SUSE Linux Enterprise Kernel Crypto API Cryptographic Module

Software Module Version 2.1

FIPS 140-2 Non-Proprietary Security Policy

Document Version: 1.2

Document Date: 2021-07-23

Prepared by:

atsec information security corporation

9130 Jollyville Road, Suite 260

Austin, TX 78759

www.atsec.com

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Table of contents

©2021 SUSE, LLC., atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

©2021 SUSE, LLC., atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

List of Tables

List of Figures

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

1 Introduction

1.1 Purpose

This document is the non-proprietary security policy for the SUSE Linux Enterprise Kernel Crypto API Cryptographic Module version 2.1 (also referred as “Kernel Crypto API module” or “module” throughout this document). It contains the security rules under which the module must operate and describes how this module meets the requirements as specified in FIPS 140-2 (Federal Information Processing Standards Publication 140-2) for a security level 1 module.

FIPS 140-2 details the requirements of the Governments of the U.S. and Canada for cryptographic modules, aimed at the objective of protecting sensitive but unclassified information. For more information on the FIPS 140-2 standard and validation program please refer to the NIST website at http://csrc.nist.gov/.

1.2 External Resources and References

The SUSE website (www.suse.com) contains information about the module.

The Cryptographic Module Validation Program website (http://csrc.nist.gov/groups/STM/cmvp/) contains links to the FIPS 140-2 certificate and SUSE contact information.

1.3 How this Security Policy was Prepared

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

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

2 Cryptographic Module Specification

2.1 Module Overview

The SUSE Linux Enterprise Kernel Crypto API Cryptographic Module is a software cryptographic module that provides general-purpose cryptographic services. For the purpose of the FIPS 140-2 validation, the module is a software-only, multi-chip standalone cryptographic module validated at overall security level 1.

Table 1 shows the security level claimed for each of the eleven sections that comprise the FIPS 140-2 standard.

Table 1: Security levels.

FIPS 140-2 Section		Security Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1
3	Roles, Services and Authentication	1
4	Finite State Model	1
5	Physical Security	N/A
6	Operational Environment	1
7	Cryptographic Key Management	1
8	EMI/EMC	1
9	Self Tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A
Overall		**1**

Table 2 lists the software components of the cryptographic module, which define the module's logical boundary. In the table, if the filename differs between the tested platforms (Table 3), then the applicable platform will be indicated with the Operating System version (SP4 or SP5) with the filename in its own row.

The Linux kernel version, which is the output of $(uname -r), is:

- SP5 platform: 4.12.14-122.37-default.
- SP4 platform: 4.12.14-95.60-default.

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Table 2: Cryptographic module components.

Description	OS Version	Component
Static kernel binary	SP4	/boot/vmlinuz-4.12.14-95.60-default
	SP5	/boot/vmlinuz-4.12.14-122.37-default
Integrity check HMAC file for Linux kernel static binary	SP4	/boot/.vmlinuz-4.12.14-95.60-default.hmac
	SP5	/boot/.vmlinuz-4.12.14-122.37-default.hmac
Cryptographic kernel object files	SP4	/lib/modules/4.12.14-95.60-default/kernel/crypto/*.ko /lib/modules/4.12.14-95.60-default/kernel/arch/x86/crypto/*.ko (x86 platform) /lib/modules/4.12.14-95.60-default/kernel/arch/s390/crypto/*.ko (z13 platform)
	SP5	/lib/modules/4.12.14-122.37-default/kernel/crypto/*.ko /lib/modules/4.12.14-122.37-default/kernel/arch/x86/crypto/*.ko (x86 platform) /lib/modules/4.12.14-122.37-default/kernel/arch/s390/crypto/*.ko (z13 platform)
Integrity test utility	SP4 and SP5	/usr/lib64/libkcapi/fipscheck
Integrity check HMAC file for integrity test utility	SP4 and SP5	/usr/lib64/libkcapi/.fipscheck.hmac

The software block diagram (Figure 1) shows the logical boundary of the module and its interfaces with the operational environment.

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

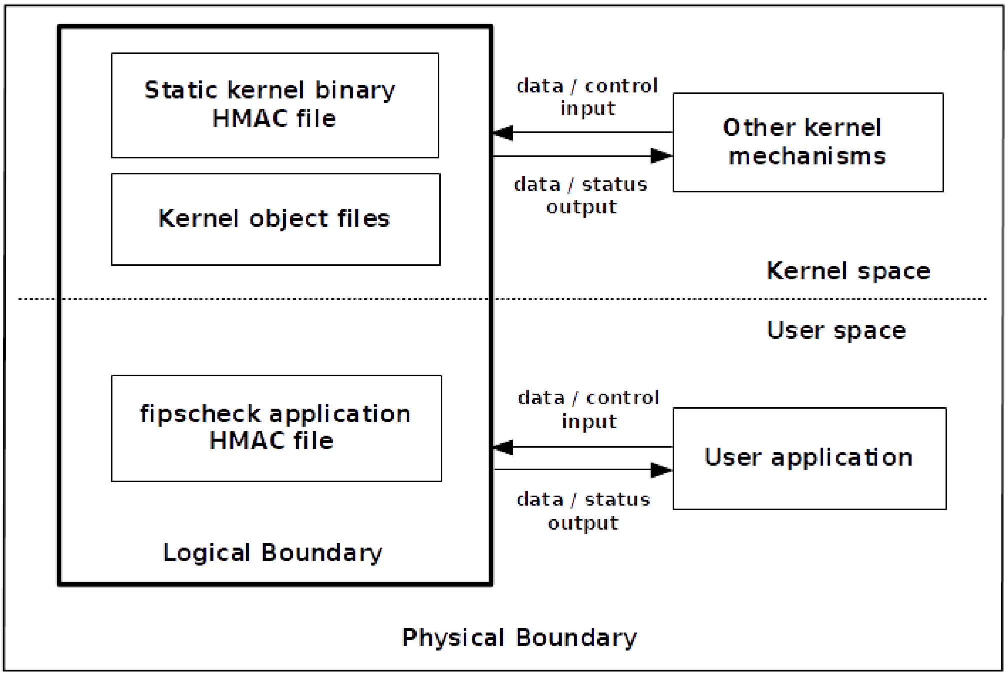

Figure 1: Software Block Diagram.

The module is aimed to run on a general purpose computer (GPC). Table 3 shows the platforms on which the module has been tested. Note the differences in the “Test Configuration” column, namely some platforms use the SP4 version of the Operating System, others use the SP5 version of the Operating System. The versions of each component within the Operational Environment is given in Table 2.

Table 3: Tested platforms.

Platform	Processor	Operating System
FUJITSU Server PRIMERGY RX4770 M5	Intel Cascade Lake Xeon Platinum 8268	SUSE Linux Enterprise Server 12 SP4 with and without PAA (AES-NI)
FUJITSU Server PRIMERGY RX4770 M5	Intel Cascade Lake Xeon Platinum 8268	SUSE Linux Enterprise Server 12 SP5 with and without PAA (AES-NI)
IBM System Z z13	z13	SUSE Linux Enterprise Server 12 SP4 with and without PAI (CPACF)
IBM System Z z13	z13	SUSE Linux Enterprise Server 12 SP5 with and without PAI (CPACF)

Note: Per FIPS 140-2 IG G.5, the Cryptographic Module Validation Program (CMVP) makes no statement as to the correct operation of the module or the security strengths of the generated keys when this module is ported and executed in an operational environment not listed on the validation certificate.

The physical boundary of the module is the surface of the case (or physical enclosure) of the tested platform. Figure 2 shows the hardware block diagram including major hardware components of a GPC representing the tested platforms.

©2021 SUSE, LLC., atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

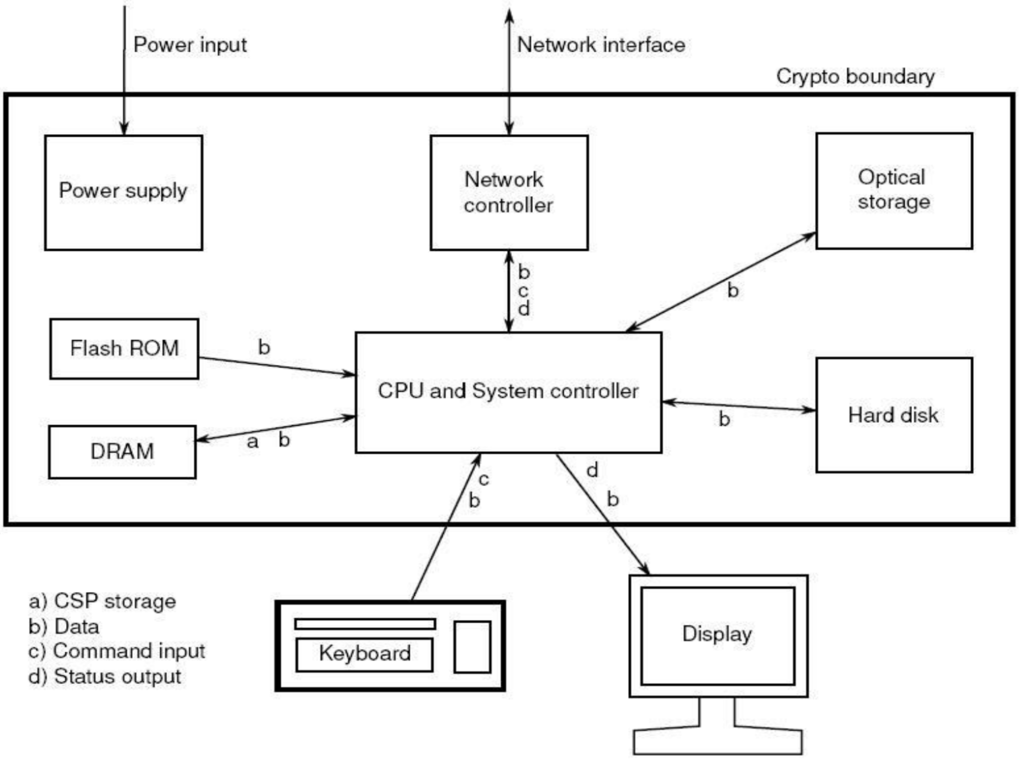

Figure 2: Hardware Block Diagram.

2.2 Modes of Operation

The module supports two modes of operation:

- FIPS mode (the Approved mode of operation): only approved or allowed security functions with sufficient security strength can be used.
- non-FIPS mode (the non-Approved mode of operation): only non-approved security functions can be used.

The module enters FIPS mode after power-up tests succeed. Once the module is operational, the mode of operation is implicitly assumed depending on the security function invoked and the security strength of the cryptographic keys.

Critical security parameters (CSPs) used or stored (in RAM) in FIPS mode are not shared with the non-FIPS mode, and vice versa. The Operational Environment provides context and memory separation between processes and users of the module.

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

3 Cryptographic 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 4 summarizes the logical interfaces.

Table 4: Ports and interfaces.

Logical Interface	Description
Data Input	API input parameters from kernel system calls, AF_ALG type socket.
Data Output	API output parameters from kernel system calls, AF_ALG type socket.
Control Input	API function calls, API input parameters from kernel system calls, AF_ALG type socket, kernel command line.
Status Output	API return values, AF_ALG type socket, kernel logs.

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

4 Roles, Services and Authentication

4.1 Roles

The module meets all FIPS 140-2 level 1 requirements for Roles and Services, implementing both User and Crypto Officer (CO) roles. The module does not allow concurrent operators. The roles execute the services as listed below.

- User role: performs all services, except module installation and configuration.
- Crypto Officer role: performs module installation and configuration.

The User and Crypto Officer roles are implicitly assumed by the entity accessing the services implemented by the module depending on the service requested. No authentication is required by the module for the assumption of a role.

4.2 Services

The module provides services to the operators that assume one of the available roles. All services are shown in Table 5 and Table 6.

For each service, the tables list the associated cryptographic algorithm(s), the role that performs the service, the cryptographic keys and other CSPs involved, and their access type(s) from the point of view of the service. The details of the approved cryptographic algorithms including the CAVP certificate numbers can be found in Table 7.

The following convention is used to specify access rights from a service to a CSP.

- *Generate (G)*: the service generates or drives the CSP.
- *Read (R):* the service outputs the CSP, so that the CSP can be read by the user.
- *Write (W)*: the service can update, import, or write (not generate) the CSP.
- *Execute (X)*: the service uses the CSP in performing a cryptographic operation.
- *Zeroize (Z)*: the service zeroizes the existing CSP.
- *N/A*: the service does not access any CSP during its operation.

4.2.1 Services in the Approved Mode

Table 5 lists the services available in FIPS mode.

Table 5: Services in FIPS mode of operation.

Service	Algorithm	Role	Keys/CSPs	Access
Symmetric encryption and decryption	AES	User	AES key	Execute
	Three-key Triple-DES	User	Triple-DES key	Execute
Random number generation	DRBG	User	Entropy input string, Internal state	Generate, Write, Execute
Message digest	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512	User	None	N/A
	HMAC	User	HMAC key	Execute

©2021 SUSE, LLC., atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Service	Algorithm	Role	Keys/CSPs	Access
Message authentication code (MAC)	CMAC-AES	User	AES key	Execute
	CMAC-Triple-DES	User	Triple-DES key	Execute
Encrypt-then-MAC operation	AES-[CBC, CTR], HMAC-[SHA-1, SHA2]	User	AES key, HMAC key	Execute
	Triple-DES-CBC, HMAC-[SHA-1, SHA2]		Triple-DES key, HMAC key	Execute
Key wrapping (asymmetric)	RSA encrypt/decrypt primitives	User	RSA public/private keys	Execute
Key wrapping (symmetric)	AES-KW, AES-CCM, AES-GCM,	User	AES key	Execute
Key wrapping combination (symmetric)	Combination AES-[CBC, CTR], Triple-DES-CBC and HMAC	User	AES key, Triple-DES key, HMAC key	Execute
Digital signature verification	Verify signature operation using RSA PKCS#1v1.5	User	RSA public/private keys	Execute
Zeroization	N/A	User	All CSPs	Zeroize
Self-tests	See Section 9	User	None	N/A
Other services				
Error detection code	crc32c[1], crct10dif	User	None	N/A
Data compression	deflate, lz4, lz4hc, lzo, zlib, 842	User	None	N/A
Memory copy operation	ecb(cipher_null)	User	None	N/A
Show status	N/A	User	None	N/A
Module installation and configuration	N/A	Crypto Officer	None	N/A

4.2.2 Services in the Non-Approved Mode

Table 6 lists the services only available in non-FIPS mode of operation. The details of the non-approved cryptographic algorithms available in non-FIPS mode can be found in Table 9.

Table 6: Services in non-FIPS mode of operation.

Service	Algorithm	Role	Keys	Access
	AES-XTS with 192-bit key size.	User	AES key	Execute

1 This algorithm does not provide any cryptographic attribute, i.e., its purpose in the module is not security relevant.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Service	Algorithm	Role	Keys	Access
Symmetric encryption and decryption	Generic GCM with external IV, RFC4106 GCM with external IV.	User	AES key	Execute
	Anubis block cipher Blowfish block cipher Camellia block cipher CAST5, CAST6 Serpent block cipher Twofish block cipher ARC4 stream cipher Salsa20 stream cipher ChaCha20 stream cipher	User	AES, anubis, blowfish, camellia, cast5, cast6, serpent, twofish, arc4, salsa20, chacha20 keys	Execute
	z13 platform: AES and Triple-DES algorithms with “generic” implementation	User	AES, Triple-DES keys	Execute
Message digest	GHASH outside the GCM context. MD4, MD5 RIPEMD Tiger Hashing Whirlpool z13 platform: SHA3 and SHA2 algorithms with “generic” implementation	User	None	N/A
Message authentication code (MAC)	HMAC with key smaller than 112-bit keys. z13 platform: HMAC with “generic” implementation	User	HMAC key	Execute
RSA signature generation	RSA sign primitive operation.	User	RSA public/private keys	Execute
RSA signature verification	RSA verify primitive operation with keys smaller than 2048 bits.	User	RSA public/private keys	Execute
RSA key wrapping	RSA keys smaller than 2048 bits.	User	RSA public/private keys	Execute
EC key generation	EC key generation.	User	EC public/private keys	Generate, Execute, Write

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Service	Algorithm	Role	Keys	Access
Shared Secret Computation	Diffie-Hellman EC Diffie-Hellman	User	Diffie-Hellman public/private keys	Execute
			EC Diffie-Hellman public/private keys	Execute
			Shared Secret	Generate, Execute, Write

4.3 Operator Authentication

The module does not implement user authentication. The role of the user is implicitly assumed based on the service requested.

4.4 Algorithms

The module provides multiple implementations of algorithms. Different implementations can be invoked by using the unique algorithm driver names.

Among the implementations, the module supports generic C for all algorithms; generic assembler for AES, Triple-DES ciphers, and block modes; AES-NI for AES; CLMUL for GHASH within the GCM block mode; AVX, AVX2, SSSE3 for SHA algorithms; constant-time C implementation for ciphers, and CPACF instructions for z13.

Appendix A brings a list of the names contained in the CAVP algorithm certificates that refer to the specific algorithm implementations, along with their description.

4.4.1 Approved Algorithms

Table 7 lists the approved algorithms, the CAVP certificates, and other associated information of the cryptographic implementations available in the FIPS mode.

Note: the module does not implement all the algorithms, as FIPS approved algorithms, for which the CAVP certificates were issued.

Table 7: Approved Cryptographic Algorithms.

Algorithm	Mode/Method	Key Lengths, Curves or Moduli (bits)	Use	Standard	CAVP Certs.
AES	ECB	128, 192, 256	Data Encryption and Decryption	FIPS197, SP800-38A	A694 (RFC4106EIV_CPACF_ASM) A695 (RFC4106IIV_CPACF_ASM) A696 (CPACF_ASM) A698 (RFC4106EIV_CTI_C) A699 (RFC4106IIV_CTI_C) A702 (RFC4106EIV_C_C) A703 (RFC4106IIV_C_C) A706 (RFC4106EIV_CPACF_C) A707 (RFC4106IIV_CPACF_C) A709 (RFC4106EIV_AESNI_ASM) A710 (RFC4106IIV_AESNI_ASM)

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm	Mode/Method	Key Lengths, Curves or Moduli (bits)	Use	Standard	CAVP Certs.
					A714 (RFC4106EIV_X86ASM_C) A715 (RFC4106IIV_X86ASM_C) A719 (RFC4106EIV_AESNI_C) A720 (RFC4106IIV_AESNI_C)
	ECB, CBC, CTR	128, 192, 256	Data Encryption and Decryption	FIPS197, SP800-38A	A692 (C_C) A696 (CPACF_ASM) A701 (CTI_C) A708 (CPACF_C) A711 (AESNI_ASM) A716 (X86ASM_C) A721 (AESNI_C)
	KW	128, 192, 256	Key Wrapping	SP800-38F	A691 (KW_C_C) A700 (KW_CTI_C) A705 (KW_CPACF_C) A713 (KW_X86ASM_C) A718 (KW_AESNI_C)
	XTS	128, 256	Data Encryption and Decryption for Data Storage	SP800-38E	A692 (C_C) A696 (CPACF_ASM) A701 (CTI_C) A708 (CPACF_C) A711 (AESNI_ASM) A716 (X86ASM_C) A721 (AESNI_C)
	GCM with external IV	128, 192, 256	Data Decryption[2]	SP800-38D	A692 (C_C) A696 (CPACF_ASM) A701 (CTI_C) A708 (CPACF_C) A711 (AESNI_ASM) A716 (X86ASM_C) A721 (AESNI_C)
	GCM with internal IV (RFC4106)	128, 192, 256	Data Encryption	SP800-38D RFC4106	A695 (RFC4106IIV_CPACF_ASM) A699 (RFC4106IIV_CTI_C) A703 (RFC4106IIV_C_C) A707 (RFC4106IIV_CPACF_C) A710 (RFC4106IIV_AESNI_ASM) A715 (RFC4106IIV_X86ASM_C) A720 (RFC4106IIV_AESNI_C)
	GCM with external IV (RFC4106)	128, 192, 256	Data Decryption	SP800-38D RFC4106	A694 (RFC4106EIV_CPACF_ASM) A698 (RFC4106EIV_CTI_C) A702 (RFC4106EIV_C_C) A706 (RFC4106EIV_CPACF_C)

2 This algorithm was tested for encryption and decryption, however the encryption operation is not approved in the FIPS mode due to the use of external IV.

©2021 SUSE, LLC., atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm	Mode/Method	Key Lengths, Curves or Moduli (bits)	Use	Standard	CAVP Certs.
					A709 (RFC4106EIV_AESNI_ASM) A714 (RFC4106EIV_X86ASM_C) A719 (RFC4106EIV_AESNI_C)
	CMAC	128, 192, 256	MAC Generation and Verification	SP800-38B	A692 (C_C) A696 (CPACF_ASM) A701 (CTI_C) A708 (CPACF_C) A716 (X86ASM_C) A721 (AESNI_C)
	CCM	128, 192, 256	Data Encryption and Decryption	SP800-38C	A692 (C_C) A696 (CPACF_ASM) A701 (CTI_C) A708 (CPACF_C) A716 (X86ASM_C) A721 (AESNI_C)
	GMAC	128, 192, 256	MAC Generation and Verification	SP800-38D	A692 (C_C) A696 (CPACF_ASM) A701 (CTI_C) A708 (CPACF_C) A716 (X86ASM_C) A721 (AESNI_C)
DRBG	CTR_DRBG: AES-128, AES-192, AES-256 with derivation function, with and without PR	N/A	Deterministic Random Bit Generation	SP800-90A	A692 (C_C) A694 (RFC4106EIV_CPACF_ASM) A695 (RFC4106IIV_CPACF_ASM) A696 (CPACF_ASM) A698 (RFC4106EIV_CTI_C) A699 (RFC4106IIV_CTI_C) A701 (CTI_C) A702 (RFC4106EIV_C_C) A703 (RFC4106IIV_C_C) A706 (RFC4106EIV_CPACF_C) A707 (RFC4106IIV_CPACF_C) A708 (CPACF_C) A709 (RFC4106EIV_AESNI_ASM) A710 (RFC4106IIV_AESNI_ASM) A711 (AESNI_ASM) A714 (RFC4106EIV_X86ASM_C) A715 (RFC4106IIV_X86ASM_C) A716 (X86ASM_C) A719 (RFC4106EIV_AESNI_C) A720 (RFC4106IIV_AESNI_C) A721 (AESNI_C)

©2021 SUSE, LLC., atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm	Mode/Method	Key Lengths, Curves or Moduli (bits)	Use	Standard	CAVP Certs.
	Hash_DRBG: SHA-1, SHA2-256, SHA2-384, SHA2-512 with and without PR	N/A	Deterministic Random Bit Generation	SP800-90A	A692 (C_C) A694 (RFC4106EIV_CPACF_ASM) A695 (RFC4106IIV_CPACF_ASM) A698 (RFC4106EIV_CTI_C) A699 (RFC4106IIV_CTI_C) A701 (CTI_C) A702 (RFC4106EIV_C_C) A703 (RFC4106IIV_C_C) A706 (RFC4106EIV_CPACF_C) A707 (RFC4106IIV_CPACF_C) A708 (CPACF_C) A709 (RFC4106EIV_AESNI_ASM) A710 (RFC4106IIV_AESNI_ASM) A711 (AESNI_ASM) A714 (RFC4106EIV_X86ASM_C) A715 (RFC4106IIV_X86ASM_C) A716 (X86ASM_C) A719 (RFC4106EIV_AESNI_C) A720 (RFC4106IIV_AESNI_C) A721 (AESNI_C) A723 (AVX2) A724 (AVX) A725 (SSSE3) A726 (X86ASM_ASM)
	HMAC_DRBG: HMAC-[SHA-1, SHA2-256, SHA2-384, SHA2-512] with and without PR	N/A	Deterministic Random Bit Generation	SP800-90A	A692 (C_C) A694 (RFC4106EIV_CPACF_ASM) A695 (RFC4106IIV_CPACF_ASM) A698 (RFC4106EIV_CTI_C) A699 (RFC4106IIV_CTI_C) A701 (CTI_C) A702 (RFC4106EIV_C_C) A703 (RFC4106IIV_C_C) A706 (RFC4106EIV_CPACF_C) A707 (RFC4106IIV_CPACF_C) A708 (CPACF_C) A709 (RFC4106EIV_AESNI_ASM) A710 (RFC4106IIV_AESNI_ASM) A711 (AESNI_ASM) A714 (RFC4106EIV_X86ASM_C) A715 (RFC4106IIV_X86ASM_C) A716 (X86ASM_C) A719 (RFC4106EIV_AESNI_C)

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm	Mode/Method	Key Lengths, Curves or Moduli (bits)	Use	Standard	CAVP Certs.
					A720 (RFC4106IIV_AESNI_C) A721 (AESNI_C) A723 (AVX2) A724 (AVX) A725 (SSSE3) A726 (X86ASM_ASM)
HMAC	HMAC-[SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512]	112 or greater	Message authentication code	FIPS198-1	A692 (C_C) A708 (CPACF_C) A723 (AVX2) A724 (AVX) A725 (SSSE3)
	HMAC-[SHA3-224, SHA3-256, SHA3-384, SHA3-512]	112 or greater	Message authentication code	FIPS198-1	A690 (SHA3_C_C)
RSA	PKCS#1v1.5: SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	2048, 3072, 4096	Digital Signature Verification	FIPS186-4	A692 (C_C) A708 (CPACF_C) A723 (AVX2) A724 (AVX) A725 (SSSE3)
SHA-3	SHA3-224, SHA3-256, SHA3-384, SHA3-512	N/A	Message Digest	FIPS202	A690 (SHA3_C_C)
SHS	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	N/A	Message Digest	FIPS180-4	A692 (C_C) A708 (CPACF_C) A723 (AVX2) A724 (AVX) A725 (SSSE3)
	SHA-1, SHA2-256, SHA2-512	N/A	Message Digest	FIPS180-4	A722 (MB)
Triple-DES (three-key)	ECB, CBC, CTR	192	Data Encryption and Decryption	SP800-67 SP800-38A	A692 (C_C) A696 (CPACF_ASM) A708 (CPACF_C) A716 (X86ASM_C) A726 (X86ASM_ASM)
	CMAC	192	MAC Generation and Verification	SP800-67 SP800-38B	A692 (C_C) A696 (CPACF_ASM) A708 (CPACF_C) A716 (X86ASM_C)
KTS	AES-KW	128, 192, 256	Key wrapping	SP800-38F	A691 (KW_C_C) A700 (KW_CTI_C) A705 (KW_CPACF_C) A713 (KW_X86ASM_C) A718 (KW_AESNI_C)
	AES-GCM, AES-CCM	128, 192, 256	Key wrapping	SP800-38F	A692 (C_C) A696 (CPACF_ASM)

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm	Mode/Method	Key Lengths, Curves or Moduli (bits)	Use	Standard	CAVP Certs.
					A701 (CTI_C) A708 (CPACF_C) A711 (AESNI_ASM) A716 (X86ASM_C) A721 (AESNI_C) A695 (RFC4106IIV_CPACF_ASM) A699 (RFC4106IIV_CTI_C) A703 (RFC4106IIV_C_C) A707 (RFC4106IIV_CPACF_C) A710 (RFC4106IIV_AESNI_ASM) A715 (RFC4106IIV_X86ASM_C) A720 (RFC4106IIV_AESNI_C) A694 (RFC4106EIV_CPACF_ASM) A698 (RFC4106EIV_CTI_C) A702 (RFC4106EIV_C_C) A706 (RFC4106EIV_CPACF_C) A709 (RFC4106EIV_AESNI_ASM) A714 (RFC4106EIV_X86ASM_C) A719 (RFC4106EIV_AESNI_C)
	Combination AES-CBC and HMAC-[SHA-1, SHA2-256, SHA2-512]; AES-CTR and HMAC-[SHA-1, SHA2-256, SHA2-384, SHA2-512]	128, 192, 256	Key wrapping	SP800-38F FIPS140-2 IG D.9	AES A692 (C_C) A696 (CPACF_ASM) A701 (CTI_C) A708 (CPACF_C) A711 (AESNI_ASM) A716 (X86ASM_C) A721 (AESNI_C) HMAC A692 (C_C) A708 (CPACF_C) A723 (AVX2) A724 (AVX) A725 (SSSE3)
	Combination Triple-DES-CBC and HMAC-[SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512]	192	Key wrapping	SP800-38F FIPS140-2 IG D.9	Triple-DES A692 (C_C) A696 (CPACF_ASM) A708 (CPACF_C) A716 (X86ASM_C) A726 (X86ASM_ASM) HMAC A692 (C_C) A708 (CPACF_C) A723 (AVX2) A724 (AVX)

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm	Mode/Method	Key Lengths, Curves or Moduli (bits)	Use	Standard	CAVP Certs.
					A725 (SSSE3)

4.4.2 Non-Approved-But-Allowed Algorithms

Table 8 describes the non-Approved but allowed algorithms in FIPS mode:

Table 8: Non-Approved but Allowed Algorithms.

Algorithm	Use
NDRNG	The module obtains the entropy data from NDRNG to seed the DRBG
RSA encrypt/decrypt primitives with keys equal or greater than 2048 bits (including 15360 bits or more)	Key transport; allowed per [FIPS140-2_IG] D.8

4.4.3 Non-Approved Algorithms

Table 9 shows the non-Approved cryptographic algorithms implemented in the module that are only available in non-FIPS mode.

Table 9: Non-Approved Cryptographic Algorithms.

Algorithm	Implementation name	Use
Generic GCM encryption with external IV	gcm(aes) with external IV	Data encryption
RFC4106 GCM encryption with external IV	rfc4106(gcm(aes)) with external IV	Data encryption
AES-XTS with 192-bit keys	xts	Data encryption and decryption
Anubis block cipher	anubis-generic	Data encryption and decryption
Blowfish block cipher	Any blowfish	Data encryption and decryption
Camellia block cipher	Any camellia	Data encryption and decryption
CAST5, CAST6	cast5-generic, cast6-generic	Data encryption and decryption
Serpent block cipher	Any serpent	Data encryption and decryption
Twofish block cipher	Any twofish	Data encryption and decryption
ARC4 stream cipher	ecb(arc4)-generic	Data encryption and decryption
Salsa20 stream cipher	Any salsa20	Data encryption and decryption
ChaCha20 stream cipher	Any chacha20	Data encryption and decryption
RSA digital signature sign primitive operation	rsa	Digital signature generation

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm	Implementation name	Use
RSA digital signature verify primitive operation with keys smaller than 2048 bits.	rsa	Digital signature verification
EC Key Generation	ecdh	EC key generation and shared secret computation
RSA encrypt/decrypt with keys smaller than 2048 bits	rsa	Key wrapping
HMAC with less than 112 keys	hmac	Message authentication code
MD4, MD5	md4, md5	Message digest
RACE Integrity Primitives Evaluation Message Digest (RIPEMD)	Any rmd	Message digest
Tiger Hashing	Any tgr	Message digest
Whirlpool	Any wp	Message digest
GHASH	ghash	Message digest outside the GCM mode
Random number generator	Any cprng	Random number generator
EC Diffie-Hellman shared secret computation	ecdh	Shared secret computation
Diffie-Hellman shared secret computation	dh	Shared secret computation
z13 Platform Only (in addition to the above algorithms)		
Algorithms in Table 7 with “generic” implementation	aes-generic des3_ede-generic sha3-224-generic sha3-256-generic sha3-384-generic sha3-512-generic sha512-generic sha384-generic hmac([sha3*]-generic) hmac([sha512, sha384]-generic)	Data encryption and decryption Message digest Message authentication code

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

5 Physical Security

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

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

6 Operational Environment

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

6.1 Policy

The operating system is restricted to a single operator mode of operation (i.e., concurrent operators are explicitly excluded). The application that requests cryptographic services is the single user of the module, even when the application is serving multiple of its own users.

The ptrace system call, the debugger gdb and strace shall not be used. In addition, other tracing mechanisms offered by the Linux environment, such as ftrace or systemtap shall not be used.

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

7 Cryptographic Key Management

Table 10 summarizes the Critical Security Parameters (CSPs) that are used by the cryptographic services implemented in the module:

Table 10: LIfe cycle of keys and other CSPs.

Name	Generation	Entry and Output	Zeroization
AES key	N/A	The keys are passed into the module via API input parameters in plaintext.	Zeroized when freeing the cipher handler.
Triple-DES key			
HMAC key			
Entropy input string	Obtained from NDRNG.	N/A	Zeroized when freeing the cipher handler.
DRBG internal state: V value, C value, key (if applicable) and seed material	Derived from entropy input as defined in SP800-90A.	N/A	Zeroized when freeing the cipher handler.
RSA public/private keys	N/A	The key is passed into the module via API input parameters in plaintext.	Zeroized when freeing the cipher handler.

The following sections describe how keys and other CSPs are managed during their life cycle.

7.1 Random Number Generation

The module employs a SP 800-90A DRBG as a random number generator for the creation of random numbers. In addition, the module provides a Random Number Generation service to applications. The DRBG supports the Hash_DRBG, HMAC_DRBG and CTR_DRBG mechanisms. The DRBG is initialized during module initialization.

The module uses a Non-Deterministic Random Number Generator (NDRNG) as the entropy source. The NDRNG is based on the Linux RNG and the CPU Time Jitter RNG, both within the module's logical boundary.

For seeding the DRBG, the module obtains an amount of entropy input data from the NDRNG that is 1.5 times the security strength expected for the DRBG method. For reseeding, the module obtains an amount of entropy input data equivalent to the security strength of the DRBG method.

For seeding and reseeding, the module ensures the following entropy amount per each DRBG security strength:

- DRBG with 128 bits of security strength: 128 bits of entropy in the entropy input from the NDRNG.
- DRBG with 192 bits of security strength: 192 bits of entropy in the entropy input from the NDRNG.
- DRBG with 256 bits of security strength: 256 bits of entropy in the entropy input from the NDRNG.

Therefore, the module ensures that the NDRNG always provides the required amount of entropy to meet the security strength of the DRBG methods during initialization (seed) and reseeding.

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

The module performs conditional self-tests on the output of NDRNG to ensure that consecutive random numbers do not repeat, and performs DRBG health tests as defined in Section 11.3 of [SP800-90A].

7.2 Key Generation

The module does not provide any dedicated key generation service. However, the Random Number Generation service can be called by the user to obtain random numbers that can be used as keying material for symmetric algorithms and MAC.

7.3 Key Establishment

The module provides AES key wrapping per [SP800-38F] and RSA key wrapping or encapsulation as allowed by [FIPS140-2_IG] D.9.

The module provides approved key transport methods as permitted by IG D.9. These key transport methods are provided either by using an approved key wrapping algorithm, an approved authenticated encryption mode, or a combination method of approved symmetric encryption (AES, Triple-DES) and HMAC-SHA-1/SHA-2 for the AES/Triple-DES modes and HMAC implementations in Table 7.

Table 7 and Table 8 specify the key sizes allowed in the FIPS mode of operation for the specific algorithms that participate in the key agreement methods. According to “Table 2: Comparable strengths” in [SP800-57], the key sizes of AES and Triple-DESkey wrapping, and RSA key wrapping provide the following security strengths:

- AES-KW key wrapping; key establishment methodology provides between 128 and 256 bits of encryption strength.
- AES-GCM and AES-CCM authenticated encryption key wrapping: key establishment methodology provides between 128 and 256 bits of encryption strength.
- Combination method of approved AES-CBC and AES-CTR block mode and message authentication code (HMAC-SHA-1/SHA-2) key wrapping: key establishment methodology provides between 128 and 256 bits of encryption strength.
- Combination method of approved Triple-DES-CBC block mode and message authentication code (HMAC-SHA-1/SHA-2) key wrapping: key establishment methodology provides 112 bits of encryption strength.
- RSA key wrapping: key establishment methodology provides between 112 and 256 bits of encryption strength.

7.4 Key/CSP Entry and Output

The module does not support manual key entry. It supports electronic entry of symmetric keys, HMAC keys and asymmetric keys via API input parameters in plaintext form. The module does not produce key output outside its physical boundary.

The keys can thus be entered into the module or output from the module in plaintext via API parameters, to and from the calling application only.

7.5 Key/CSP Storage

The module does not perform persistent storage of keys. The keys and CSPs are stored as plaintext in the RAM. The only exceptions are the HMAC keys and RSA public keys used for the integrity tests, which are stored in the module and rely on the operating system for protection.

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

7.6 Key/CSP Zeroization

The application is responsible for calling the appropriate destruction functions from the Kernel Crypto API. When a calling application calls the appropriate API function, that function overwrites the memory with zeros and deallocates the memory when the cipher handler is freed.

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

8 Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC)

The test platforms as shown in Table 3 are compliant to 47 CFR FCC Part 15, Subpart B, Class A (Business use).

©2021 SUSE, LLC., atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

9 Self Tests

The module performs both power-up self-tests at module initialization and conditional tests during operation to ensure that the module is not corrupted and that the cryptographic algorithms work as expected.

The services are only available when the power-up self-tests have succeeded. If the power-up self-tests pass, a success message is recorded in the dmesg log and the module transitions to be operational, and then all cryptographic services are available. If the power-up self-tests fail, the module outputs an error message and enters the error state.

On-demand self-tests can be invoked by rebooting the operating system.

Conditional tests are performed during the operation of the module. If a conditional test is successful, the module remains operational. If a conditional test fails, the module outputs an error message and enters the error state.

When the module is in the error state, data output is inhibited, and no further operations are possible. The operating system must be rebooted to recover from the error state.

Table 11 lists all the self-tests performed by the module. For algorithms that have more than one implementation in the module (per Table 7), the module performs the self-tests independently for each of these implementations.

Table 11: Self tests.

Self Test	Description
Power-up tests performed at power-up and on demand	
Cryptographic Algorithm Known Answer Tests (KATs)	KATs for AES in ECB, CBC, CTR, GCM, CCM and XTS modes; encryption and decryption are performed separately.[3] KATs for Triple-DES in ECB, CBC and CTR modes; encryption and decryption are performed separately. KATs for AES and Triple-DES CMAC, MAC generation and verification. KATs for SHA-1, SHA2-224, SHA2-256, SHA2-384 and SHA2-512. KATs for SHA3-224, SHA3-256, SHA3-384 and SHA3-512. KATs for HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384 and HMAC-SHA2-512. KATs for HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384 and HMAC-SHA3-512. KATs for Hash_DRBG (SHA2-256), HMAC_DRBG (HMAC-SHA2-256), and CTR_DRBG (AES-128, AES-192, AES-256), with and without PR, and health tests per Section 11.3 of SP800-90A. KAT for RSA public key encryption and private key decryption with 2048 bit keys. KAT for RSA signature verification is covered by the integrity tests performed on kernel object files.
Software Integrity Test	The module uses the HMAC-SHA2-256 algorithm for the integrity test of the static kernel binary and the fipscheck application. The HMAC calculation is performed by the fipscheck application itself. The module uses RSA signature verification using SHA2-256 with a 4096-bit key for the integrity test of each of the kernel object files loaded during boot-up time.

[3] The KAT for AES-KW is covered by the KAT for AES-GCM per IG 9.4.

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Self Test	Description
Conditional tests performed during operation	
Continuous Random Number Generator Test (CRNGT)	The module performs conditional self-tests on the output of NDRNG to ensure that consecutive random numbers do not repeat.
On demand execution of self tests	
On Demand Testing	Invocation of the self tests on demand can be achieved by rebooting the operating system. This will trigger a new power-up self-test procedure.

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

10 Guidance

10.1 Crypto Officer Guidance

The binaries of the module are contained in the RPM packages for delivery. The Crypto Officer shall follow this Security Policy to configure the operational environment and install the module to be operated as a FIPS 140-2 validated module.

Table 12 shows the RPM packages that contain the FIPS validated module. They are separated by Operational System version and are applicable for all the tested platforms for the specific Operational System version.

Table 12: RPM packages.

Operational System (All Platforms)	RPM Package
Linux Enterprise Server 12 SP4	dracut-044.2-10.32.1.rpm dracut-fips-044.2-10.32.1.rpm kernel-default-4.12.14-95.60.rpm libkcapi-0.13.0-8.1.rpm
Linux Enterprise Server 12 SP5	dracut-044.2-10.32.1.rpm dracut-fips-044.2-10.32.1.rpm kernel-default-4.12.14-122.37.1.rpm libkcapi-0.13.0-8.1.rpm

Additional kernel components that register with the Kernel Crypto API shall not be loaded as the kernel configuration is fixed in approved mode.

10.1.1 Module Installation

The Crypto Officer shall install the RPM packages containing the module as listed in Table 12 using the zypper tool. The integrity of the RPM package is automatically verified during the installation, and the Crypto Officer shall not install the RPM package if there is any integrity error.

10.1.2 Operating Environment Configurations

The operating environment needs to be configured to support FIPS, so the following steps shall be performed with the root privilege:

Install the dracut-fips RPM package:

```
# zypper install dracut-044.2-10.32.1.rpm dracut-fips-044.2-10.32.1.rpm
```

Recreate the INITRAMFS image:

```
# dracut -f
```

After regenerating the initrd, the Crypto Officer has to append the following parameter in the /etc/default/grub configuration file in the GRUB_CMDLINE_LINUX_DEFAULT line:

```
fips=1
```

After editing the configuration file, please run the following command to change the setting in the boot loader:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

©2021 SUSE, LLC., atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

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

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

The partition of /boot is located on /dev/sda1 in this example. Therefore, the following string needs to be appended in the aforementioned grub file:

```
"boot=/dev/sda1"
```

Reboot to apply these settings.

Now, the operating environment is configured to support FIPS operation. The Crypto Officer should check the existence of the file /proc/sys/crypto/fips_enabled, and verify it contains a numeric value “1”. If the file does not exist or does not contain “1”, the operating environment is not configured to support FIPS and the module will not operate as the proper FIPS validated module.

10.2 User Guidance

10.2.1 Cipher References and Priority

The cryptographic module provides multiple implementations of different algorithms as shown in Section 4.4 and Appendix B.

If more than one of the kernel components are loaded, the respective implementation can be requested by using the following cipher mechanism strings with the initialization calls (such as crypto_alloc_blkcipher):

- aes-generic kernel component: “aes-generic”.
- aesni-intel kernel component: "__aes-aesni".
- aes-x86_64 kernel component: "aes-asm".

The AES cipher can also be loaded by simply using the string "aes" with the initialization call. In this case, the AES implementation whose kernel component is loaded with the highest priority is used. The following priority exists:

- aesni-intel.
- aes-x86_64.
- aes-generic.

For example: If the kernel components aesni-intel and aes-asm are loaded and the caller uses the initialization call (such as crypto_alloc_blkcipher) with the cipher string of "aes", the aesni-intel implementation is used. On the other hand, if only the kernel components of aes-x86_64 and aes-generic are loaded, the cipher string of "aes" implies that the aes-x86_64 implementation is used.

The discussion about the naming and priorities of the AES implementation also applies when cipher strings are used that include the block modes, such as "cbc(aes-asm)", "cbc(aes)", or "cbc(__aes-aesni)".

When using the module, the user shall utilize the Linux kernel crypto API-provided memory allocation mechanisms. In addition, the user shall not use the function copy_to_user() on any portion of the data structures used to communicate with the Linux kernel crypto API.

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

10.2.2 AES XTS

As specified in SP800-38E, the AES algorithm in XTS mode is designed for the cryptographic protection of data on storage devices. Thus, it can only be used for the disk encryption functionality offered by dm-crypt (i.e., the hard disk encryption scheme). For dm-crypt, the length of a single data unit encrypted with AES XTS mode is at most 65,536 bytes (64KiB of data), which does not exceed 2^{20} AES blocks (16MiB of data).

To meet the requirement stated in IG A.9, the module implements a check to ensure that the two AES keys used in AES XTS mode are not identical.

Note: AES-XTS shall be used with 128 and 256-bit keys only. AES-XTS with 192-bit keys is not an Approved service.

10.2.3 AES GCM IV

In case the module's power is lost and then restored, the key used for the AES GCM encryption or decryption shall be redistributed.

When a GCM IV is used for encryption, the module complies with IG A.5 Scenario 2 [FIPS140-2_IG], in which the GCM IV is generated internally at its entirety randomly by the module's DRGB. The DRBG seeds itself from the NDRNG, which is within the module's boundary. The GCM IV is 96 bits in length. Per Section 7.1, this 96-bit IV contains 96 bits of entropy.

When a GCM IV is used for decryption, the responsibility for the IV generation lies with the party that performs the AES GCM encryption and therefore there is no restriction on the IV generation.

10.2.4 Triple-DES encryption

Data encryption using the same three-key Triple-DES key shall not exceed 2^{16} Triple-DES blocks (2GiB of data), in accordance with SP800-67 and IG A.13. The user of the module is responsible for ensuring the module's compliance with this requirement.

10.3 Handling Self Test Errors

Self test failure within the kernel crypto API module will panic the kernel and the operating system will not load and/or halt immediately.

Error recovery and return to operational state can be accomplished by rebooting the system. If the failure continues, the Crypto Officer must re-install the software package and make sure to follow all instructions. If the software was downloaded, the package hash value must be verified to confirm a proper download. Please contact SUSE if these steps do not resolve the problem.

The kernel dumps self-test success and failure messages into the kernel message ring buffer. Post boot, the messages are moved to */var/log/messages*.

Use **dmesg** to read the contents of the kernel ring buffer. The format of the ringbuffer (**dmesg**) output is:

```
alg: self-tests for %s (%s) passed
```

Typical messages are similar to "alg: self-tests for xts(aes) (xts(aes-x86_64)) passed" for each algorithm/sub-algorithm type.

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

11 Mitigation of Other Attacks

The module does not offer mitigation of other attacks.

©2021 SUSE, LLC., atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Appendix A Glossary and Abbreviations

AES	Advanced Encryption Specification
AES_NI	Intel Advanced Encryption Standard (AES) New Instructions
AVX	Advanced Vector Extensions
AVX2	Advanced Vector Extensions 2
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining Message Authentication Code
CE	Cryptographic Extensions
CLMUL	Carry-less Multiplication
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CPACF	CP Assist for Cryptographic Functions
CSP	Critical Security Parameter
CTI	Constant-Time
CTR	Counter Mode
DES	Data Encryption Standard
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
FIPS	Federal Information Processing Standards Publication
GCM	Galois Counter Mode
HMAC	Hash-based Message Authentication Code, keyed-Hash Message Authenticated Code
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
PKCS	Public Key Cryptography Standards
PR	Prediction Resistance
RNG	Random Number Generator
RPM	Red hat Package Manager
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SSSE3	Supplemental Streaming SIMD (Single Instruction, Multiple Data) Extensions 3

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

XTS	XEX Tweakable Block Cipher with Ciphertext Stealing

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Appendix B Algorithm Implementations

Table 13 describes the names utilized in the algorithm certificates and the implementations to which they refer for the test platforms.

Table 13: Algorithm implementations and their names in the CAVP certificates.

Name	Description
AESNI_ASM	AES cipher with AESNI, and block modes with assembler
AESNI_C	AES cipher with AESNI, and block modes with C
AVX	SHA using AVX instructions
AVX2	SHA using AVX2 instructions
C_C	Cipher in C, block modes in C
CPACF_ASM	AES and Triple-DES using CPACF instructions, block mode using assembler
CPACF_C	AES and Triple-DES using CPACF instructions, block mode using C
CPACF_SHA3	SHA3 using CPACF instructions
CTI_C	Cipher in constant time C implementation, block mode in C
KW_AESNI_C	AES-KW: AES with AESNI, block mode with generic non-optimized C
KW_C_C	AES-KW: AES in C, block mode in C
KW_CPACF_C	AES-KW: AES using CP Assist for Cryptographic Functions (CPACF), block mode using C
KW_CTI_C	AES-KW: AES using constant-time C implementation, block mode using C
KW_X86ASM_C	AES-KW: AES using assembler implementation, block mode using C
MB	Multi-Buffer SHA implementation
RFC4106EIV_AESNI_ASM	RFC4106 GCM with external IV generation: AES using AESNI, block mode using assembler
RFC4106EIV_AESNI_C	RFC4106 GCM with external IV generation: AES using AESNI, block mode using C
RFC4106EIV_C_C	RFC4106 GCM with external IV generation: AES using C, block mode using C
RFC4106EIV_CPACF_ASM	RFC4106 GCM with external IV generation: AES using CP Assist for Cryptographic Functions (CPACF), block mode using assembler
RFC4106EIV_CPACF_C	RFC4106 GCM with external IV generation: AES using CP Assist for Cryptographic Functions (CPACF), block mode using C
RFC4106EIV_CTI_C	RFC4106 GCM with external IV generation: AES using constant-time C, block mode using C

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description
RFC4106EIV_X86ASM_C	RFC4106 GCM with external IV generation: AES using assembler block mode using C
RFC4106IIV_AESNI_ASM	RFC4106 GCM with internal IV generation: AES using AESNI, block mode using assembler
RFC4106IIV_AESNI_C	RFC4106 GCM with internal IV generation: AES using AESNI, block mode using C
RFC4106IIV_C_C	RFC4106 GCM with internal IV generation: AES using C, block mode using C
RFC4106IIV_CPACF_ASM	RFC4106 GCM with internal IV generation: AES using CP Assist for Cryptographic Functions (CPACF), block mode using assembler
RFC4106IIV_CPACF_C	RFC4106 GCM with internal IV generation: AES using CP Assist for Cryptographic Functions (CPACF), block mode using C
RFC4106IIV_CTI_C	RFC4106 GCM with internal IV generation: AES using constant-time C, block mode using C
RFC4106IIV_X86ASM_C	RFC4106 GCM with internal IV generation: AES using assembler block mode using C
SHA3_C_C	SHA-3 implementation in generic C
SSSE3	SHA implementation using SSSE3
X86ASM_ASM	Cipher in assembler, block mode in assembler
X86ASM_C	Cipher in assembler, block mode in C

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

Appendix C References

FIPS 140-2	**FIPS PUB 140-2 - Security Requirements for Cryptographic Modules** http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf
FIPS 140-2_IG	**Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program** August 28, 2020 http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf
FIPS180-4	**Secure Hash Standard (SHS)** http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
FIPS186-4	**Digital Signature Standard (DSS)** http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
FIPS197	**Advanced Encryption Standard** http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
FIPS198-1	**The Keyed Hash Message Authentication Code (HMAC)** http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf
FIPS202	**SHA-3 Standard Permutation-Based Hash and Extendable-Output Functions** https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf
PKCS#1	**Public Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1** http://www.ietf.org/rfc/rfc3447.txt
SP800-38A	**NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques** http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf
SP800-38B	**NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication** http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38b.pdf
SP800-38C	NIST Special Publication 800-38C - Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf
SP800-38D	**NIST Special Publication 800-38D - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC** http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf
SP800-38E	NIST Special Publication 800-38E - Recommendation for Block Cipher Modes of Operation: The XTS AES Mode for Confidentiality on Storage Devices http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38e.pdf
SP800-38F	**NIST Special Publication 800-38F - Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping** http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.

SP800-67 **NIST Special Publication 800-67 Revision 1 - Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher**
http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-67r1.pdf

SP800-90A NIST Special Publication 800-90A Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

SP800-131A NIST Special Publication 800-131A Revision 1- Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf

©2021 SUSE, LLC., atsec information security.

This document can be reproduced and distributed only whole and intact, including this copyright notice.