ORACLE
Linux

FIPS 140-2 Non-Proprietary Security Policy

Oracle Linux 8 libgcrypt Cryptographic Module

FIPS 140-2 Level 1 Validation

Software Version: R8-8.4.0

Date: May 11th, 2022

Title: Oracle Linux 8 libgcrypt Cryptographic Module Security Policy
Date: May 11th, 2022
Author: Oracle Security Evaluations – Global Product Security
Contributing Authors:
Atsec Information Security
Oracle Linux Engineering

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

TABLE OF CONTENTS

List of Tables

List of Figures

1. Introduction

1.1 Overview

This document is the Security Policy for the Oracle Linux 8 libgcrypt Cryptographic Module by Oracle Corporation. Oracle Linux 8 libgcrypt Cryptographic Module is also referred to as “the Module or Module”. This Security Policy specifies the security rules under which the module shall operate to meet the requirements of FIPS 140-2 Level 1. It also describes how the Oracle Linux 8 libgcrypt Cryptographic Module functions in order to meet the FIPS requirements, and the actions that operators must take to maintain the security of the module.

This Security Policy describes the features and design of the Oracle Linux 8 libgcrypt Cryptographic Module using the terminology contained in the FIPS 140-2 specification. FIPS 140-2, Security Requirements for Cryptographic Module specifies the security requirements that will be satisfied by a cryptographic module utilized within a security system protecting sensitive but unclassified information. The NIST/CCCS Cryptographic Module Validation Program (CMVP) validates cryptographic module to FIPS 140-2. Validated products are accepted by the Federal agencies of both the USA and Canada for the protection of sensitive or designated information.

1.2 Document Organization

The FIPS 140-2 submission package contains:

- Oracle Linux 8 libgcrypt Cryptographic Module Non-Proprietary Security Policy
- Other supporting documentation as additional references

With the exception of this Non-Proprietary Security Policy, the FIPS 140-2 Validation Documentation is proprietary to Oracle and is releasable only under appropriate non-disclosure agreements. For access to these documents, please contact Oracle.

2. Oracle Linux 8 libgcrypt Cryptographic Module

2.1 Functional Overview

The Oracle Linux 8 libgcrypt Cryptographic Module is a software library implementing general purpose cryptographic algorithms. The module provides cryptographic services to applications running in the user space of the underlying operating system through an application program interface (API).

2.2 FIPS 140-2 Validation Scope

The following table shows the security level for each of the eleven sections of the validation. See Table 1 below.

Security Requirements Section	Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles and Services and Authentication	1
Finite State Machine Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	3
Mitigation of Other Attacks	1

Table 1: FIPS 140-2 Security Requirements

ORACLE

3. Cryptographic Module Specification

3.1 Definition of the Cryptographic Module

The Oracle Linux 8 libgcrypt Cryptographic Module is defined as a software only multi-chip standalone module as defined by the requirements within FIPS PUB 140-2. The logical cryptographic boundary of the module consists of shared library file and its integrity check HMAC file, which are delivered through the Oracle Yum Server Package Manager (RPM) as listed below:

All components of the module will be in the libgcrypt RPM version libgcrypt-1.8.5-4.0.1.el8.x86_64.rpm or libgcrypt-1.8.5-4.0.1.el8.aarch64.rpm.

When installed on the system, the module comprises the following files:

- /usr/lib64/libgcrypt.so.20.2.5
- /usr/lib64/.libgcrypt.so.20.hmac

Figure 1 shows the logical block diagram of the module executing in memory on the host system.

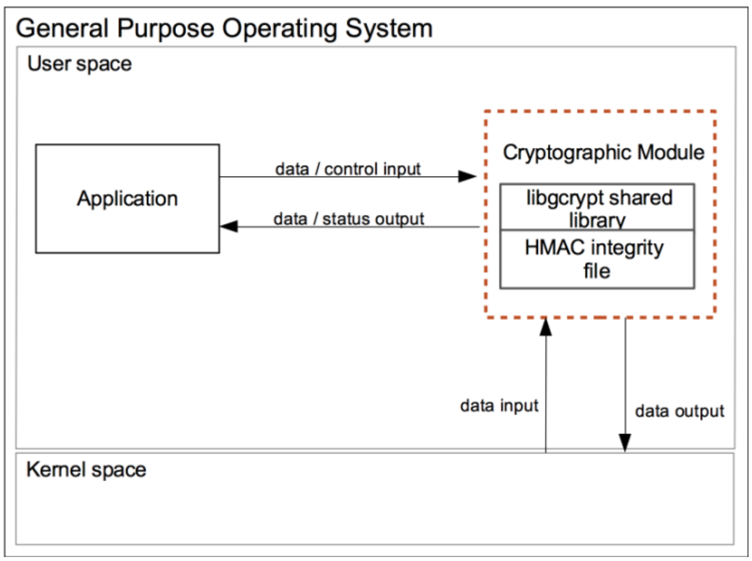

Figure 1: Oracle Linux 8 libgcrypt Logical Cryptographic Boundary

3.2 Definition of the Physical Cryptographic Boundary

The module is aimed to run on a general-purpose computer. No components are excluded from the requirements of FIPS PUB 140-2. The physical boundary is the surface of the case of the target platform, as shown in the diagram below:

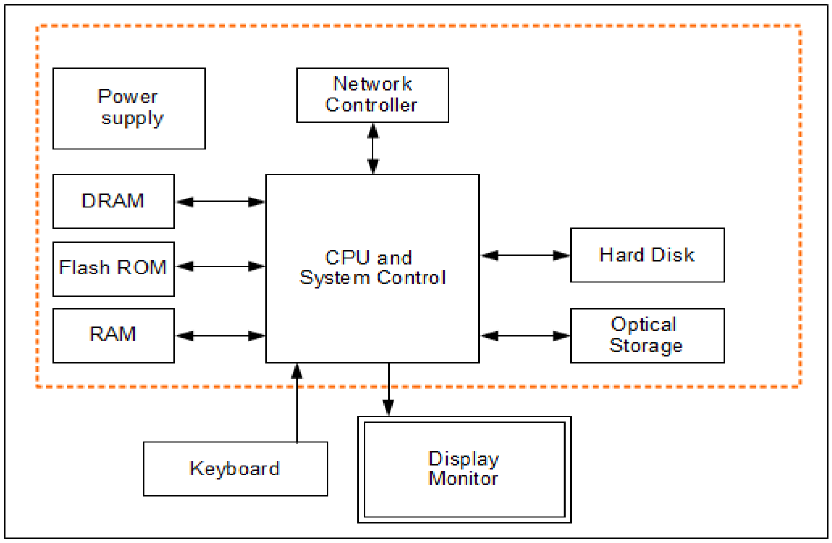

Figure 2: Oracle Linux 8 libgcrypt Hardware Block Diagram

3.3 Modes of Operation

The module supports two modes of operation: FIPS approved and non-approved modes. The mode is implicitly assumed depending on the services and security functions invoked. The module turns to the FIPS approved mode after the initialization and the power-on self-tests have completed successfully. Using a non-Approved service listed in Table 9 will result in the module implicitly entering the non-FIPS mode of operation

3.4 Approved Security Functions

The Oracle Linux 8 libgcrypt Cryptographic Module contains the following FIPS Approved Algorithms listed in the table below. Note that not all algorithms/modes tested with a corresponding CAVP cert are implemented/used by the module.

Approved or Allowed Security Functions		Certificate
Symmetric Algorithms		
AES	**SSSE3:** AES in CBC, ECB, CFB8, CFB128, OFB, CTR, KW, CCM, CMAC and XTS modes (Key sizes 128, 192, 256 for all modes except XTS Mode where key sizes are 128 and 256)	A 1785
	Full Acceleration: AES in CBC, ECB, CFB8, CFB128, OFB, CTR, KW, CCM, CMAC and XTS modes (Key sizes 128, 192, 256 for all modes except XTS Mode where key sizes are 128 and 256)	A 1786
	No Acceleration: AES in CBC, ECB, CFB8, CFB128, OFB, CTR, KW, CCM, CMAC and XTS modes (Key sizes 128, 192, 256 for all modes except XTS Mode where key sizes are 128 and 256)	A 1787
	AESNI_AVX: AES in CBC, ECB, CFB8, CFB128, OFB, CTR, KW, CCM, CMAC and XTS modes (Key sizes 128, 192, 256 for all modes except XTS Mode where key sizes are 128 and 256)	A 1788
Triple DES	**Full Acceleration:** CBC, ECB, CFB8, CFB64, OFB, CTR and CMAC (KO 1, d/e)	A 1786
Secure Hash Standard (SHS)		

Approved or Allowed Security Functions		Certificate
SHS	**AESNI_BMI2:** SHA-1	A 1784
	SSSE3: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	A 1785
	Full Acceleration: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE-128, SHAKE-256	A 1786
	No Acceleration: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE-128, SHAKE-256	A 1787
	AESNI_AVX: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	A 1788
	SHLD: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE-128, SHAKE-256	A 1789
Data Authentication Code		
HMAC	**AESNI_BMI2:** HMAC-SHA-1	A 1784
	SSSE3: HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	A 1785
	Full Acceleration: 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	A 1786
	No Acceleration: 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	A 1787
	AESNI_AVX: HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	A 1788
	SHLD: 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	A 1789
Asymmetric Algorithms		
RSA	**SSSE3:** **FIPS186-4:** PKCS 1.5 (Key Gen); Modulus Sizes 2048, 3072, 4096 PKCS 1.5 (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512 PKCS 1.5 (Sig Ver) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512 PSS (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512 PSS (Sig Ver) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512	A 1785
	Full Acceleration: **FIPS186-4:** PKCS 1.5 (Key Gen); Modulus Sizes 2048, 3072, 4096	A 1786

Approved or Allowed Security Functions		Certificate
	PKCS 1.5 (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512 PKCS 1.5 (Sig Ver) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512 PSS (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512 PSS (Sig Ver) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512	
	No Acceleration: PKCS 1.5 (Key Gen); Modulus Sizes 2048, 3072, 4096 PKCS 1.5 (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512 PKCS 1.5 (Sig Ver) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512 PSS (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512 PSS (Sig Ver) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512	A 1787
	AESNI_AVX: PKCS 1.5 (Key Gen); Modulus Sizes 2048, 3072, 4096 PKCS 1.5 (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512 PKCS 1.5 (Sig Ver) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512 PSS (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512 PSS (Sig Ver) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512	A 1788
	SHLD: PKCS 1.5 (Key Gen); Modulus Sizes 2048, 3072, 4096 PKCS 1.5 (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512 PKCS 1.5 (Sig Ver) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512 PSS (Sig Gen) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512 PSS (Sig Ver) Modulus Sizes 2048, 3072, 4096 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512	A 1789
DSA	**SSSE3:** **FIPS186-4:** Key Gen, PQG Gen, PQG Ver, Sig Gen, Sig Ver; Modulus Sizes 2048, 3072, with hash sizes SHA-224, SHA-256; (Modulus size 1024 and SHA-1 allowed for Sig Ver operations only)	A 1785
	Full Acceleration: **FIPS186-4:** Key Gen, PQG Gen, PQG Ver, Sig Gen, Sig Ver; Modulus Sizes 2048, 3072, with hash sizes SHA-224, SHA-256; (Modulus size 1024 and SHA-1 allowed for Sig Ver operations only)	A 1786
	No Acceleration:	A 1787

Approved or Allowed Security Functions		Certificate
	FIPS186-4: Key Gen, PQG Gen, PQG Ver, Sig Gen, Sig Ver; Modulus Sizes 2048, 3072, with hash sizes SHA-224, SHA-256; (Modulus size 1024 and SHA-1 allowed for Sig Ver operations only)	
	<u>AESNI_AVX:</u> **FIPS186-4:** Key Gen, PQG Gen, PQG Ver, Sig Gen, Sig Ver; Modulus Sizes 2048, 3072, with hash sizes SHA-224, SHA-256; (Modulus size 1024 and SHA-1 allowed for Sig Ver operations only)	A 1788
	<u>SHLD:</u> **FIPS186-4:** Key Gen, PQG Gen, PQG Ver, Sig Gen, Sig Ver; Modulus Sizes 2048, 3072, with hash sizes SHA-224, SHA-256; (Modulus size 1024 and SHA-1 allowed for Sig Ver operations only)	A 1789
ECDSA	**<u>SSSE3:</u>** Key Gen, Key Ver; Curves P-256, P-384, P-521 Sig Gen, Sig Ver; Curves P-224, P-256, P-384, P-521 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512; (SHA-1 allowed for Sig Ver operations only)	A 1785
	<u>Full Acceleration:</u> Key Gen, Key Ver; Curves P-256, P-384, P-521 Sig Gen, Sig Ver; Curves P-224, P-256, P-384, P-521 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512; (SHA-1 allowed for Sig Ver operations only)	A 1786
	<u>No Acceleration:</u> Key Gen, Key Ver; Curves P-256, P-384, P-521 Sig Gen, Sig Ver; Curves P-224, P-256, P-384, P-521 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512; (SHA-1 allowed for Sig Ver operations only)	A 1787
	<u>AESNI_AVX:</u> Key Gen, Key Ver; Curves P-256, P-384, P-521 Sig Gen, Sig Ver; Curves P-224, P-256, P-384, P-521 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512; (SHA-1 allowed for Sig Ver operations only)	A 1788
	<u>SHLD:</u> Key Gen, Key Ver; Curves P-256, P-384, P-521 Sig Gen, Sig Ver; Curves P-224, P-256, P-384, P-521 with hash sizes SHA-224, SHA-256, SHA-384, SHA-512; (SHA-1 allowed for Sig Ver operations only)	A 1789
Random Number Generation		
DRBG	**<u>SSSE3:</u>** **CTR_DRBG**: [Prediction Resistance Tested: Enabled and Not Enabled; Supports Reseed: (AES-128, AES-192, AES-256)] **HMAC_DRBG**: [Prediction Resistance Tested: Enabled and Not Enabled; (HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512)] **Hash_DRBG:** [Prediction Resistance Tested: Enabled and Not Enabled; (SHA-1, SHA-256, SHA-512)]	A 1785
	<u>Full Acceleration:</u> **CTR_DRBG**: [Prediction Resistance Tested: Enabled and Not Enabled; Supports Reseed: (AES-128, AES-192, AES-256)] **HMAC_DRBG**: [Prediction Resistance Tested: Enabled and Not Enabled; (HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512)] **Hash_DRBG:** [Prediction Resistance Tested: Enabled and Not Enabled; (SHA-1, SHA-	A 1786

Approved or Allowed Security Functions		Certificate
	256, SHA-512)]	
	No Acceleration: **CTR_DRBG**: [Prediction Resistance Tested: Enabled and Not Enabled; Supports Reseed: (AES-128, AES-192, AES-256)] **HMAC_DRBG**: [Prediction Resistance Tested: Enabled and Not Enabled; (HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512)] **Hash_DRBG:** [Prediction Resistance Tested: Enabled and Not Enabled; (SHA-1, SHA-256, SHA-512)]	A 1787
	AESNI_AVX: **CTR_DRBG**: [Prediction Resistance Tested: Enabled and Not Enabled; Supports Reseed: (AES-128, AES-192, AES-256)] **HMAC_DRBG**: [Prediction Resistance Tested: Enabled and Not Enabled; (HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512)] **Hash_DRBG:** [Prediction Resistance Tested: Enabled and Not Enabled; (SHA-1, SHA-256, SHA-512)]	A 1788
	SHLD: **HMAC_DRBG**: [Prediction Resistance Tested: Enabled and Not Enabled; (HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512)] **Hash_DRBG:** [Prediction Resistance Tested: Enabled and Not Enabled; (SHA-1, SHA-256, SHA-512)]	A 1789
RSA Key Transport Scheme (SP 800-56Br2)		
KTS-RSA	**SSSE3:** Function: partialVal; Modulo: 2048, 3072, 4096, 6144, 8192; Key Generation Methods: rsakpg1-basic; Scheme: KTS-OAEP-basic: KAS Role: initiator, responder; Key Transport Method Hash Algorithms: SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512	A 1785
	Full Acceleration: Function: partialVal; Modulo: 2048, 3072, 4096, 6144, 8192; Key Generation Methods: rsakpg1-basic; Scheme: KTS-OAEP-basic: KAS Role: initiator, responder; Key Transport Method Hash Algorithms: SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512	A 1786
	No Acceleration: Function: partialVal; Modulo: 2048, 3072, 4096, 6144, 8192; Key Generation Methods: rsakpg1-basic; Scheme: KTS-OAEP-basic: KAS Role: initiator, responder; Key Transport Method Hash Algorithms: SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512	A 1787
	AESNI_AVX: Function: partialVal; Modulo: 2048, 3072, 4096, 6144, 8192; Key Generation Methods: rsakpg1-basic; Scheme: KTS-OAEP-basic: KAS Role: initiator, responder;	A 1788

Approved or Allowed Security Functions		Certificate
	Key Transport Method Hash Algorithms: SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512	
	SHLD: Function: partialVal; Modulo: 2048, 3072, 4096, 6144, 8192; Key Generation Methods: rsakpg1-basic; Scheme: KTS-OAEP-basic: KAS Role: initiator, responder; Key Transport Method Hash Algorithms: SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512	A 1789
Password Based Key Derivation Function		
PBKDF	**SSSE3:** Hash Algorithm: SHA1, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256	A 1785
	Full Acceleration: Hash Algorithm: SHA1, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256	A 1786
	No Acceleration: Hash Algorithm: SHA1, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256	A 1787
	AESNI_AVX: Hash Algorithm: SHA1, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256	A 1788
	SHLD: Hash Algorithm: SHA1, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256	A 1789
Key Transport Scheme (KTS)		
KTS	**SSSE3:** AES-KW (D/E; Key Sizes 128 , 192, 256)	A 1785
	Full Acceleration: AES-KW (D/E; Key Sizes 128 , 192, 256)	A 1786
	No Acceleration: AES-KW (D/E; Key Sizes 128 , 192, 256)	A 1787
	AESNI_AVX: AES-KW (D/E; Key Sizes 128 , 192, 256)	A 1788
Entropy (NIST SP 800-90B)		
ENT (NP)	**NIST SP 800-90B**	N/A

Table 2: FIPS Approved Security Functions

3.5 Non-Approved but Allowed Security Functions

The following are considered non-Approved but allowed security functions:

Algorithm	Usage
RSA PKCS#1-v1.5 Key Wrapping with key sizes greater than 2048-bit up to 16384 bits	Key size between 2048 bits and 16384 bits or more; key establishment methodology provides between 112 and 256 bits of encryption strength.

Table 3: Non-Approved but Allowed Security Functions

3.6 Non-Approved Security Functions

The following services are non-Approved and use of these algorithms will put the module in the non-approved mode of operation implicitly. The services associated with these algorithms are specified in section 7.3:

Algorithm	Usage
AES-GCM	Authenticated Encrypt/Decrypt
ARC4	Encrypt/Decrypt
Blake2	Encrypt/Decrypt
Blowfish	Encrypt/Decrypt
Camellia	Encrypt/Decrypt
CAST5	Encrypt/Decrypt
ChaCha20	Encrypt/Decrypt
DES	Encrypt/Decrypt
IDEA	Encrypt/Decrypt
RC2	Encrypt/Decrypt
Salsa20	Encrypt/Decrypt
SEED	Encrypt/Decrypt
Serpent	Encrypt/Decrypt
Twofish	Encrypt/Decrypt
2-Key Triple-DES	Encryption
RSA	Key generation, signature generation, key wrapping with keys less than 2048 bits
	Signature verification with keys smaller than 1024 bits modulus size
DSA	Parameter verification, Parameter/Key generation/Signature generation and verification with keys not listed in Table 2
EdDSA	Key generation, signature generation and signature verification
GOST	28147 Encryption, R 34.11-94 Hashing, R 34.11.2012 (Stribog) Hashing
CSPRNG	Generating random numbers
Tiger	Hashing
MD4	Hashing
MD5	Hashing
Whirlpool	Hashing
RIPEMD 160	Hashing
HMAC with non-compliant key size	HMAC with less than 112-bit keys
El Gamal	Key generation/encryption/decryption/signature generation and verification
CRC 32	Cyclic redundancy check
OpenPGP Salted and Iterated/Salted	Password based KDF (RFC 4880)
SHA-1	Used for signature Generation

Table 4: Non-Approved Functions

4. Module Ports and Interfaces

As a software-only module, the module does not have physical ports. For the purpose of the FIPS 140-2 validation, the physical ports are interpreted to be the physical ports of the hardware platform on which it runs. The logical interfaces are the application program interface (API) through which applications request services.

Table below shows a mapping of FIPS 140 interfaces to logical ports:

FIPS 140 Interface	Module Interfaces
Data Input	API input parameters for data
Data Output	API output parameters for data
Control Input	API and API input parameter for control
Status Output	API return codes and error message

Table 5: Mapping of FIPS 140 Logical Interfaces to Logical Ports

The Data Input interface consists of the input parameters of the API functions. The Data Output interface consists of the output parameters of the API functions. The Control Input interface consists of the API function calls and the input parameters used to control the behavior of the module. The Status Output interface includes the return values of the API functions and status sent through log messages.

5. Physical Security

The Module is comprised of software only and thus does not claim any physical security.

6. Operational Environment

The module operates in a modifiable operational environment per FIPS 140-2 level 1 specifications.

6.1 Tested Environments

The Module was tested on the following environments with and without PAA i.e. AES-NI:

Operating Environment	Processor	Hardware
Oracle Linux 8.4 64-bit	Intel® Xeon® Platinum 8167M	Oracle Server X7-2C
Oracle Linux 8.4 64-bit	AMD EPYC™ 7551	Oracle Server E1-2C
Oracle Linux 8.4 64-bit	Ampere®Altra® Neoverse-N1	Oracle Server A1-2C

Table 6: Tested Operating Environments

6.2 Vendor Affirmed Environments

The following platforms have not been tested as part of the FIPS 140-2 level 1 certification however Oracle "vendor affirms" that these platforms are equivalent to the tested and validated platforms. Additionally, Oracle affirms that the module will function the same way and provide the same security services on any of the systems listed below.

Operating Environment	Hardware
Oracle Linux 8 64-bit	Oracle X Series Servers
Oracle Linux 8 64-bit	Oracle E Series Servers
Oracle Linux 8 64-bit	Oracle A Series Servers

Table 7: Vendor Affirmed Operating Environments

Note: CMVP makes no statement as to the correct operation of the module when so ported if the specific operational environment is not listed on the validation certificate.

6.3 Policy

The operating system is restricted to a single operator (concurrent operators are explicitly excluded). The application that request cryptographic services is the single user of the module, even when the application is serving multiple clients. In operational mode, the ptrace(2) system call, the debugger (gdb(1)), and strace(1) shall be not used.

7. Roles, Services and Authentication

This section defines the roles, services, and authentication mechanisms and methods with respect to the applicable FIPS 140-2 requirements.

7.1 Roles

The module supports the following roles:

- **User Role**: performs all services, except module installation.
- **Crypto Officer Role**: performs module installation.

The User and Crypto Officer roles are implicitly assumed by the entity accessing the module services.

7.2 FIPS Approved or allowed services

The table below provides a full description of FIPS Approved services provided by the module and lists the roles allowed to invoke each service. In the table below, the "U" represents a User Role, and "CO" denotes a Crypto Officer role.

U	CO	Service Name	Service Description	Keys and CSP(s)	Access Type(s)
X		Symmetric Encryption/ Decryption	Encrypts or decrypts a block of data using Triple-DES and/or AES	AES, Triple-DES Key	R, X
X		Get Key Length	cipher_get_keylen() function	N/A	-
X		Get Block Length	cipher_get_blocksize() function	N/A	-
X		Check Availability of Algorithm	cipher_get_blocksize() function	N/A	-
X		Hash	Secure Hash Algorithm (SHS) hashes a block of data.	N/A	-
X		Keyed Hash (HMAC)	authenticate data using HMAC-SHA	HMAC Key	R, X
X		Digital Signature Generation and Verification	Sign and verify operation	RSA, DSA, ECDSA Keys	R, X
X		Asymmetric Key Generation	Key pair generation for RSA, DSA or ECDSA	RSA, DSA, ECDSA Keys	R, W, X
X		Asymmetric Encryption/ Decryption	Encrypts or decrypts using Approved RSA key size	RSA Key pair	R, X
X		Random Number generation	Generate random numbers based on the SP 800-90A DRBG	Entropy input string, seed and internal state	R, W, X
X		Key Wrapping	AES-KW KTS, KTS-RSA, RSA PKCS#1-v1.5	AES Key, RSA Keys	R, X
X		Key Derivation	PBKDF	PBKDF password and PBKDF derived key	R, W, X

U	CO	Service Name	Service Description	Keys and CSP(s)	Access Type(s)
X		Self-Tests	Performs power-up self-test on demand	N/A	-
X		Zeroization	Zeroization performed by functions gcry_cipher_close(), gcry_sexp_release(), gcry_drbg_uninstantiate(),gcry_md_close()	All CSP's listed above.	R, W, X
X		Show Status	Show status of the module state	None	-
	X	Module Installation	Install and configure the module	None	-

R – Read, W – Write, X – Execute

Table 8: FIPS Approved or allowed Services and Descriptions

7.3 Non-FIPS Approved Services and Descriptions

The following table shows the services available in the non-FIPS mode:

U	CO	Service Name	Service Description	Keys	Access Type(s)
X		Symmetric Encryption/Decryption	Encrypts or decrypts using non-Approved algorithms listed in Table 4.	ARC4, Blake2, Blowfish, Camellia, CAST5, ChaCha20, DES, IDEA, RC2, Salsa20, GOST, SEED, Serpent, Twofish, 2-Key Triple-DES Key (encrypt only)	R, X
X		Authenticated Symmetric Encryption/Decryption	Encrypts or decrypts using AES-GCM	AES-GCM Key	R, X
X		Asymmetric Encryption/Decryption	Encrypts or decrypts using non-Approved RSA key size or using ElGamal	RSA key < 2048 bits, ElGamal keys	R, X
X		Digital Signature Generation and Verification	Sign or verify operations with non-Approved RSA or DSA key or signature generation using SHA-1 or ElGamal keys or EdDSA	RSA, DSA keys listed in Table 4, ElGamal keys, EdDSA	R, X
X		Asymmetric Key Generation	Generation of non-Approved RSA and DSA keys or ElGamal keys or EdDSA	RSA, DSA keys listed in Table 4, ElGamal keys, EdDSA	R, W, X
X		Hash	Hashing using non-Approved hash functions listed in Table 4.	N/A	-
X		Keyed Hash (HMAC)	HMAC-SHA Service with Key Sizes < 112 bits	HMAC Key	R, X
X		Cyclic Redundancy Code	CRC 32	N/A	-
X		Random Number Generation	Generating Random Numbers using CSPRNG	Entropy input string, seed and internal state	R, W, X
X		Password Based KDF	OpenPGP Salted and Iterated/Salted (RFC 4880)	Derived key	R, W, X

Table 9: Non-FIPS Approved Operator Services and Descriptions

7.4 Operator Authentication

The module is a Level 1 software-only cryptographic module and does not implement authentication. The role is implicitly assumed based on the service requested.

8. Cryptographic Key Management

Below is a list of the CSPs/keys details concerning storage, generation and zeroization:

CSP Name	Generation	Input/Output	Storage	Zeroization
AES Keys	N/A.	The Key is passed into the module via API input parameter	RAM	Automatically zeroized when freeing the cipher handler by calling gcry_free ()
Triple-DES Keys			RAM	
DSA Private Key	Generated using FIPS 186-4 and the random value used is generated using SP800- 90A DRBG	The Key is passed into and out of the module via API input parameter	RAM	Automatically zeroized when freeing the cipher handler by calling gcry_free ()
RSA Private Key			RAM	
ECDSA Private Key			RAM	Automatically zeroized when freeing the cipher handler by calling gcry_ecc_curve_free()
DRBG Entropy Input String	The seed data obtained from getrandom()	Obtained from CPU jitter source.	RAM	Automatically zeroized when freeing DRBG handler by calling gcry_free()
DRBG internal state (Seed, V, C, Key)	Generated during DRBG initialization as defined in SP 800-90A	N/A	RAM	
HMAC Key	N/A.	The Key is passed into the module via API input parameter.	RAM	Automatically zeroized when freeing the cipher handler by calling gcry_free()
PBKDF Derived Key	Generated during PBKDF	API output parameters are within the physical boundaries of the module. Output to calling application	RAM	Automatically zeroized when freeing the cipher handler by calling gcry_free()
PBKDF Password	N/A	Input password used for PBKDF function.	RAM	Automatically zeroized when freeing the cipher handler by calling gcry_free()

Table 10: CSP Table

8.1 Random Number Generation

The module provides an SP800-90A-compliant Deterministic Random Bit Generator (DRBG) for creation of key components of asymmetric keys, and random number generation.

The seeding (and automatic reseeding) of the DRBG is done with getrandom().

The module employs the Deterministic Random Bit Generator (DRBG) based on [SP 800-90A] for the random number generation. The DRBG supports the Hash_DRBG, HMAC_DRBG and CTR_DRBG mechanisms. The module performs the DRBG health tests as defined in section 11.3 of [SP 800-90A]. The module uses CPU jitter as a noise source provided by the operational environment which is within the module's physical boundary but outside of the module's logical boundary. The source is compliant with [SP 800-90B] and marked as ENT(NP) on the certificate. The entropy provided from the entropy source provides 230 bits of entropy and the module requires up to 256 bits of entropy. The caveat, "The module generates cryptographic keys whose strengths are modified by available entropy" applies.

8.2 Key Generation

The Key Generation methods implemented in the module for Approved services in FIPS mode is compliant with [SP 800-133].

For generating RSA and DSA keys the module implements asymmetric key generation services compliant with [FIPS 186-4] and [SP 800-90A]. A seed (i.e. the random value) used in asymmetric key generation is obtained from [SP 800-90A] DRBG using unmodified output from the Approved DRBG. The module does not offer a dedicated service for generating keys for symmetric algorithms or for HMAC. In accordance with FIPS 140-2 IG D.12, the cryptographic module performs Cryptographic Key Generation (CKG) for asymmetric keys as per SP 800-133 (vendor affirmed).

The module does not support manual key entry or intermediate key generation output. In addition, the module does not produce key output outside its physical boundary. The keys can be entered or output from the module in plaintext form via API parameters, to and from the calling application only.

8.3 Key/CSP Storage

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

8.4 Key/CSP Zeroization

The memory occupied by keys is allocated by regular memory allocation operating system calls. The application is responsible for calling the appropriate destruction functions listed in Table 10. The destruction functions overwrite the memory occupied by keys with "zeros" and deallocates the memory with the regular memory deallocation operating system call. In case of abnormal termination, or swap in/out of a physical memory page of a process, the keys in physical memory are overwritten by the Linux kernel before the physical memory is allocated to another process.

8.5 Key Establishment

The module provides RSA key wrapping using public key encryption and private key decryption primitives as allowed by [FIPS140-2_IG] D.9. RSA provides the following security strengths:

- RSA key wrapping with PKCS#1-v1.5 provides between 112 and 256 bits of encryption strength; Allowed per IG D.9

The module provides approved key transport methods compliant to SP 800-38F according to IG D.9. The key transport method is provided by:

- AES-KW

Therefore, the following caveats apply:

- KTS (AES Certs. #A1785, #A1786, #A1787 and #A1788; key establishment methodology provides between 128 and 256 bits of encryption strength)

The module provides approved key transport methods compliant to SP 800-56B according to IG D.9. The key transport method is provided by:

- KTS-RSA

Therefore, the following caveats apply:

- RSA key wrapping with OAEP; KTS-RSA (Certs. #A1785, #A1786, #A1787, #A1788 and #A1789; key establishment methodology provides between 112 and 200 bits of encryption strength).

The module also supports password-based key derivation (PBKDF). The implementation is compliant with option 1a of [SP 800-132]. Keys derived from passwords or passphrases using this method can only be used in storage applications.

9. Self-Tests

The Module perform self-tests to ensure the integrity of the Module, and the correctness of the cryptographic functionality at start up. In addition, some functions require continuous verification, such as the Random Number Generator. All of these tests are listed and described in this section.

9.1 Power-Up Self-Tests

The module performs power-on self-tests (POST) automatically during loading of the module by making use of default entry point (DEP) without any user intervention. While the power-up tests are in progress, services are not available and input or output is not possible: the module is single-threaded and will not return to the calling application until the self-tests are completed successfully. If any of the self-test fails module enters Error state. Only if all the self-tests are successful then the module is operational and cryptographic functionality is available for use.

The integrity of the module is verified by comparing the HMAC-SHA-256 value calculated at run time with the HMAC value stored in the hmac file that was computed at build time.

Algorithm	Test
AES (128, 192, 256)	KAT with CBC, CFB, CTR modes for encryption and decryption are tested separately
AES-CMAC	KAT
Triple-DES	KAT with CBC, CFB, CTR modes for encryption and decryption are tested separately
Triple-DES-CMAC	KAT
DSA	KAT for signature generation and verification using L=2048, N=224 and SHA-224
RSA	KAT for signature generation and verification using key size of 2048 and SHA-256 KTS-RSA self test is covered with signature generation and verification KAT as allowed by IG D.9
ECDSA	KAT of signature generation/verification using P-256 curve and SHA-256
SHA	KAT for (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512)
HMAC	KAT for (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)
PBKDF	KAT using SHA-256
DRBG	KAT for (Hash, HMAC, and CTR)
DRBG	Health test per section 11.3 of SP 800-90A DRBG
Module Integrity	HMAC-SHA-256 integrity test

Table 11: Power-On Self-Tests

9.2 On Demand Self-Tests

The module provides the Self-Test service to perform self-tests on demand. This service performs the same cryptographic algorithm tests executed during power-up. To invoke the on-demand self-tests, the user can make a call to the gcry_control (GCRYCTL_SELFTEST) function. During the execution of the on-demand self-tests, services are not available and no data output or input is possible.

9.3 Conditional Self-Tests

The following table provides the lists of the conditional self-tests. If any of the conditional test fails, the Module enters the Error state.

Algorithm	Test
RSA key generation	Pairwise Consistency Test: signature generation and verification, encryption and decryption
DSA key generation	Pairwise Consistency Test: signature generation and verification
ECDSA key generation	Pairwise Consistency Test: signature generation and verification

Table 12: Conditional Self-Tests

9.4 Error State/Fatal Error State

The Module enters Error state with error message, on failure of POST or conditional test. In Error state, all data output is inhibited and no cryptographic operation is allowed. The error can be recovered by calling gcry_control (GCRYCTL_SELFTEST) function that reruns the POST.

The module also has a Fatal Error state which is entered when random numbers are requested in error state or when requesting cipher operations on a deallocated cipher handle. When the Fatal error is reached, the abort() function is called which terminates the module and it is not available for use.

The module needs to be reloaded in order to recover from either error state or fatal error state.

10. Guidance

This section provides guidance for the Cryptographic Officer and the User to maintain proper use of the module per FIPS 140-2 requirements.

10.1 Crypto-Officer Guidance

The version of the RPM containing the validated module is stated in section 3.1 above. The RPM package of the Module can be downloaded from the Oracle Linux 8 "Security Validation (Update 4)" yum repository and installed by standard tools recommended for the installation of Oracle packages on an Oracle Linux system (for example, yum, RPM, and the RHN remote management tool). The integrity of the RPM is automatically verified during the installation of the Module and the Crypto Officer shall not install the RPM file if the Oracle Linux Yum Server indicates an integrity error. The RPM files listed in section 3 are signed by Oracle and during installation; Yum performs signature verification which ensures as secure delivery of the cryptographic module. If the RPM packages are downloaded manually, then the CO should run 'rpm –K <rpm-file-name>' command after importing the builder's GPG key to verify the package signature. In addition, the CO can also verify the hash of the RPM package to confirm a proper download.

Recommended method

The system-wide cryptographic policies package (crypto-policies) contains a tool that completes the installation of cryptographic modules and enables self-checks in accordance with the requirements of Federal Information Processing Standard (FIPS) Publication 140-2. We call this step "FIPS enablement". The tool named fips-mode-setup installs and enables or disables all the validated FIPS modules and it is the recommended method to install and configure an Oracle Linux 8 system.

1. Install RPM file e.g for x86_64 use dnf command:
    # dnf install libgcrypt-1.8.5-4.0.1.el8.x86_64.rpm
2. Switch the system to FIPS enablement in Oracle Linux 8:
    # fips-mode-setup --enable
    Setting system policy to FIPS
    FIPS mode will be enabled.
    Reboot the system for the setting to take effect.
3. Restart your system:
    # reboot
4. After the restart, you can check the current state:
    # fips-mode-setup --check
    FIPS mode is enabled.

Note: As a side effect of the enablement procedure the fips-mode-enable tool also changes the system wide cryptographic policy level to a level named "FIPS", this level helps applications by changing configuration defaults to approved algorithms.

10.1.1 AES Hardware Acceleration and Manual Method

According to the libgcrypt FIPS 140-2 Security Policy, the libgcrypt module supports the AES-NI Intel processor instruction and ARM AES optimizations set as an approved cipher. Both architecture optimizations are used by the Module.

In case you configured a full disk encryption using AES, you may use the aforementioned optimizations for a higher performance compared to the software-only implementation.

Verify that your processor offers AES hardware acceleration by calling the following command:

    cat /proc/cpuinfo | grep aes

If the command returns a list of properties, including the “aes” string, your CPU provides the AES hardware acceleration. If the command returns nothing, AES hardware acceleration is not supported.

The recommended method automatically performs all the necessary steps. The following steps can be done manually but are not recommended and are not required if the systems has been installed with the fips-mode-setup tool:

- Create a file named /etc/system-fips, the contents of this file are never checked
- Ensure to invoke the command ‘fips-finish-install --complete’ on the installed system
- Ensure that the kernel boot line is configured with the fips=1 parameter set
- Reboot the system

NOTE: If /boot or /boot/efi resides on a separate partition, the kernel parameter boot=<boot partition> must be supplied. The partition can be identified with the command "df | grep boot". For example:

    $ df | grep boot

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
/dev/sda1	233191	30454	190296	14%	/boot

The partition of the /boot file system is located on /dev/sda1 in this example.

Therefore the parameter boot=/dev/sda1 needs to be appended to the kernel command line in addition to the parameter fips=1.

10.2 User Guidance

Applications using libgcrypt need to call gcry_control(GCRYCTL_INITIALIZATION_FINISHED, 0) after initialization is done: that ensures that the DRBG is properly seeded, among others. gcry_control(GCRYCTL_TERM_SECMEM) needs to be called before the process is terminated. The function gcry_set_allocation_handler() may not be used.

The user must not call malloc/free to create/release space for keys, let libgcrypt manage space for keys, which will ensure that the key memory is overwritten before it is released. See the documentation file doc/gcrypt.texi within the source code tree for complete instructions for use.

The information pages are included within the developer package. The user can find the documentation at the following location after having installed the developer package:

    /usr/share/info/gcrypt.info-1.gz
    /usr/share/info/gcrypt.info-2.gz
    /usr/share/info/gcrypt.info.gz

10.2.1 Triple-DES Keys

Data encryption using the same three-key Triple-DES key shall not exceed 2^{16} Triple-DES (64-bit) blocks, in accordance to [SP 800-67] and IG A.13 in [FIPS 140-2-IG]. The user of the module is responsible for ensuring the module's compliance with this requirement.

10.2.2 AES-XTS Guidance

The length of a single data unit encrypted or decrypted with the XTS-AES shall not exceed 2^{20} AES blocks that is 16MB of data per AES-XTS instance. An XTS instance is defined in section 4 of SP 800-38E.

The module implements a check to ensure that the two AES keys used in XTS-AES algorithm are not identical.

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

10.2.3 Key Derivation Using SP 800-132 PBKDF

The module provides password-based key derivation (PBKDF), compliant with SP 800-132. The module supports option 1a from section 5.4 of [SP 800-132], in which the Master Key (MK) or a segment of it is used directly as the Data Protection Key (DPK).

In accordance to [SP 800-132], the following requirements shall be met.

- Derived keys shall only be used in storage applications. The Master Key (MK) shall not be used for other purposes. The length of the MK or DPK shall be of 112 bits or more.
- A portion of the salt, with a length of at least 128 bits, shall be generated randomly using the SP800-90A DRBG,
- The iteration count shall be selected as large as possible, as long as the time required to generate the key using the entered password is acceptable for the users. The minimum value shall be 1000.
- Passwords or passphrases, used as an input for the PBKDF, shall not be used as cryptographic keys.
- The length of the password or passphrase shall be of at least 20 characters, and shall consist of lower-case, upper-case and numeric characters. The probability of guessing the value is estimated to be $1/62^{20}$ = 10^{-36}, which is less than 2^{-112}.

The calling application shall also observe the rest of the requirements and recommendations specified in [SP800-132].

11. Mitigation of Other Attacks

libgcrypt uses a blinding technique for RSA decryption to mitigate real world timing attacks over a network: Instead of using the RSA decryption directly, a blinded value (y = x·re mod n) is decrypted and the unblinded value (x' = y'·r-1 mod n) returned. The blinding value "r" is a random value with the size of the modulus "n" and generated with `GCRY_WEAK_RANDOM' random level.

Weak Triple-DES keys are detected as follows:

In DES there are 64 known keys which are weak because they produce only one, two, or four different subkeys in the subkey scheduling process. The keys in this table have all their parity bits cleared.

```
static byte weak_keys[64][8] =
{
      { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }, /*w weak keys*/
      { 0x00, 0x00, 0x1e, 0x1e, 0x00, 0x00, 0x0e, 0x0e },
      { 0x00, 0x00, 0xe0, 0xe0, 0x00, 0x00, 0xf0, 0xf0 },
      { 0x00, 0x00, 0xfe, 0xfe, 0x00, 0x00, 0xfe, 0xfe },
      { 0x00, 0x1e, 0x00, 0x1e, 0x00, 0x0e, 0x00, 0x0e }, /*sw semi-weak keys*/
      { 0x00, 0x1e, 0x1e, 0x00, 0x00, 0x0e, 0x0e, 0x00 },
      { 0x00, 0x1e, 0xe0, 0xfe, 0x00, 0x0e, 0xf0, 0xfe },
      { 0x00, 0x1e, 0xfe, 0xe0, 0x00, 0x0e, 0xfe, 0xf0 },
      { 0x00, 0xe0, 0x00, 0xe0, 0x00, 0xf0, 0x00, 0xf0 }, /*sw*/
      { 0x00, 0xe0, 0x1e, 0xfe, 0x00, 0xf0, 0x0e, 0xfe },
      { 0x00, 0xe0, 0xe0, 0x00, 0x00, 0xf0, 0xf0, 0x00 },
      { 0x00, 0xe0, 0xfe, 0x1e, 0x00, 0xf0, 0xfe, 0x0e },
      { 0x00, 0xfe, 0x00, 0xfe, 0x00, 0xfe, 0x00, 0xfe }, /*sw*/
      { 0x00, 0xfe, 0x1e, 0xe0, 0x00, 0xfe, 0x0e, 0xf0 },
      { 0x00, 0xfe, 0xe0, 0x1e, 0x00, 0xfe, 0xf0, 0x0e },
      { 0x00, 0xfe, 0xfe, 0x00, 0x00, 0xfe, 0xfe, 0x00 },
      { 0x1e, 0x00, 0x00, 0x1e, 0x0e, 0x00, 0x00, 0x0e },
      { 0x1e, 0x00, 0x1e, 0x00, 0x0e, 0x00, 0x0e, 0x00 }, /*sw*/
      { 0x1e, 0x00, 0xe0, 0xfe, 0x0e, 0x00, 0xf0, 0xfe },
      { 0x1e, 0x00, 0xfe, 0xe0, 0x0e, 0x00, 0xfe, 0xf0 },
      { 0x1e, 0x1e, 0x00, 0x00, 0x0e, 0x0e, 0x00, 0x00 },
      { 0x1e, 0x1e, 0x1e, 0x1e, 0x0e, 0x0e, 0x0e, 0x0e }, /*w*/
      { 0x1e, 0x1e, 0xe0, 0xe0, 0x0e, 0x0e, 0xf0, 0xf0 },
      { 0x1e, 0x1e, 0xfe, 0xfe, 0x0e, 0x0e, 0xfe, 0xfe },
      { 0x1e, 0xe0, 0x00, 0xfe, 0x0e, 0xf0, 0x00, 0xfe },
      { 0x1e, 0xe0, 0x1e, 0xe0, 0x0e, 0xf0, 0x0e, 0xf0 }, /*sw*/
      { 0x1e, 0xe0, 0xe0, 0x1e, 0x0e, 0xf0, 0xf0, 0x0e },
      { 0x1e, 0xe0, 0xfe, 0x00, 0x0e, 0xf0, 0xfe, 0x00 },
      { 0x1e, 0xfe, 0x00, 0xe0, 0x0e, 0xfe, 0x00, 0xf0 },
      { 0x1e, 0xfe, 0x1e, 0xfe, 0x0e, 0xfe, 0x0e, 0xfe }, /*sw*/
      { 0x1e, 0xfe, 0xe0, 0x00, 0x0e, 0xfe, 0xf0, 0x00 },
      { 0x1e, 0xfe, 0xfe, 0x1e, 0x0e, 0xfe, 0xfe, 0x0e },
      { 0xe0, 0x00, 0x00, 0xe0, 0xf0, 0x00, 0x00, 0xf0 },
      { 0xe0, 0x00, 0x1e, 0xfe, 0xf0, 0x00, 0x0e, 0xfe },
```

```
{ 0xe0, 0x00, 0xe0, 0x00, 0xf0, 0x00, 0xf0, 0x00 }, /*sw*/
{ 0xe0, 0x00, 0xfe, 0x1e, 0xf0, 0x00, 0xfe, 0x0e },
{ 0xe0, 0x1e, 0x00, 0xfe, 0xf0, 0x0e, 0x00, 0xfe },
{ 0xe0, 0x1e, 0x1e, 0xe0, 0xf0, 0x0e, 0x0e, 0xf0 },
{ 0xe0, 0x1e, 0xe0, 0x1e, 0xf0, 0x0e, 0xf0, 0x0e }, /*sw*/
{ 0xe0, 0x1e, 0xfe, 0x00, 0xf0, 0x0e, 0xfe, 0x00 },
{ 0xe0, 0xe0, 0x00, 0x00, 0xf0, 0xf0, 0x00, 0x00 },
{ 0xe0, 0xe0, 0x1e, 0x1e, 0xf0, 0xf0, 0x0e, 0x0e },
{ 0xe0, 0xe0, 0xe0, 0xe0, 0xf0, 0xf0, 0xf0, 0xf0 }, /*w*/
{ 0xe0, 0xe0, 0xfe, 0xfe, 0xf0, 0xf0, 0xfe, 0xfe },
{ 0xe0, 0xfe, 0x00, 0x1e, 0xf0, 0xfe, 0x00, 0x0e },
{ 0xe0, 0xfe, 0x1e, 0x00, 0xf0, 0xfe, 0x0e, 0x00 },
{ 0xe0, 0xfe, 0xe0, 0xfe, 0xf0, 0xfe, 0xf0, 0xfe }, /*sw*/
{ 0xe0, 0xfe, 0xfe, 0xe0, 0xf0, 0xfe, 0xfe, 0xf0 },
{ 0xfe, 0x00, 0x00, 0xfe, 0xfe, 0x00, 0x00, 0xfe },
{ 0xfe, 0x00, 0x1e, 0xe0, 0xfe, 0x00, 0x0e, 0xf0 },
{ 0xfe, 0x00, 0xe0, 0x1e, 0xfe, 0x00, 0xf0, 0x0e },
{ 0xfe, 0x00, 0xfe, 0x00, 0xfe, 0x00, 0xfe, 0x00 }, /*sw*/
{ 0xfe, 0x1e, 0x00, 0xe0, 0xfe, 0x0e, 0x00, 0xf0 },
{ 0xfe, 0x1e, 0x1e, 0xfe, 0xfe, 0x0e, 0x0e, 0xfe },
{ 0xfe, 0x1e, 0xe0, 0x00, 0xfe, 0x0e, 0xf0, 0x00 },
{ 0xfe, 0x1e, 0xfe, 0x1e, 0xfe, 0x0e, 0xfe, 0x0e }, /*sw*/
{ 0xfe, 0xe0, 0x00, 0x1e, 0xfe, 0xf0, 0x00, 0x0e },
{ 0xfe, 0xe0, 0x1e, 0x00, 0xfe, 0xf0, 0x0e, 0x00 },
{ 0xfe, 0xe0, 0xe0, 0xfe, 0xfe, 0xf0, 0xf0, 0xfe },
{ 0xfe, 0xe0, 0xfe, 0xe0, 0xfe, 0xf0, 0xfe, 0xf0 }, /*sw*/
{ 0xfe, 0xfe, 0x00, 0x00, 0xfe, 0xfe, 0x00, 0x00 },
{ 0xfe, 0xfe, 0x1e, 0x1e, 0xfe, 0xfe, 0x0e, 0x0e },
{ 0xfe, 0xfe, 0xe0, 0xe0, 0xfe, 0xfe, 0xf0, 0xf0 },
{ 0xfe, 0xfe, 0xfe, 0xfe, 0xfe, 0xfe, 0xfe, 0xfe } /*w*/ };
```

Acronyms, Terms and Abbreviations

Term	Definition
AES	Advanced Encryption Standard
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CFB	Cipher Feedback
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CCCS	Canadian Centre for Cyber Security
CSP	Critical Security Parameter
CTR	Counter Mode
DES	Data Encryption Standard
DSA	Digital Signature Algorithm
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standard
HMAC	(Keyed) Hash Message Authentication Code
KAT	Known Answer Test
MAC	Message Authentication Code
KDF	Key Derivation Function
NIST	National Institute of Standards and Technology
OFB	Output Feedback Mode
OS	Operating System
PBKDF	Password Based Key Derivation Function
PCT	Pairwise Consistency Test
PKCS	Public Key Cryptographic Standard
POST	Power On Self-Test
PR	Prediction Resistance
PSS	Probabilistic Signature Scheme
PUB	Publication
RNG	Random Number Generator
RSA	Rivest, Shamir, Adleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard

Table 13: Acronyms

References

Reference	Full Specification Name
[FIPS 140-2]	Security Requirements for Cryptographic modules, May 25, 2001
[FIPS 180-4]	Secure Hash Standard (SHS) March 2012 http://csrc.nist.gov/publications/fips/fips180-4/fips 180-4.pdf
[FIPS 186-4]	Digital Signature Standard (DSS) July 2013 http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
[FIPS 197]	Advanced Encryption Standard November 2001 http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
[FIPS 198-1]	The Keyed Hash Message Authentication Code (HMAC) July 2008 http://csrc.nist.gov/publications/fips/fips198 1/FIPS-198 1_final.pdf
[PKCS #1]	Public Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1 February 2003 http://www.ietf.org/rfc/rfc3447.txt
[RFC 5649]	Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm September 2009 http://www.ietf.org/rfc/rfc5649.txt
[SP 800-38A]	NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques December 2001 http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf
[SP 800-38B]	NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication June 2016 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38b.pdf
[SP 800-38C]	NIST Special Publication 800-38C - Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality May 2004 http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf
[SP 800-38E]	Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices
[SP 800-38F]	Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping
[SP 800-56CR2]	Recommendation for Key-Derivation Methods in Key-Establishment Schemes
[SP 800- 67R1]	Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher
[SP 800-89]	Recommendation for Obtaining Assurances for Digital Signature Applications
[SP 800-90Ar1]	Recommendation for Random Number Generation Using Deterministic Random Bit Generators
[SP 800-90B]	Recommendation for the Entropy Sources Used for Random Bit Generation
[SP 800- 131Ar2]	Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths

Table 14: References