If the 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 Linux kernel immediately stops executing	Any self-test failure	Restart of the module	Kernel Panic

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, cryptographic algorithm self-tests, and entropy source start-up tests can be invoked on demand by unloading and subsequently re-initializing the module. The pair-wise consistency tests can be invoked on demand by requesting the key pair generation service.

11 Life-Cycle Assurance

11.1 Startup Procedures

The module is distributed as a part of the Oracle Linux 8 (OL8) and Oracle Linux 9 (OL9) RPM packages kernel-uek-5.15.0-101.103.2.1.el8uek and kernel-uek-5.15.0-101.103.2.1.el9uek, libkcapi-1.2.0-2.0.1.el8 and libkcapi-1.3.1-3.0.1.el9, libkcapi-hmaccalc-1.2.0-2.0.1.el8 and libkcapi-hmaccalc-1.3.1-3.0.1.el9.

The modules 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 OL8 and OL9 systems operate in the approved mode by executing the fips-mode-setup --check command, which should output "FIPS mode is enabled."

After installation of the kernel-uek-5.15.0-101.103.2.1.el8uek and kernel-uek-5.15.0-101.103.2.1.el9uek, libkcapi-1.2.0-2.0.1.el8 and libkcapi-1.3.1-3.0.1.el9, libkcapi-hmaccalc-1.2.0-2.0.1.el8 and libkcapi-hmaccalc-1.3.1-3.0.1.el9 RPM packages, 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:

Oracle Linux 8:

Oracle Linux 8 Unbreakable Enterprise Kernel 7 Crypto API

Oracle Linux 9:

Oracle Linux 9 Unbreakable Enterprise Kernel 7 Crypto API

Then, the Crypto Officer must execute the "cat /proc/sys/crypto/fips_version" and "rpm -q libkcapi" commands. These commands must output the following (one line per output):

Oracle Linux 8 Intel and AMD Platforms:

5.15.0-101.103.2.1.el8uek.x86_64

libkcapi-1.2.0-2.0.1.el8.x86_64

Oracle Linux 8 Ampere Platform:

5.15.0-101.103.2.1.el8uek.aarch64

libkcapi-1.2.0-2.0.1.el8.aarch64

Oracle Linux 9 Intel and AMD Platforms:

5.15.0-101.103.2.1.el9uek.x86_64

libkcapi-1.3.1-3.0.1.el9.x86_64

Oracle Linux 9 Ampere Platform:

5.15.0-101.103.2.1.el9uek.aarch64

libkcapi-1.3.1-3.0.1.el9.aarch64

11.2 Administrator Guidance

11.2.1 AES GCM IV

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

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.

The module does not implement IPsec. The module's implementation of AES GCM is used together with an application that runs outside the module's cryptographic boundary. This application must use RFC 7296 compliant IKEv2 to establish the shared secret SKEYSEED from which the AES GCM encryption keys are derived.

The design of the IPsec 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.

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 crypto_aead_encrypt API function with an AES GCM handle. When this is the case, the API will not set an approved service indicator, as described in Table 15.

11.2.2 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.

11.2.3 SP 800-56Arev3 Assurances

To comply with the assurances found in Section 5.6.2 of SP 800-56Ar3, the operator must use elliptic curve Diffie-Hellman shared secret computation algorithm with Bluetooth protocol. Additionally, the module's approved key pair generation service (see section 2.8) 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 DH public key, and the partial public key validation of the peer EC public key, complying with Section 5.6.2.2.2 of SP 800-56Ar3. The module is compliant to IG D.F scenario 2 path (1).

11.2.4 RSA

For RSA signature verification, the module supports modulus sizes 2048, 3072, and 4096 bits compliant to IG C.F. All supported modulus sizes have been CAVP tested.

11.3 Non-Administrator Guidance

There is no non-administrator guidance.

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 kernel-uek-5.15.0-101.103.2.1.el8uek and kernel-uek-5.15.0-101.103.2.1.el9uek, libkcapi-1.2.0-2.0.1.el8 and libkcapi-1.3.1-3.0.1.el9, libkcapi-hmaccalc-1.2.0-2.0.1.el8 and libkcapi-hmaccalc-1.3.1-3.0.1.el9 RPM packages can be uninstalled from the Oracle Linux 8 and Oracle Linux 9 systems.

12 Mitigation of Other Attacks

The module does not offer mitigation of other attacks and therefore this section is not applicable.

Appendix A. 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
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECDH	Elliptic Curve Diffie-Hellman
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliiptic Curve Digital Signature Algorithm
ENT (NP)	Non-physical Entropy Source
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, Adleman
SHA	Secure Hash Algorithm
SSC	Shared Secret Computation
SSP	Sensitive Security Parameter
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 186-4	Digital Signature Standard (DSS) July 2013 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.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
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 800-56Ar3	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