Canonical Ltd.

Canonical Ltd. Ubuntu 22.04 GnuTLS Cryptographic Module

Version 3.7.3-4ubuntu1.2+Fips1.1

FIPS 140-3 Non-Proprietary Security Policy

Version 1.3

Last updated: 10-22-2024

Prepared by:

atsec information security corporation

4516 Seton Center Pkwy, Suite 250

Austin, TX 78759

www.atsec.com

Table of Contents

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

List of Tables

List of Figures

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

1 General

1.1 Overview

This document is the non-proprietary FIPS 140-3 Security Policy for version 3.7.3-4ubuntu1.2+Fips1.1 of the Canonical Ltd. Ubuntu 22.04 GnuTLS Cryptographic Module. It has a one-to-one mapping to SP 800-140B starting with section B.2.1 named “General” that maps to section 1 in this document and ending with section B.2.12 named “Mitigation of other attacks” that maps to section 12 in this document.

1.2 Security Levels

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

Table 1: Security Levels

1.3 Additional Information

N/A

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

The Canonical Ltd. Ubuntu 22.04 GnuTLS Cryptographic Module (hereafter referred to as "the module") provides cryptographic services to applications running in the user space of the underlying operating system through a C language Application Program Interface (API).

Module Type: Software

Module Embodiment: MultiChipStand

Module Characteristics:

Cryptographic Boundary:

The software block diagram below shows the cryptographic boundary of the module, and its interfaces with the operational environment.

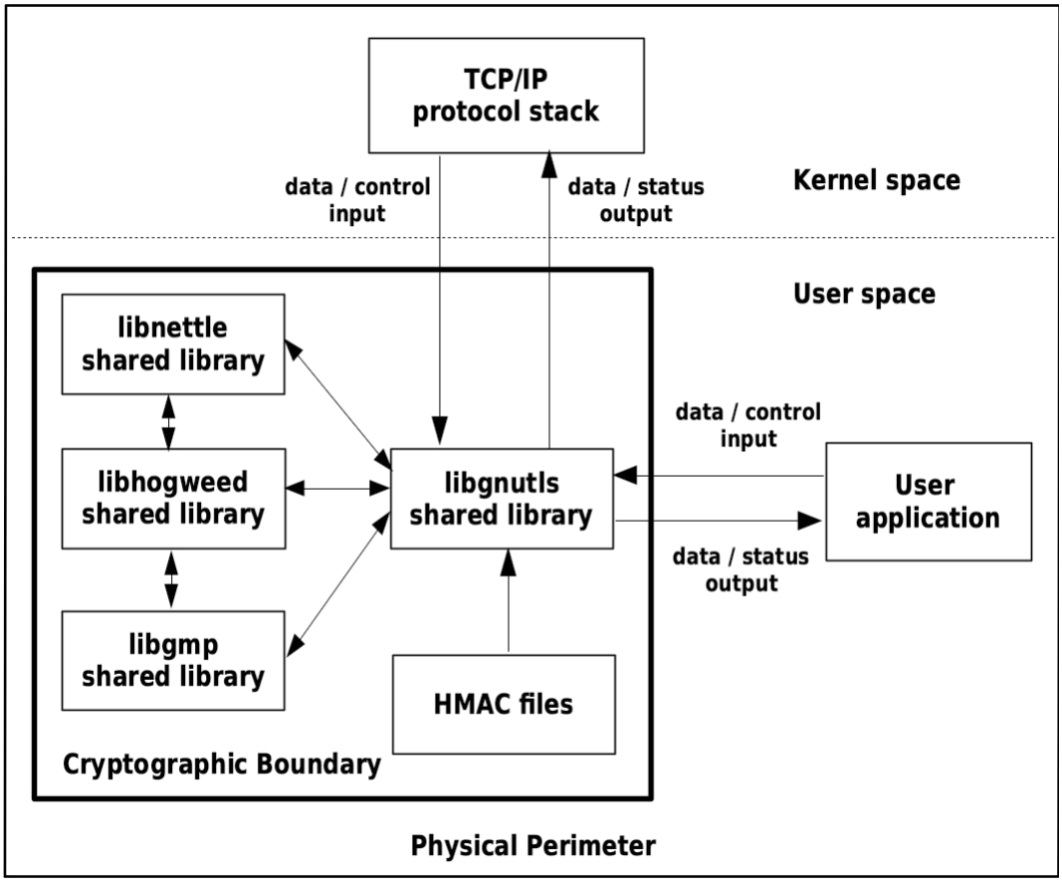

Figure 1: Block Diagram

Tested Operational Environment's Physical Perimeter (TOEPP):

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

The TOEPP (tested operational environment's physical perimeter) of the module is defined as the general-purpose computer on which the module is installed.

2.2 Tested and Vendor Affirmed Module Version and Identification

Tested Module Identification – Hardware:

N/A for this module.

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

Package or File Name	Software/ Firmware Version	Features	Integrity Test
libgnutls.so.30, libnettle.so.8, libhogweed.so.6, libgmp.so.10 on Supermicro SYS-1019P-WTR	3.7.3-4ubuntu1.2+Fips1.1	N/A	HMAC-SHA2-256
libgnutls.so.30, libnettle.so.8, libhogweed.so.6, libgmp.so.10 on Amazon Web Services (AWS) c6g.metal	3.7.3-4ubuntu1.2+Fips1.1	N/A	HMAC-SHA2-256
libgnutls.so.30, libnettle.so.8, libhogweed.so.6, libgmp.so.10 on IBM z15	3.7.3-4ubuntu1.2+Fips1.1	N/A	HMAC-SHA2-256

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

Tested Module Identification – Hybrid Disjoint Hardware:

N/A for this module.

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Ubuntu 22.04 LTS (Jammy Jellyfish)	Supermicro SYS-1019P-WTR	Intel® Xeon® Gold 6226	Yes	N/A	3.7.3-4ubuntu1.2+Fips1.1
Ubuntu 22.04 LTS (Jammy Jellyfish)	Amazon Web Services (AWS) c6g.metal	AWS Graviton2	Yes	N/A	3.7.3-4ubuntu1.2+Fips1.1
Ubuntu 22.04 LTS (Jammy Jellyfish)	IBM z15	z15	Yes	N/A	3.7.3-4ubuntu1.2+Fips1.1

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Ubuntu 22.04 LTS (Jammy Jellyfish)	Supermicro SYS-1019P-WTR	Intel® Xeon® Gold 6226	No	N/A	3.7.3-4ubuntu1.2+Fips1.1
Ubuntu 22.04 LTS (Jammy Jellyfish)	Amazon Web Services (AWS) c6g.metal	AWS Graviton2	No	N/A	3.7.3-4ubuntu1.2+Fips1.1
Ubuntu 22.04 LTS (Jammy Jellyfish)	IBM z15	z15	No	N/A	3.7.3-4ubuntu1.2+Fips1.1

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

The module makes use of hardware acceleration provided by the hardware platform. Namely, AES-NI from the Intel based platform, NEON and Cryptography Extension for the Graviton2 based platform and CPACF for the z15 based platforms, listed in the *Tested Operational Environments - Software, Firmware, Hybrid* table. Out of these, only CPACF is considered as PAI and other two are considered as PAA.

Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid:

N/A for this module.

CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when so ported if the specific operational environment is not listed on the validation certificate.

2.3 Excluded Components

N/A

2.4 Modes of Operation

Modes List and Description:

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

Table 4: Modes List and Description

When the module starts up successfully, after passing all the pre-operational and conditional cryptographic algorithms self-tests (CASTs), the module is operating in the approved mode of operation by default. Please see section 4 for the details on service indicator provided by the module that identifies when an approved service is called.

Mode Change Instructions and Status:

If the module is in the approved mode, it can be transitioned to the non-approved mode by calling one of the non-approved services listed in section 4. If the module is in the non-approved mode, the module can be transitioned to the approved mode by calling one of the approved services listed in section 4.

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Degraded Mode Description:

N/A

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A3665	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A3665	Key Length - 128, 256	SP 800-38C
AES-GCM	A3665	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D
HMAC-SHA-1	A3665	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A3665	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A3665	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A3665	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A3665	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
SHA-1	A3665	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-224	A3665	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-256	A3665	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-384	A3665	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-512	A3665	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
AES-CBC	A3667	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A3667	Direction - Generation, Verification Key Length - 128, 256	SP 800-38B
AES-GCM	A3667	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D
AES-GMAC	A3667	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D
Counter DRBG	A3667	Prediction Resistance - No Mode - AES-256 Derivation Function Enabled - No	SP 800-90A Rev. 1
ECDSA KeyGen (FIPS186-4)	A3667	Curve - P-256, P-384, P-521 Secret Generation Mode - Testing Candidates	FIPS 186-4

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm	CAVP Cert	Properties	Reference
ECDSA KeyVer (FIPS186-4)	A3667	Curve - P-256, P-384, P-521	FIPS 186-4
ECDSA SigGen (FIPS186-4)	A3667	Component - No Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
ECDSA SigVer (FIPS186-4)	A3667	Component - No Curve - P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
HMAC-SHA-1	A3667	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A3667	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A3667	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A3667	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A3667	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A3667	Domain Parameter Generation Methods - P-256, P-384, P-521 Scheme - ephemeralUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3
KAS-FFC-SSC Sp800-56Ar3	A3667	Domain Parameter Generation Methods - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 Scheme - dhEphem - KAS Role - initiator, responder	SP 800-56A Rev. 3
KDF TLS (CVL)	A3667	TLS Version - v1.0/1.1	SP 800-135 Rev. 1
PBKDF	A3667	Iteration Count - Iteration Count: 10-1000 Increment 1 Password Length - Password Length: 8-128 Increment 1	SP 800-132
RSA KeyGen (FIPS186-4)	A3667	Key Generation Mode - B.3.2 Modulo - 2048, 3072, 4096 Hash Algorithm - SHA2-384 Primality Tests - Table C.2 Private Key Format - Standard	FIPS 186-4
RSA SigGen (FIPS186-4)	A3667	Signature Type - PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096	FIPS 186-4
RSA SigVer (FIPS186-4)	A3667	Signature Type - PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096	FIPS 186-4
Safe Primes Key Generation	A3667	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	SP 800-56A Rev. 3
SHA-1	A3667	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-224	A3667	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-256	A3667	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm	CAVP Cert	Properties	Reference
SHA2-384	A3667	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-512	A3667	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
TLS v1.2 KDF RFC7627 (CVL)	A3667	Hash Algorithm - SHA2-256, SHA2-384	SP 800-135 Rev. 1
AES-CBC	A3708	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A3708	Key Length - 128, 256	SP 800-38C
AES-CMAC	A3708	Direction - Generation, Verification Key Length - 128, 256	SP 800-38B
AES-GCM	A3708	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D
AES-CBC	A3709	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A3709	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D
AES-CBC	A3711	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A3711	Key Length - 128, 256	SP 800-38C
AES-CMAC	A3711	Direction - Generation, Verification Key Length - 128, 256	SP 800-38B
AES-GCM	A3711	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D
AES-CBC	A3712	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A3712	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D
AES-CBC	A3713	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A3713	Direction - Decrypt, Encrypt IV Generation - External IV Generation Mode - 8.2.1 Key Length - 128, 256	SP 800-38D
AES-CBC	A3714	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A3714	Direction - Generation, Verification Key Length - 128, 256	SP 800-38B
HMAC-SHA-1	A3714	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A3714	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A3714	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A3714	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm	CAVP Cert	Properties	Reference
HMAC-SHA2-512	A3714	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
SHA-1	A3714	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-224	A3714	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-256	A3714	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-384	A3714	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-512	A3714	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
AES-CFB8	A3670	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A3716	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A3717	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-XTS Testing Revision 2.0	A3668	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38E
HMAC-SHA-1	A3710	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A3710	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A3710	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A3710	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A3710	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
SHA-1	A3710	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-224	A3710	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-256	A3710	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-384	A3710	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
SHA2-512	A3710	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 180-4
KDA HKDF Sp800-56Cr1	A3666	Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-65336 Increment 8 HMAC Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	SP 800-56C Rev. 2
SHA3-224	A3669	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-256	A3669	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-384	A3669	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-512	A3669	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm	CAVP Cert	Properties	Reference
SHA3-224	A3715	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-256	A3715	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-384	A3715	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202
SHA3-512	A3715	Message Length - Message Length: 0-65536 Increment 8 Large Message Sizes - 1, 2, 4, 8	FIPS 202

Table 5: Approved Algorithms

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
CKG (asymmetric)	RSA:Asymmetric ECDSA:Asymmetric EC Diffie-Hellman :Asymmetric Safe primes:Asymmetric	N/A	SP 800-133r2 section 4 example 1 without the use of V (refer to additional comment 2 of IG D.H)
CKG (symmetric)	AES:Symmetric HMAC:Symmetric	N/A	SP 800-133r2 section 4 example 1 without the use of V (refer to additional comment 2 of IG D.H)

Table 6: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms with No Security Claimed:

Name	Caveat	Use and Function
MD5	Only allowed as the PRF in TLSv1.0 and v1.1 per IG 2.4.A	Message digest used in TLS 1.0 / 1.1 KDF only for legacy use

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

Non-Approved, Not Allowed Algorithms:

Name	Use and Function
Blowfish	Symmetric encryption; Symmetric decryption
Camellia	Symmetric encryption; Symmetric decryption
CAST	Symmetric encryption; Symmetric decryption
ChaCha20	Symmetric encryption; Symmetric decryption
Chacha20 and Poly1305	Authenticated encryption; Authenticated decryption
CMAC with Triple-DES	Message authentication code (MAC)

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Use and Function
DES	Symmetric encryption; Symmetric decryption
Diffie-Hellman (with domain parameters other than safe primes)	Key agreement; Shared secret computation
DSA	Key generation; Domain parameter generation; Digital signature generation; Digital signature verification
ECDSA (with curves other than P-256, P-384, P-512)	Key generation; Public key verification
ECDSA (with curves other than P-256, P-384, P-512 or hash functions other than SHA2-224, SHA2-256, SHA2-384, SHA2-512)	Digital signature generation; Digital signature verification
EC Diffie-Hellman (with curves other than P-256, P-384, P-512)	Key agreement; Shared secret computation
GMAC	Message authentication code (MAC)
GOST	Symmetric encryption; Symmetric decryption; Message digest
HMAC (with keys smaller than 112-bits)	Message authentication code (MAC)
HMAC (with GOST)	Message authentication code (MAC)
MD2, MD4, MD5	Message digest; Message authentication code (MAC)
PBKDF (with non-approved message digest algorithms or using input parameters not meeting requirements stated in section 2.7 of the security policy)	Key derivation
RC2, RC4	Symmetric encryption; Symmetric decryption
RMD160	Message digest; Message authentication code (MAC)
RSA (with keys smaller than 2048 bits or greater than 4096 bits)	Key generation
RSA (with keys smaller than 2048 bits or greater than 4096 bits and/or hash functions other than SHA2-224, SHA2-256, SHA2-384, SHA2-512)	Digital signature generation
RSA (with keys smaller than 1024 bits or greater than 4096 bits and/or hash functions other than SHA2-224, SHA2-256, SHA2-384, SHA2-512)	Digital signature verification
RSA (encapsulation and un-encapsulation with any key sizes)	Key encapsulation; Key un-encapsulation
Salsa20	Symmetric encryption; Symmetric decryption
SEED	Symmetric encryption; Symmetric decryption
Serpent	Symmetric encryption; Symmetric decryption
SRP	Key agreement
STREEBOG	Message digest; Message authentication code (MAC)
Triple-DES	Symmetric encryption; Symmetric decryption
Twofish	Symmetric encryption; Symmetric decryption
UMAC	Message authentication code (MAC)
Yarrow	Random number generation
AES-GCM (when not used in the context of the TLS protocol)	Symmetric encryption; Symmetric decryption

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Table 8: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
Symmetric encryption	BC-UnAuth BC-Auth	Symmetric encryption. AES-GCM is considered approved by the module only used in the context of the TLS protocol.	AES-CBC:128, 192, 256-bit keys with 128-256 bits of key strength AES-CCM:128, 256-bit keys with 128 and 256 bits of key strength AES-GCM:128, 256-bit keys with 128 and 256 bits of key strength AES-CFB8:128, 192, 256-bit keys with 128-256 bits of key strength AES-XTS Testing Revision 2.0:128, 256-bit keys with 128 and 256 bits of key strength	AES-CBC AES-CBC AES-CBC AES-CBC AES-CBC AES-CBC AES-CBC AES-CCM AES-CCM AES-CCM AES-GCM AES-GCM AES-GCM AES-GCM AES-GCM AES-GCM AES-GCM AES-CFB8 AES-CFB8 AES-CFB8 AES-XTS Testing Revision 2.0
Symmetric decryption	BC-UnAuth BC-Auth	Symmetric decryption. AES-GCM is considered approved by the module only used in the context of the TLS protocol	AES-CBC:128, 192, 256-bit keys with 128-256 bits of key strength AES-CCM:128, 256-bit keys with 128 and 256 bits of key strength AES-GCM:128, 256-bit keys with 128 and 256 bits of key strength AES-CFB8:128, 192, 256-bit keys with 128-256 bits of key strength AES-XTS Testing Revision 2.0:128, 256-bit keys with 128 and 256 bits of key strength	AES-CBC AES-CBC AES-CBC AES-CBC AES-CBC AES-CBC AES-CBC AES-CCM AES-CCM AES-CCM AES-GCM AES-GCM AES-GCM AES-GCM AES-GCM AES-GCM AES-GCM AES-CFB8 AES-CFB8 AES-CFB8 AES-XTS Testing Revision 2.0
Message authentication code (MAC)	MAC	Message authentication code (MAC)	HMAC-SHA-1:112-524288 bit keys with	HMAC-SHA-1 HMAC-SHA-1 HMAC-SHA-1

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Type	Description	Properties	Algorithms
			strength of 112-256 bits HMAC-SHA2-224:112-524288 bit keys with strength of 112-256 bits HMAC-SHA2-256:112-524288 bit keys with strength of 112-256 bits HMAC-SHA2-384:112-524288 bit keys with strength of 112-256 bits HMAC-SHA2-512:112-524288 bit keys with strength of 112-256 bits AES-CMAC:128, 256-bit keys with 128 and 256 bits of key strength AES-GMAC:128, 256-bit keys with 128 and 256 bits of key strength	HMAC-SHA-1 HMAC-SHA2-224 HMAC-SHA2-224 HMAC-SHA2-224 HMAC-SHA2-224 HMAC-SHA2-256 HMAC-SHA2-256 HMAC-SHA2-256 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-384 HMAC-SHA2-384 HMAC-SHA2-384 HMAC-SHA2-512 HMAC-SHA2-512 HMAC-SHA2-512 HMAC-SHA2-512 SHA-1 SHA-1 SHA-1 SHA-1 SHA2-224 SHA2-224 SHA2-224 SHA2-224 SHA2-256 SHA2-256 SHA2-256 SHA2-256 SHA2-384 SHA2-384 SHA2-384 SHA2-384 SHA2-512 SHA2-512 SHA2-512 SHA2-512 AES-CMAC AES-CMAC AES-CMAC AES-CMAC AES-GMAC
Message digest	SHA	Message digest		SHA-1 SHA-1 SHA-1 SHA-1 SHA2-224 SHA2-224 SHA2-224 SHA2-224 SHA2-256 SHA2-256 SHA2-256 SHA2-256 SHA2-384

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Type	Description	Properties	Algorithms
				SHA2-384 SHA2-384 SHA2-384 SHA2-512 SHA2-512 SHA2-512 SHA2-512 SHA3-224 SHA3-256 SHA3-384 SHA3-512 SHA3-224 SHA3-256 SHA3-384 SHA3-512
Deterministic random bit generation	CKG DRBG	Deterministic random bit generation in compliance with SP800-90Ar1	Counter DRBG:256-bit keys with 256 bits of key strength	Counter DRBG
Asymmetric key generation	AsymKeyPair-KeyGen CKG	Asymmetric key generation	ECDSA KeyGen (FIPS186-4):P-256, P-384, P-521 elliptic curves with 128-256 bits of key strength RSA KeyGen (FIPS186-4):2048, 3072, 4096-bit keys with 112-149 bits of key strength Safe Primes Key Generation:2048, 3072, 4096, 6144, 8192-bit keys with 112-200 bits of key strength	ECDSA KeyGen (FIPS186-4) RSA KeyGen (FIPS186-4) Safe Primes Key Generation Counter DRBG
Public key verification	AsymKeyPair-KeyVer	Public key verification	ECDSA KeyVer (FIPS186-4):P-256, P-384, P-521 elliptic curves with 128-256 bits of key strength	ECDSA KeyVer (FIPS186-4)
Digital signature generation	DigSig-SigGen	Digital signature generation	ECDSA SigGen (FIPS186-4):P-256, P-384, P-521 elliptic curves with 128-256 bits of strength RSA SigGen	ECDSA SigGen (FIPS186-4) RSA SigGen (FIPS186-4) Counter DRBG SHA2-224 SHA2-224 SHA2-224

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Type	Description	Properties	Algorithms
			(FIPS186-4):2048, 3072, 4096-bit keys with 112-149 bits of key strength	SHA2-224 SHA2-256 SHA2-256 SHA2-256 SHA2-256 SHA2-384 SHA2-384 SHA2-384 SHA2-384 SHA2-512 SHA2-512 SHA2-512 SHA2-512
Digital signature verification	DigSig-SigVer	Digital signature verification	ECDSA SigVer (FIPS186-4):P-256, P-384, P-521 elliptic curves with 128-256 bits of key strength RSA SigVer (FIPS186-4):2048, 3072, 4096-bit keys with 112-149 bits of key strength	ECDSA SigVer (FIPS186-4) RSA SigVer (FIPS186-4) SHA2-224 SHA2-224 SHA2-224 SHA2-224 SHA2-256 SHA2-256 SHA2-256 SHA2-256 SHA2-384 SHA2-384 SHA2-384 SHA2-384 SHA2-512 SHA2-512 SHA2-512 SHA2-512
(EC Diffie-Hellman) shared secret computation	KAS-SSC	EC Diffie-Hellman shared secret computation compliant with scenario 2(1) of IG D.F	KAS-ECC-SSC Sp800-56Ar3:P-256, P-384, P-521 elliptic curves with 128-256 bits of strength	KAS-ECC-SSC Sp800-56Ar3
(Diffie-Hellman) shared secret computation	KAS-SSC	EC Diffie-Hellman shared secret computation compliant with scenario 2(1) of IG D.F	KAS-FFC-SSC Sp800-56Ar3:2048, 3072, 4096, 6144, 8192-bit keys with 112-200 bits of key strength	KAS-FFC-SSC Sp800-56Ar3
Key derivation	KAS-135KDF KAS-56CKDF PBKDF	Key derivation	KDF TLS (CVL):TLS derived secret with 112 to 256 bits of key strength PBKDF:128-4096	KDF TLS PBKDF TLS v1.2 KDF RFC7627 KDA HKDF Sp800-56Cr1 HMAC-SHA-1

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Type	Description	Properties	Algorithms
			bit keys with strength of 128-256 bits TLS v1.2 KDF RFC7627 (CVL):TLS derived secret with 112 to 256 bits of key strength KDA HKDF Sp800-56Cr1:TLS derived secret with 112 to 256 bits of key strength	HMAC-SHA-1 HMAC-SHA-1 HMAC-SHA-1 HMAC-SHA2-224 HMAC-SHA2-224 HMAC-SHA2-224 HMAC-SHA2-224 HMAC-SHA2-256 HMAC-SHA2-256 HMAC-SHA2-256 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-384 HMAC-SHA2-384 HMAC-SHA2-384 HMAC-SHA2-512 HMAC-SHA2-512 HMAC-SHA2-512 HMAC-SHA2-512 SHA-1 SHA-1 SHA-1 SHA-1 SHA2-224 SHA2-224 SHA2-224 SHA2-224 SHA2-256 SHA2-256 SHA2-256 SHA2-256 SHA2-384 SHA2-384 SHA2-384 SHA2-384 SHA2-512 SHA2-512 SHA2-512 SHA2-512
Key wrapping	KTS-Wrap	Key wrapping/unwrapping using AES-CCM, AES-GCM, or AES-CBC with HMAC with 128-bit or 256-bit keys; used in the context of the TLS protocol, in compliance with IG D.G and additional comment 8 of IG D.G	AES-CBC:128, 192, 256-bit keys with 128-256 bits of key strength AES-CCM:128, 256-bit keys with 128 and 256 bits of key strength AES-GCM:128, 256-bit keys with 128 and 256 bits of key strength HMAC-SHA-1:112-524288 bit keys with strength of 112-	AES-CCM AES-CCM AES-CCM AES-GCM AES-GCM AES-GCM AES-GCM AES-GCM AES-GCM AES-GCM AES-CBC AES-CBC AES-CBC AES-CBC AES-CBC AES-CBC

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Type	Description	Properties	Algorithms
			256 bits HMAC-SHA2-224:112-524288 bit keys with strength of 112-256 bits HMAC-SHA2-256:112-524288 bit keys with strength of 112-256 bits HMAC-SHA2-384:112-524288 bit keys with strength of 112-256 bits HMAC-SHA2-512:112-524288 bit keys with strength of 112-256 bits	AES-CBC AES-CBC HMAC-SHA-1 HMAC-SHA-1 HMAC-SHA-1 HMAC-SHA-1 HMAC-SHA2-224 HMAC-SHA2-224 HMAC-SHA2-224 HMAC-SHA2-224 HMAC-SHA2-256 HMAC-SHA2-256 HMAC-SHA2-256 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-384 HMAC-SHA2-384 HMAC-SHA2-384 HMAC-SHA2-512 HMAC-SHA2-512 HMAC-SHA2-512 HMAC-SHA2-512 SHA-1 SHA-1 SHA-1 SHA-1 SHA2-224 SHA2-224 SHA2-224 SHA2-224 SHA2-256 SHA2-256 SHA2-256 SHA2-256 SHA2-384 SHA2-384 SHA2-384 SHA2-384 SHA2-512 SHA2-512 SHA2-512 SHA2-512
EC Diffie-Hellman	KAS-Full	EC Diffie-Hellman compliant with scenario 2(2) of IG D.F	KAS-ECC-SSC Sp800-56Ar3:P-256, P-384, P-521 elliptic curves with 128-256 bits of strength KDF TLS (CVL):TLS derived secret with 112 to 256 bits of key strength	KAS-ECC-SSC Sp800-56Ar3 KDF TLS TLS v1.2 KDF RFC7627 KDA HKDF Sp800-56Cr1

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Type	Description	Properties	Algorithms
			TLS v1.2 KDF RFC7627 (CVL):TLS derived secret with 112 to 256 bits of key strength KDA HKDF Sp800-56Cr1:TLS derived secret with 112 to 256 bits of key strength	
Diffie-Hellman	KAS-Full	Diffie-Hellman compliant with scenario 2(2) of IG D.F	KAS-FFC-SSC SP800-56Ar3:2048, 3072, 4096, 6144, 8192-bit keys with 112-200 bits of key strength KDF TLS (CVL):TLS derived secret with 112 to 256 bits of key strength TLS v1.2 KDF RFC7627 (CVL):TLS derived secret with 112 to 256 bits of key strength KDA HKDF Sp800-56Cr1:TLS derived secret with 112 to 256 bits of key strength	KAS-FFC-SSC Sp800-56Ar3 KDF TLS TLS v1.2 KDF RFC7627 KDA HKDF Sp800-56Cr1

Table 9: Security Function Implementations

2.7 Algorithm Specific Information

Hash Algorithms

In compliance with IG C.B, every approved hash algorithm implementation was CAVP tested and validated on all the module's operational environments. Section 2.5 of this security policy contains a table of the CAVP certificates of the approved hash functions.

For the higher-level algorithms that use the approved hash functions - Counter DRBG, ECDSA SigGen, ECDSA SigVer, HMAC, KDA HKDF Sp800-56Cr1, KDF TLS (CVL), TLS v1.2 KDF RFC7627 (CVL), PBKDF2, RSA SigGen, RSA SigVer – every implemented combination for which CAVP testing exists was CAVP

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

tested and validated on all the module's operational environments. Section 2.5 of this security policy contains a table of the CAVP certificates of these higher-level algorithms.

SHA-3

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

RSA Key Generation

In compliance with IG C.E, the module generates RSA signature keys using an approved method of FIPS 186-4: generation of random primes that are provably prime. The CAVP certificate #A3667 indicates that the RSA key generating algorithm has been tested and validated for conformance to the methods in FIPS 186-4.

RSA Signature Generation and Signature Verification

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

AES-GCM

The module implements AES GCM for being used in the TLS v1.2 and v1.3 protocols. AES GCM IV generation is compliant with [FIPS140-3_IG] IG C.H for both protocols as follows:

- For TLS v1.2, IV generation is compliant with scenario 1.a of IG C.H and [RFC5288]. The module supports acceptable AES-GCM cipher suites from section 3.3.1 of [SP800-52rev2].

- For TLS v1.3, IV generation is compliant with scenario 5 of IG C.H and [RFC8446]. The module supports acceptable AES-GCM cipher suites from section 3.3.1 of [SP800-52rev2].

The IV generated in both scenarios is only used within the context of the TLS protocol implementation. The nonce_explicit part of the IV does not exhaust the maximum number of possible values for a given session key. The design of the TLS protocol in this module implicitly ensures that the nonce_explicit, or counter portion of the IV will not exhaust all its possible values.

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

AES-XTS

The AES algorithm in XTS mode can only be used for the cryptographic protection of data on storage devices, as specified in SP800-38E. The length of a single data unit encrypted with the AES-XTS shall not exceed 2^{20} AES blocks, that is 16MB of data.

To meet the requirement stated in IG C.I, the module implements a check that ensures, before performing any cryptographic operation, that the two AES keys used in AES-XTS mode are not identical.

Key Agreement Methods

To comply with the assurances found in Section 5.6.2 of SP 800-56Ar3, the operator must use the module together with an application that implements the TLS protocol. Additionally, the module's approved

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

"Asymmetric key generation" service must be used to generate ephemeral Diffie-Hellman or EC Diffie-Hellman key pairs, or the key pairs must be obtained from another FIPS-validated module.

As part of this service, the module will internally perform the full public key validation of the generated public key. The module's shared secret computation service will internally perform the full public key validation of the peer public key, complying with Sections 5.6.2.2.1 and 5.6.2.2.2 of SP 800-56Ar3.

Key Transport Methods

Please refer to section 2.10 of this security policy.

Cryptographic Key Generation

In compliance with IG D.H, the module generates symmetric keys and seeds for asymmetric keys using the method described in section 4 example 1 of SP 800-133r2 without the use of V (direct DRBG output as described in additional comment 2 of IG D.H).

Please refer to section 2.9 for more information on the key generation methods employed by the module.

DRBGs

In compliance with IG D.L, the entropy input, DRBG seed, DRBG internal state (values of V and Key) are considered CSPs.

The DRBG internal state is contained within the DRBG mechanism boundary and is not accessible by other mechanisms.

PBKDF2

The module provides password-based key derivation (PBKDF), compliant with SP800-132 and IG D.N. The module supports option 1a from section 5.4 of SP800-132, in which the Master Key (MK) or a segment of it is used directly as the Data Protection Key (DPK).

In accordance with SP800-132, 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 112 bits or more (this is verified by the module to determine the service is approved).
- A portion of the salt, with a length of at least 128 bits (this is verified by the module to determine the service is approved), shall be generated randomly using the SP800-90Arev1 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 (this is verified by the module to determine the service is approved).
- 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 8 characters (this is verified by the module to determine the service is approved), and shall consist of lower-case, upper-case, and numeric characters. The probability of guessing the value is estimated to be 1/62^{8} = 10^{-14}, which is less than 2^{-112}. If the password consists of only digits (worst case), the probability of guessing the value is estimated to be 10^{-8} which is less than 2^{-112}.

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

The requirements of input parameters (derived key length, salt length, iteration count and password length) are verified by the module when providing the service indicator.

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

TLS v1.2 KDF

In compliance with IG D.Q, the module supports the TLS 1.2 KDF with the extended master secret: TLS v1.2 KDF RFC7627 (CVL).

2.8 RBG and Entropy

Cert Number	Vendor Name
E60	Canonical

Table 10: Entropy Certificates

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
Userspace CPU Time Jitter RNG Entropy Source	Non-Physical	Ubuntu 22.04 LTS (Jammy Jellyfish) on Supermicro SYS-1019P-WTR on Intel® Xeon® Gold 6226; Ubuntu 22.04 LTS (Jammy Jellyfish) on Amazon Web Services (AWS) c6g.metal on AWS Graviton2; Ubuntu 22.04 LTS (Jammy Jellyfish) on IBM z15 on z15	256	256	LFSR

Table 11: Entropy Sources

The module implements CTR_DRBG with AES-256, according to SP800-90Arev1, without a derivation function and without prediction resistance. The module uses an SP800-90B-compliant entropy source as specified above. This entropy source is located within the physical perimeter, but outside of the cryptographic boundary of the module. The public use document of the entropy source is found at:

https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/entropy/E60_PublicUse.pdf

The module obtains 384 bits from the entropy source to seed the DRBG, and 256 bits to reseed it, sufficient to provide a DRBG with 256 bits of security strength. The largest key strength generated by the module is 256 bits.

2.9 Key Generation

The module generates symmetric keys and seeds for asymmetric keys using the method described in section 4 example 1 of SP 800-133r2 without the use of V (direct output of DRBG as described in additional comment 2 of IG D.H).

The module generates the following keys:

- RSA (asymmetric): 2048, 3072, 4096-bit keys with 112-149 bits of key strength
- ECDSA (asymmetric): P-256, P-384, P-521 elliptic curves with 128-256 bits of key strength

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

- ECDH (asymmetric): P-256, P-384, P-521 elliptic curves with 128-256 bits of key strength
- Safe Primes (asymmetric): 2048, 3072, 4096, 6144, 8192-bit keys with 112-200 bits of key strength
- AES (symmetric): 128, 192, 256-bit keys with 128-256 bits of key strength
- HMAC (symmetric): 112-524288 bit keys with 112-256 bits of key strength

The generation of RSA keys is compliant with section B.3.2 of FIPS 186-4 (provable primes).

The generation of ECDSA keys is compliant with section B.4.2 of FIPS186-4 (testing candidates).

The generation of EC Diffie-Hellman keys is performed using the ECDSA key generation method, which is compliant with FIPS 186-4 and SP 800-56Ar3.

The generation of Diffie-Hellman keys is compliant with SP 800-56Ar3 (testing candidates). The module generates keys using safe primes defined in RFC7919 and RFC3526.

Additionally, the module implements the following key derivation methods:

- KDF TLS (CVL), compliant with SP800-135r1: derivation of secret keys in the context of TLS 1.0/1.1
- TLS v1.2 KDF RFC7627 (CVL), compliant with SP800-135r1: derivation of secret keys in the context of TLS 1.2
- KDA HKDF Sp800-56Cr1, compliant with SP800-56Cr1: derivation of secret keys in the context of SP800-56Ar3 key agreement schemes
- PBKDF2, compliant with option 1a of SP800-132: derivation of keys for use in storage applications

2.10 Key Establishment

Key Agreement

The module provides Diffie-Hellman and EC Diffie-Hellman shared secret computation compliant with SP800-56Arev3, in accordance with scenario 2 (1) of IG D.F and used as part of the TLS protocol key exchange in accordance with scenario 2 (2) of IG D.F; that is, the shared secret computation (KAS-FFC-SSC and KAS-ECC-SSC) followed by the derivation of the keying material using KDF TLS (CVL), TLS v1.2 KDF RFC7627 (CVL), or KDA HKDF Sp800-56Cr1. The Diffie-Hellman shared secret computation, EC Diffie-Hellman shared secret computation, KDF TLS (CVL), TLS v1.2 KDF RFC7627 (CVL), and KDA HKDF Sp800-56Cr1 have been CAVP tested.

The EC Diffie-Hellman shared secret computation uses the Ephemeral Unified Model. The module supports EC Diffie-Hellman shared secret computation with P-256, P-384, and P-521 curves which have a security strength of 128-256 bits.

The Diffie-Hellman shared secret computation uses the DH Ephemeral scheme. The module supports Diffie-Hellman shared secret computation with the MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, and ffdhe8192 groups which have a security strength of 112-200 bits.

Key Transport

The module provides key wrapping (KTS), compliant with IG D.G, using AES-CCM, AES-GCM, and AES-CBC with HMAC, used in the context of the TLS protocol cipher suites with 128-bit or 256-bit keys, with strengths of 128 bits and 256 bits respectively. When using AES-CBC with HMAC, the entire wrapped message is authenticated. AES-CCM, AES-GCM, AES-GCM, and HMAC have been tested and validated by the CAVP and the algorithms' certificate numbers are in section 2.5 of the security policy.

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

2.11 Industry Protocols

The TLS protocol implementation provides both server and client sides. To operate in the approved mode, digital certificates used for server and client authentication shall comply with the restrictions of key size and message digest algorithms imposed by SP800-131Arev2.

No parts of the TLS protocol, other than the approved cryptographic algorithms and the KDFs, have been tested by the CAVP and CMVP.

2.12 Additional Information

N/A

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	API input parameters, kernel I/O network or files on filesystem, TLS protocol input messages.
N/A	Data Output	API output parameters, kernel I/O network or files on filesystem, TLS protocol output messages.
N/A	Control Input	API function calls, API input parameters for control.
N/A	Status Output	API return codes, API output parameters for status output.

Table 12: Ports and Interfaces

The module does not have a control output interface.

3.2 Trusted Channel Specification

N/A

3.3 Control Interface Not Inhibited

N/A

3.4 Additional Information

N/A

© 2024 Canonical Ltd. / 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 Authentication Methods

N/A for this module.

4.2 Roles

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

Table 13: Roles

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Symmetric key generation	Generate AES and HMAC key	gnutls_fips140_get_operation_state() returns GNUTLS_FIPS140_OP_APPROVED	Key size	Key	Deterministic random bit generation	Crypto Officer - AES key: G,R - HMAC key: G,R
Symmetric encryption	Perform AES encryption	gnutls_fips140_get_operation_state() returns GNUTLS_FIPS140_OP_APPROVED	Key, IV (for AEAD), Plaintext	Ciphertext, MAC tag (for AEAD)	Symmetric encryption	Crypto Officer - AES key: W,E
Symmetric decryption	Perform AES decryption	gnutls_fips140_get_operation_state() returns GNUTLS_FIPS140_OP_APPROVED	Key, IV (for AEAD), Ciphertext, , MAC tag (for AEAD)	Plaintext	Symmetric decryption	Crypto Officer - AES key: W,E
Asymmetric key generation	Generate RSA, DH, or ECDSA/ECDH key pairs	gnutls_fips140_get_operation_state() returns GNUTLS_FIPS140_OP_APPROVED	RSA key size, Elliptic Curve, or Safe prime group	Key pair	Asymmetric key generation	Crypto Officer - RSA public key: G,R - RSA private key: G,R - ECDSA public key: G,R - ECDSA private key: G,R - EC Diffie-

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Hellman public key: G,R - EC Diffie-Hellman private key: G,R - Diffie-Hellman public key: G,R - Diffie-Hellman private key: G,R - Intermediate key generation value: G,E,Z
Digital signature generation	Generate RSA or ECDSA signature	gnutls_fips140_get_operation_state() returns GNUTLS_FIPS140_OP_APPROVED	Message, hash algorithm, private key	Digital signature	Digital signature generation	Crypto Officer - RSA private key: W,E - ECDSA private key: W,E
Digital signature verification	Verify RSA or ECDSA signature	gnutls_fips140_get_operation_state() returns GNUTLS_FIPS140_OP_APPROVED	Message, signature, hash algorithm, public key	Verification result (success/failure)	Digital signature verification	Crypto Officer - RSA public key: W,E - ECDSA public key: W,E
Public key verification	Verify ECDSA public key	gnutls_fips140_get_operation_state() returns GNUTLS_FIPS140_OP_APPROVED	ECDSA public key	Verification result (success/failure)	Public key verification	Crypto Officer - ECDSA public key: W,E
Random number generation	Generate random bitstrings	gnutls_fips140_get_operation_state() returns GNUTLS_FIPS140_OP_APPROVED	Number of bits	Random bytes	Deterministic random bit generation	Crypto Officer - Entropy input: W,E - DRBG seed: G,E - DRBG internal state: V

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						value, key: G,E
Message digest	Compute SHA hashes	gnutls_fips140_get_operation_state() returns GNUTLS_FIPS140_OP_APPROVED	Message	Hash digest	Message digest	Crypto Officer
Message authentication code (MAC)	Compute AES-based CMAC or AES-based GMAC or HMAC	gnutls_fips140_get_operation_state() returns GNUTLS_FIPS140_OP_APPROVED	Message, HMAC key or AES key	Message authentication code (MAC)	Message authentication code (MAC)	Crypto Officer - AES key: W,E - HMAC key: W,E
Diffie-Hellman shared secret computation	Perform DH shared secret computation	gnutls_fips140_get_operation_state() returns GNUTLS_FIPS140_OP_APPROVED	Diffie-Hellman private key, Diffie-Hellman public key from peer	Shared secret	(Diffie-Hellman) shared secret computation	Crypto Officer - Diffie-Hellman public key: W,E - Diffie-Hellman private key: W,E - Diffie-Hellman Shared secret: G,R
EC Diffie-Hellman shared secret computation	Perform ECDH shared secret computation	gnutls_fips140_get_operation_state() returns GNUTLS_FIPS140_OP_APPROVED	EC Diffie-Hellman private key, EC Diffie-Hellman public key from peer	Shared secret	(EC Diffie-Hellman) shared secret computation	Crypto Officer - EC Diffie-Hellman public key: W,E - EC Diffie-Hellman private key: W,E - EC Diffie-Hellman Shared secret: G,R
HMAC-based key derivation	Perform key derivation	gnutls_fips140_get_operation_state() returns GNUTLS_FIPS140_OP_APPROVED	Diffie-Hellman shared secret or EC Diffie-	HKDF derived key	Key derivation	Crypto Officer - Diffie-Hellman Shared secret: W,E

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
			Hellman shared secret			- EC Diffie-Hellman Shared secret: W,E - HKDF derived key: G,R
Transport Layer Security (TLS) network protocol	Provided supported cipher suites in approved mode	gnutls_fips140_get_operation_state() returns GNUTLS_FIPS140_OP_APPROVED	Cipher-suites (see Appendix A for the complete list of valid cipher suites), Digital Certificate, Public and Private Keys, Application Data	Return codes and/or log messages, Application data	Symmetric encryption Symmetric decryption Message authentication code (MAC) Digital signature generation Digital signature verification (EC Diffie-Hellman) shared secret computation (Diffie-Hellman) shared secret computation Key derivation Key wrapping	Crypto Officer - RSA public key: W,E - ECDSA public key: W,E - Diffie-Hellman public key: W,E - Diffie-Hellman private key: W,E - EC Diffie-Hellman public key: W,E - EC Diffie-Hellman private key: W,E - TLS pre-master secret: G,E - TLS derived secret: G,E - TLS master secret: G,E - RSA private key: W,E - ECDSA private key: W,E

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Show status	Show module status	Implicit (always approved)	None	Return codes and/or log messages	None	Crypto Officer
Zeroization	Zeroize SSPs	Implicit (always approved)	Context containing SSPs	None	None	Crypto Officer - AES key: Z - HMAC key: Z - RSA public key: Z - RSA private key: Z - ECDSA public key: Z - ECDSA private key: Z - Diffie-Hellman public key: Z - Diffie-Hellman private key: Z - EC Diffie-Hellman public key: Z - EC Diffie-Hellman private key: Z - Diffie-Hellman Shared secret: Z - EC Diffie-Hellman Shared secret: Z - PBKDF password or passphrase: Z

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- PBKDF derived key: Z - Entropy input: Z - DRBG seed: Z - DRBG internal state: V value, key: Z - TLS pre-master secret: Z - TLS master secret: Z - TLS derived secret: Z - Intermediate key generation value: Z - HKDF derived key: Z
Self-tests	Perform self-tests	Implicit (always approved)	Module reset	Result of self-tests (pass/fail)	None	Crypto Officer
Show module name and version	Show module name and version	Implicit (always approved)	None	Name and version information	None	Crypto Officer
TLS key derivation	Perform key derivation	gnutls_fips140_get_operation_state() returns GNUTLS_FIPS140_OP_APPROVED	TLS pre-master secret	TLS derived secret	Key derivation	Crypto Officer - TLS pre-master secret: W,E - TLS master secret: G,E - TLS derived secret: G,R

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Password-based key derivation	Perform key derivation	gnutls_fips140_get_operation_state() returns GNUTLS_FIPS140_OP_APPROVED	PBKDF password or passphrase	PBKDF derived key	Key derivation	Crypto Officer - PBKDF password or passphrase: W,E - PBKDF derived key: G,R
Diffie-Hellman	Perform DH key agreement	gnutls_fips140_get_operation_state() returns GNUTLS_FIPS140_OP_APPROVED	Diffie-Hellman private key, Diffie-Hellman public key from peer	HKDF derived key, TLS derived secret	Diffie-Hellman	Crypto Officer - Diffie-Hellman public key: W,E - Diffie-Hellman private key: W,E - TLS derived secret: G,R - HKDF derived key: G,R
EC Diffie-Hellman	Perform ECDH key agreement	gnutls_fips140_get_operation_state() returns GNUTLS_FIPS140_OP_APPROVED	EC Diffie-Hellman private key, EC Diffie-Hellman public key from peer	HKDF derived key, TLS derived secret	EC Diffie-Hellman	Crypto Officer - EC Diffie-Hellman public key: W,E - EC Diffie-Hellman private key: W,E - TLS derived secret: G,R - HKDF derived key: G,R

Table 14: Approved Services

The “Indicator” column shows the service indicator API functions that must be used to verify the service indicator for each of the services. The function gnutls_fips140_get_operation_state() indicates GNUTLS_FIPS140_OP_NOT_APPROVED or GNUTLS_FIPS140_OP_APPROVED depending on whether the API invoked corresponds to an approved or non-approved algorithm.

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

4.4 Non-Approved Services

Name	Description	Algorithms	Role
Symmetric encryption; Symmetric decryption (non-approved)	Blowfish, Camellia, CAST, ChaCha20, DES, Salsa20, SEED, Serpent, Triple-DES, Twofish, AES-GCM (when not used in the context of the TLS protocol), GOST, RC2, RC4	Blowfish Camellia CAST ChaCha20 DES Salsa20 SEED Serpent Triple-DES Twofish AES-GCM (when not used in the context of the TLS protocol) GOST RC2, RC4	CO
Authenticated encryption; Authenticated decryption (non-approved)	Chacha20 and Poly1305	Chacha20 and Poly1305	CO
Message authentication code (non-approved)	HMAC (with GOST), HMAC (with keys smaller than 112-bits), UMAC, CMAC with Triple-DES, GMAC	HMAC (with GOST) HMAC (with keys smaller than 112-bits) UMAC CMAC with Triple-DES GMAC	CO
Message digest (non-approved)	MD2, MD4, MD5, RMD160, STREEBOG, GOST	MD2, MD4, MD5 RMD160 STREEBOG GOST	CO
Key derivation (non-approved)	PBKDF (with non-approved message digest algorithms or using input parameters not meeting requirements stated in section 2.7 of the security policy)	PBKDF (with non-approved message digest algorithms or using input parameters not meeting requirements stated in section 2.7 of the security policy)	CO
Domain parameter generation (non-approved)	DSA	DSA	CO
Public key verification (non-approved)	ECDSA (with curves other than P-256, P-384, P-512)	ECDSA (with curves other than P-256, P-384, P-512)	CO
Key generation (non-approved)	RSA (with keys smaller than 2048 bits or greater than 4096 bits), DSA, ECDSA (with curves other than P-256, P-384, P-512)	RSA (with keys smaller than 2048 bits or greater than 4096 bits) DSA ECDSA (with curves other than P-256, P-384, P-512)	CO
Digital signature verification (non-approved)	RSA (with keys smaller than 1024 bits or greater than 4096 bits and/or hash functions other than SHA2-224, SHA2-256, SHA2-384, SHA2-512), DSA, ECDSA	RSA (with keys smaller than 1024 bits or greater than 4096 bits and/or hash functions other than SHA2-	CO

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Algorithms	Role
	(with curves other than P-256, P-384, P-512 or hash functions other than SHA2-224, SHA2-256, SHA2-384, SHA2-512)	224, SHA2-256, SHA2-384, SHA2-512) DSA ECDSA (with curves other than P-256, P-384, P-512 or hash functions other than SHA2-224, SHA2-256, SHA2-384, SHA2-512)	
Digital signature generation (non-approved)	RSA (with keys smaller than 2048 bits or greater than 4096 bits and/or hash functions other than SHA2-224, SHA2-256, SHA2-384, SHA2-512), DSA, ECDSA (with curves other than P-256, P-384, P-512 or hash functions other than SHA2-224, SHA2-256, SHA2-384, SHA2-512)	RSA (with keys smaller than 2048 bits or greater than 4096 bits and/or hash functions other than SHA2-224, SHA2-256, SHA2-384, SHA2-512) ECDSA (with curves other than P-256, P-384, P-512 or hash functions other than SHA2-224, SHA2-256, SHA2-384, SHA2-512) DSA	CO
Key agreement; Shared secret computation (non-approved)	SRP, Diffie-Hellman (with domain parameters other than safe primes), EC Diffie-Hellman (with curves other than P-256, P-384, P-512)	SRP Diffie-Hellman (with domain parameters other than safe primes) EC Diffie-Hellman (with curves other than P-256, P-384, P-512)	CO
Key encapsulation; Key un-encapsulation (non-approved)	RSA (encapsulation and un-encapsulation with any key sizes)	RSA (encapsulation and un-encapsulation with any key sizes)	CO
Random number generation (non-approved)	Yarrow	Yarrow	CO

Table 15: Non-Approved Services

4.5 External Software/Firmware Loaded

The module does not support the loading of external software/firmware.

4.6 Bypass Actions and Status

N/A

4.7 Cryptographic Output Actions and Status

N/A

4.8 Additional Information

N/A

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

5 Software/Firmware Security

5.1 Integrity Techniques

The integrity of the module is verified by comparing an HMAC-SHA2-256 value calculated at run time with the HMAC value stored in the .hmac file that was computed at build time for each software component of the module listed in section 2. If the HMAC values do not match, the test fails, and the module enters the error state.

5.2 Initiate on Demand

The pre-operational integrity self-test can be initiated on demand by calling the Self-Test service (via the gnutls_fips140_run_self_tests() function) or by powering-off and reloading the module. During the execution of the on-demand integrity self-test, services are not available, and no data output is possible.

5.3 Open-Source Parameters

N/A

5.4 Additional Information

N/A

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

How Requirements are Satisfied:

N/A

6.2 Configuration Settings and Restrictions

The module shall be installed as stated in section 11. The operating system provides process isolation and memory protection mechanisms that ensure appropriate separation for memory access among the processes on the system. Each process has control over its own data and uncontrolled access to the data of other processes is prevented.

6.3 Additional Information

N/A

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

7 Physical Security

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

7.1 Mechanisms and Actions Required

N/A for this module.

N/A

7.2 User Placed Tamper Seals

Number:

Placement:

Surface Preparation:

Operator Responsible for Securing Unused Seals:

Part Numbers:

N/A

7.3 Filler Panels

N/A

7.4 Fault Induction Mitigation

N/A

7.5 EFP/EFT Information

Temp/Voltage Type	Temperature or Voltage	EFP or EFT	Result
LowTemperature			
HighTemperature			
LowVoltage			
HighVoltage			

Table 16: EFP/EFT Information

7.6 Hardness Testing Temperature Ranges

Temperature Type	Temperature
LowTemperature	
HighTemperature	

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Table 17: Hardness Testing Temperatures

7.7 Additional Information

N/A

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

8 Non-Invasive Security

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

8.1 Mitigation Techniques

N/A

8.2 Effectiveness

N/A

8.3 Additional Information

N/A

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
RAM	Temporary storage for SSPs used by the module as part of service execution.	Dynamic

Table 18: Storage Areas

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
API input parameters	Operator calling application (within TOEPP)	Cryptographic module	Plaintext	Manual	Electronic	
API output parameters	Cryptographic module	Operator calling application (within TOEPP)	Plaintext	Manual	Electronic	

Table 19: SSP Input-Output Methods

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Wipe and Free memory block allocated	Zeroizes the SSPs contained within the cipher handle.	Memory occupied by SSPs is overwritten with zeroes and then it is released, which renders the SSP values irretrievable. The completion of the zeroization routine indicates that the zeroization has been completed	To zeroize AES keys, call gnutls_cipher_deinit() or gnutls_aead_cipher_deinit(); to zeroize HMAC keys, call gnutls_hmac_deinit(); to zeroize RSA or ECDSA public/private keys, call gnutls_privkey_deinit() or gnutls_x509_privkey_deinit() or gnutls_rsa_params_deinit(); to zeroize Diffie-Hellman public/private keys, call gnutls_dh_params_deinit() or gnutls_pk_params_clear(); to zeroize EC Diffie-Hellman public/private keys, call gnutls_pk_params_clear(); to zeroize (Diffie-Hellman) Shared secret or (EC Diffie-Hellman) Shared secret, call zeroize_key(); to zeroize entropy input, DRBG seed, or DRBG internal state, call gnutls_global_deinit(); to zeroize TLS pre-master secret, TLS master secret, TLS derived secret, or PBKDF derived key, call gnutls_deinit()

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Zeroization Method	Description	Rationale	Operator Initiation
Automatic	Automatically zeroized by the module when no longer needed	Memory occupied by SSPs is overwritten with zeroes, which renders the SSP values irretrievable.	N/A
Module Reset	De-allocates the volatile memory used to store SSPs	Volatile memory used by the module is overwritten within nanoseconds when power is removed. The completion of module power-off indicates that the zeroization has been completed	By unloading and reloading the module

Table 20: SSP Zeroization Methods

All data output is inhibited when the module is performing zeroization.

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES key	Used for Symmetric encryption; Symmetric decryption; Message authentication code (MAC);	128, 192, 256 bits - 128, 192, 256 bits	Symmetric key - CSP	Deterministic random bit generation		Symmetric encryption Symmetric decryption Message authentication code (MAC)
HMAC key	Used for Message Authentication Code (MAC)	112 to 256 bits - 112 to 256 bits	Symmetric key - CSP	Deterministic random bit generation		Message authentication code (MAC)
RSA public key	Used for Digital signature verification; Transport Layer Security (TLS) network protocol	2048, 3072, 4096-bit modulus - 112, 128, 149 bits	Public key - PSP	Asymmetric key generation		Digital signature verification
RSA private key	Used for Digital signature generation; Transport Layer Security (TLS)	2048, 3072, 4096-bit modulus - 112, 128, 149 bits	Private key - CSP	Asymmetric key generation		Digital signature generation

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	network protocol					
ECDSA public key	Used for Digital signature verification; Public key verification; Transport Layer Security (TLS) network protocol	P-256, P-384, P-521 - 128, 192, 256 bits	Public key - PSP	Asymmetric key generation		Digital signature verification
ECDSA private key	Used for Digital signature generation; Transport Layer Security (TLS) network protocol	P-256, P-384, P-521 - 128, 192, 256 bits	Private key - CSP	Asymmetric key generation		Digital signature generation
Diffie-Hellman public key	Used for Shared secret computation; Transport Layer Security (TLS) network protocol	ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 - 112 to 200 bits	Public key - PSP	Asymmetric key generation		(Diffie-Hellman) shared secret computation
Diffie-Hellman private key	Used for Shared secret computation; Transport Layer Security (TLS) network protocol	ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048,	Private key - CSP	Asymmetric key generation		(Diffie-Hellman) shared secret computation

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		MODP-3072, MODP-4096, MODP-6144, MODP-8192 - 112 to 200 bits				
EC Diffie-Hellman public key	Used for Shared secret computation; Transport Layer Security (TLS) network protocol	P-256, P-384, P-521 - 128, 192, 256 bits	Public key - PSP	Asymmetric key generation		(EC Diffie-Hellman) shared secret computation
EC Diffie-Hellman private key	Used for Shared secret computation; Transport Layer Security (TLS) network protocol	P-256, P-384, P-521 - 128, 192, 256 bits	Private key - CSP	Asymmetric key generation		(EC Diffie-Hellman) shared secret computation
Diffie-Hellman Shared secret	Used for Key derivation	ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 - 112 to 200 bits	Shared secret - CSP		(Diffie-Hellman) shared secret computation	Key derivation
EC Diffie-Hellman Shared secret	Used for Key derivation	P-256, P-384, P-521 - 128, 192, 256 bits	Shared secret - CSP		(EC Diffie-Hellman) shared secret computation	Key derivation

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
PBKDF password or passphrase	Used for Key derivation	20 characters or more - N/A	Password - CSP			Key derivation
PBKDF derived key	Used for protection of storage data	112 to 256 bits - 112 to 256 bits	Symmetric key - CSP	Key derivation		
Entropy input	Used for Random number generation	256 to 384 bits - 256 bits	Entropy Input - CSP			Deterministic random bit generation
DRBG seed	Used for Random number generation	256 to 384 bits - 256 to 384 bits	Seed - CSP	Deterministic random bit generation		Deterministic random bit generation
DRBG internal state: V value, key	Used for Random number generation. This SSP is a CSP in compliance with IG D.L	V: 128 bits; Key: 256 bits - 256 bits	Internal state - CSP	Deterministic random bit generation		Deterministic random bit generation
TLS pre-master secret	Used for Transport Layer Security (TLS) network protocol	ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, P-256, P-384, P-521 - 112 to 256 bits	Shared secret - CSP		(EC Diffie-Hellman) shared secret computation (Diffie-Hellman) shared secret computation	Key derivation
TLS master secret	Used for Transport Layer Security (TLS) network protocol	384 bits - 112 to 256 bits	Intermediate secret value - CSP	Key derivation		Key derivation

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
TLS derived secret	Used in Transport Layer Security (TLS) network protocol	112 to 256 bits - 112 to 256 bits	Derived key - CSP	Key derivation		
Intermediate key generation value	Used in Asymmetric key generation	256 to 8192 bits - 112 to 256 bits	Intermediate key generation value - CSP	Asymmetric key generation		Asymmetric key generation
HKDF derived key	Key derived from HKDF	112 to 256 bits - 112 to 256 bits	Symmetric key - CSP	Key derivation		

Table 21: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES key	API input parameters API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Wipe and Free memory block allocated Module Reset	DRBG internal state: V value, key:Generated from
HMAC key	API input parameters API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Wipe and Free memory block allocated Module Reset	DRBG internal state: V value, key:Generated from
RSA public key	API input parameters API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Wipe and Free memory block allocated Module Reset	RSA private key:Paired With Intermediate key generation value:Generated from
RSA private key	API input parameters API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Wipe and Free memory block allocated Module Reset	RSA public key:Paired With Intermediate key generation value:Generated from
ECDSA public key	API input parameters API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Wipe and Free memory block allocated Module Reset	ECDSA private key:Paired With Intermediate key generation value:Generated from

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
ECDSA private key	API input parameters API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Wipe and Free memory block allocated Module Reset	ECDSA public key:Paired With Intermediate key generation value:Generated from
Diffie-Hellman public key	API input parameters API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Wipe and Free memory block allocated Module Reset	Diffie-Hellman Shared secret:Used to compute Diffie-Hellman private key:Paired With Intermediate key generation value:Generated from
Diffie-Hellman private key	API input parameters API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Wipe and Free memory block allocated Module Reset	Diffie-Hellman Shared secret:Used to compute Diffie-Hellman public key:Paired With Intermediate key generation value:Generated from
EC Diffie-Hellman public key	API input parameters API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Wipe and Free memory block allocated Module Reset	EC Diffie-Hellman Shared secret:Used to compute EC Diffie-Hellman private key:Paired With Intermediate key generation value:Generated from
EC Diffie-Hellman private key	API input parameters API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Wipe and Free memory block allocated Module Reset	EC Diffie-Hellman Shared secret: Used to compute EC Diffie-Hellman public key:Paired With Intermediate key generation value:Generated from
Diffie-Hellman Shared secret	API input parameters API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Wipe and Free memory block allocated Module Reset	Diffie-Hellman public key:Computed using Diffie-Hellman private key:Computed using HKDF derived key:Used to derive
EC Diffie-Hellman Shared secret	API input parameters API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Wipe and Free memory block allocated	EC Diffie-Hellman public key:Computed using EC Diffie-Hellman private key:Computed using

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
				Module Reset	HKDF derived key:Used to derive
PBKDF password or passphrase	API input parameters	RAM:Plaintext	From service invocation to service completion	Automatic	PBKDF derived key:Used to derive
PBKDF derived key	API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Wipe and Free memory block allocated Module Reset	PBKDF password or passphrase:Derived From
Entropy input		RAM:Plaintext	Until explicitly zeroized by operator	Automatic Module Reset	DRBG seed:Used to compute
DRBG seed		RAM:Plaintext	Until explicitly zeroized by operator	Automatic Module Reset	Entropy input:Computed from DRBG internal state: V value, key:Used to compute
DRBG internal state: V value, key		RAM:Plaintext	Until explicitly zeroized by operator	Wipe and Free memory block allocated Module Reset	DRBG seed:Computed from
TLS pre-master secret		RAM:Plaintext	Until explicitly zeroized by operator	Wipe and Free memory block allocated Module Reset	TLS master secret:Used to compute
TLS master secret		RAM:Plaintext	Until explicitly zeroized by operator	Wipe and Free memory block allocated Module Reset	TLS pre-master secret:Computed from TLS derived secret:Used to derive
TLS derived secret	API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Wipe and Free memory block allocated Module Reset	TLS master secret:Derived From
Intermediate key		RAM:Plaintext	From service invocation	Automatic	RSA public key:Intermediate value obtained during

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
generation value			to service completion		generated of RSA private key:Intermediate value obtained during generated of ECDSA public key:Intermediate value obtained during generated of ECDSA private key:Intermediate value obtained during generated of Diffie-Hellman public key:Intermediate value obtained during generated of Diffie-Hellman private key:Intermediate value obtained during generated of EC Diffie-Hellman public key:Intermediate value obtained during generated of EC Diffie-Hellman private key:Intermediate value obtained during generated of
HKDF derived key	API output parameters	RAM:Plaintext	Until explicitly zeroized by operator	Wipe and Free memory block allocated Module Reset	Diffie-Hellman Shared secret:Derived From EC Diffie-Hellman Shared secret:Derived From

Table 22: SSP Table 2

9.5 Transitions

N/A

9.6 Additional Information

N/A

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

10 Self-Tests

The module performs the pre-operational self-test and cryptographic algorithm self-tests (CASTs) automatically when the module is loaded into memory. The module's services are not available for use and data input and output are inhibited until the pre-operational tests and CASTs are completed successfully. If any of the pre-operational integrity self-tests or CASTs fail, an error message is returned and the module transitions to the error state.

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A3665)	SHA2-256	KAT	SW/FW Integrity	Module becomes operational and services are available for use	MAC tag computation and verification
HMAC-SHA2-256 (A3667)	SHA2-256	KAT	SW/FW Integrity	Module becomes operational and services are available for use	MAC tag computation and verification
HMAC-SHA2-256 (A3714)	SHA2-256	KAT	SW/FW Integrity	Module becomes operational and services are available for use	MAC tag computation and verification
HMAC-SHA2-256 (A3710)	SHA2-256	KAT	SW/FW Integrity	Module becomes operational and services are available for use	MAC tag computation and verification

Table 23: Pre-Operational Self-Tests

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC (A3665)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	On module initialization
AES-CBC (A3667)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	On module initialization
AES-CBC (A3708)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	On module initialization
AES-CBC (A3709)	256-bit key	KAT	CAST	Module becomes operational and services	Encryption, Decryption	On module initialization

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				are available for use		
AES-CBC (A3711)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	On module initialization
AES-CBC (A3712)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	On module initialization
AES-CBC (A3713)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	On module initialization
AES-CBC (A3714)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	On module initialization
AES-CFB8 (A3670)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	On module initialization
AES-CFB8 (A3716)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	On module initialization
AES-CFB8 (A3717)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	On module initialization
AES-GCM (A3665)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	On module initialization
AES-GCM (A3667)	256-bit key	KAT	CAST	Module becomes operational	Encryption, Decryption	On module initialization

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				and services are available for use		
AES-GCM (A3708)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	On module initialization
AES-GCM (A3709)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	On module initialization
AES-GCM (A3711)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	On module initialization
AES-GCM (A3712)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	On module initialization
AES-GCM (A3713)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	On module initialization
AES-XTS Testing Revision 2.0 (A3668)	256-bit key	KAT	CAST	Module becomes operational and services are available for use	Encryption, Decryption	On module initialization
KAS-FFC-SSC Sp800-56Ar3 (A3667)	3072-bit key and safe prime ffdhe3072	KAT	CAST	Module becomes operational and services are available for use	Primitive “Z” computation	On module initialization
Counter DRBG (A3667)	256-bit key without DF, without PR	KAT	CAST	Module becomes operational and services are available for use	Random bit generation	On module initialization
KAS-ECC-SSC Sp800-	P-256	KAT	CAST	Module becomes	Primitive “Z” Computation	On module initialization

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
56Ar3 (A3667)				operational and services are available for use		
ECDSA SigGen (FIPS186-4) (A3667)	P-256 using SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Signature generation	On module initialization
ECDSA SigVer (FIPS186-4) (A3667)	P-256 using SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Signature verification	On module initialization
KDA HKDF Sp800-56Cr1 (A3666)	HMAC-SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Key derivation	On module initialization
HMAC-SHA-1 (A3665)	SHA-1	KAT	CAST	Module becomes operational and services are available for use	MAC generation	On module initialization
HMAC-SHA-1 (A3667)	SHA-1	KAT	CAST	Module becomes operational and services are available for use	MAC generation	On module initialization
HMAC-SHA-1 (A3714)	SHA-1	KAT	CAST	Module becomes operational and services are available for use	MAC generation	On module initialization
HMAC-SHA-1 (A3710)	SHA-1	KAT	CAST	Module becomes operational and services are available for use	MAC generation	On module initialization
HMAC-SHA2-224 (A3665)	SHA-224	KAT	CAST	Module becomes operational and services are available for use	MAC generation	On module initialization

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA2-224 (A3667)	SHA-224	KAT	CAST	Module becomes operational and services are available for use	MAC generation	On module initialization
HMAC-SHA2-224 (A3714)	SHA-224	KAT	CAST	Module becomes operational and services are available for use	MAC generation	On module initialization
HMAC-SHA2-224 (A3710)	SHA-224	KAT	CAST	Module becomes operational and services are available for use	MAC generation	On module initialization
HMAC-SHA2-256 (A3665)	SHA-256	KAT	CAST	Module becomes operational and services are available for use	MAC generation	On module initialization
HMAC-SHA2-256 (A3667)	SHA-256	KAT	CAST	Module becomes operational and services are available for use	MAC generation	On module initialization
HMAC-SHA2-256 (A3714)	SHA-256	KAT	CAST	Module becomes operational and services are available for use	MAC generation	On module initialization
HMAC-SHA2-256 (A3710)	SHA-256	KAT	CAST	Module becomes operational and services are available for use	MAC generation	On module initialization
HMAC-SHA2-384 (A3665)	SHA-384	KAT	CAST	Module becomes operational and services are available for use	MAC generation	On module initialization
HMAC-SHA2-384 (A3667)	SHA-384	KAT	CAST	Module becomes operational and services	MAC generation	On module initialization

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				are available for use		
HMAC-SHA2-384 (A3714)	SHA-384	KAT	CAST	Module becomes operational and services are available for use	MAC generation	On module initialization
HMAC-SHA2-384 (A3710)	SHA-384	KAT	CAST	Module becomes operational and services are available for use	MAC generation	On module initialization
HMAC-SHA2-512 (A3665)	SHA-512	KAT	CAST	Module becomes operational and services are available for use	MAC generation	On module initialization
HMAC-SHA2-512 (A3667)	SHA-512	KAT	CAST	Module becomes operational and services are available for use	MAC generation	On module initialization
HMAC-SHA2-512 (A3714)	SHA-512	KAT	CAST	Module becomes operational and services are available for use	MAC generation	On module initialization
HMAC-SHA2-512 (A3710)	SHA-512	KAT	CAST	Module becomes operational and services are available for use	MAC generation	On module initialization
PBKDF (A3667)	HMAC-SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Key derivation	On module initialization
RSA SigGen (FIPS186-4) (A3667)	2048-bit key using SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Signature generation	On module initialization
RSA SigVer (FIPS186-4) (A3667)	2048-bit key using SHA2-256	KAT	CAST	Module becomes operational	Signature verification	On module initialization

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
				and services are available for use		
SHA3-224 (A3669)	SHA3-224	KAT	CAST	Module becomes operational and services are available for use	Hash digest	On module initialization
SHA3-224 (A3715)	SHA3-224	KAT	CAST	Module becomes operational and services are available for use	Hash digest	On module initialization
SHA3-256 (A3669)	SHA3-256	KAT	CAST	Module becomes operational and services are available for use	Hash digest	On module initialization
SHA3-256 (A3715)	SHA3-256	KAT	CAST	Module becomes operational and services are available for use	Hash digest	On module initialization
SHA3-384 (A3669)	SHA3-384	KAT	CAST	Module becomes operational and services are available for use	Hash digest	On module initialization
SHA3-384 (A3715)	SHA3-384	KAT	CAST	Module becomes operational and services are available for use	Hash digest	On module initialization
SHA3-512 (A3669)	SHA3-512	KAT	CAST	Module becomes operational and services are available for use	Hash digest	On module initialization
SHA3-512 (A3715)	SHA3-512	KAT	CAST	Module becomes operational and services are available for use	Hash digest	On module initialization

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
TLS v1.2 KDF RFC7627 (A3667)	HMAC-SHA2-256	KAT	CAST	Module becomes operational and services are available for use	Key derivation, using extended master secret	On module initialization
ECDSA KeyGen (FIPS186-4) (A3667)	SHA2-256	PCT	PCT	Module remains operational and services remain available for use	Signature generation and verification	After every ECDSA and ECDH key generation
RSA KeyGen (FIPS186-4) (A3667)	SHA2-256	PCT	PCT	Module remains operational and services remain available for use	Signature generation and verification	After every RSA key generation
Safe Primes Key Generation (A3667)	Section 5.6.2.1.4 of SP800-56Arev3	PCT	PCT	Module remains operational and services remain available for use	Modular exponentiation with private key	After every Diffie-Hellman key generation

Table 24: Conditional Self-Tests

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A3665)	KAT	SW/FW Integrity	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
HMAC-SHA2-256 (A3667)	KAT	SW/FW Integrity	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
HMAC-SHA2-256 (A3714)	KAT	SW/FW Integrity	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
HMAC-SHA2-256 (A3710)	KAT	SW/FW Integrity	On demand	Manually by unloading then loading module, or by calling the

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
				gnutls_fips140_run_self_tests() function

Table 25: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CBC (A3665)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
AES-CBC (A3667)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
AES-CBC (A3708)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
AES-CBC (A3709)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
AES-CBC (A3711)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
AES-CBC (A3712)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
AES-CBC (A3713)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
AES-CBC (A3714)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
AES-CFB8 (A3670)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CFB8 (A3716)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
AES-CFB8 (A3717)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
AES-GCM (A3665)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
AES-GCM (A3667)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
AES-GCM (A3708)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
AES-GCM (A3709)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
AES-GCM (A3711)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
AES-GCM (A3712)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
AES-GCM (A3713)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
AES-XTS Testing Revision 2.0 (A3668)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
KAS-FFC-SSC Sp800-56Ar3 (A3667)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
				gnutls_fips140_run_self_tests() function
Counter DRBG (A3667)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
KAS-ECC-SSC Sp800-56Ar3 (A3667)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
ECDSA SigGen (FIPS186-4) (A3667)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
ECDSA SigVer (FIPS186-4) (A3667)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
KDA HKDF Sp800-56Cr1 (A3666)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
HMAC-SHA-1 (A3665)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
HMAC-SHA-1 (A3667)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
HMAC-SHA-1 (A3714)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
HMAC-SHA-1 (A3710)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
HMAC-SHA2-224 (A3665)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-224 (A3667)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
HMAC-SHA2-224 (A3714)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
HMAC-SHA2-224 (A3710)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
HMAC-SHA2-256 (A3665)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
HMAC-SHA2-256 (A3667)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
HMAC-SHA2-256 (A3714)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
HMAC-SHA2-256 (A3710)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
HMAC-SHA2-384 (A3665)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
HMAC-SHA2-384 (A3667)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
HMAC-SHA2-384 (A3714)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
HMAC-SHA2-384 (A3710)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
				gnutls_fips140_run_self_tests() function
HMAC-SHA2-512 (A3665)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
HMAC-SHA2-512 (A3667)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
HMAC-SHA2-512 (A3714)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
HMAC-SHA2-512 (A3710)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
PBKDF (A3667)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
RSA SigGen (FIPS186-4) (A3667)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
RSA SigVer (FIPS186-4) (A3667)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
SHA3-224 (A3669)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
SHA3-224 (A3715)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
SHA3-256 (A3669)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
SHA3-256 (A3715)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
SHA3-384 (A3669)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
SHA3-384 (A3715)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
SHA3-512 (A3669)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
SHA3-512 (A3715)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
TLS v1.2 KDF RFC7627 (A3667)	KAT	CAST	On demand	Manually by unloading then loading module, or by calling the gnutls_fips140_run_self_tests() function
ECDSA KeyGen (FIPS186-4) (A3667)	PCT	PCT	On demand	By calling the "Asymmetric key Generation" service
RSA KeyGen (FIPS186-4) (A3667)	PCT	PCT	On demand	By calling the "Asymmetric key Generation" service
Safe Primes Key Generation (A3667)	PCT	PCT	On demand	By calling the "Asymmetric key Generation" service

Table 26: Conditional Periodic Information

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Error State	Prevents any cryptographic related operations and data output	When the integrity test or KAT (not the DRBG KAT) fail; When the DRBG	Restarting the module	GNUTLS_E_SELF_TEST_ERROR (-400); GNUTLS_E_RANDOM_FAILED (-206); GNUTLS_E_PK_GENERATION_ERROR (-403); GNUTLS_E_LIB_IN_ERROR_STATE (-402)

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Name	Description	Conditions	Recovery Method	Indicator
		KAT fails; When a newly generated RSA, ECDSA, Diffie-Hellman or EC Diffie-Hellman key pair fails the PCT; When the module is in error state and caller requests cryptographic operations;		

Table 27: Error States

The calling application can obtain the module state by calling the gnutls_fips140_get_operation_state() API function. The function returns GNUTLS_FIPS140_OP_ERROR if the module is in the Error state.

10.5 Operator Initiation of Self-Tests

The operator can initiate the pre-operational integrity self-test and cryptographic algorithm self-tests by calling the Self-Test service (via the gnutls_fips140_run_self_tests() function) or by powering-off and reloading the module. The operator can initiate a pairwise consistency self-test by calling the "Asymmetric key generation" service. During the execution of the pre-operational integrity self-test and cryptographic algorithm self-tests, services are not available, and no data output is possible.

10.6 Additional Information

N/A

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

11.1.1 Configuration of the Operating Environment

The module needs to be set to run in the FIPS validated operational environment. This can be enabled automatically via the Ubuntu Advantage tool after attaching your subscription.

(1) To install the tool, type the following commands:

$ sudo apt update

$ sudo apt install ubuntu-advantage-tools

(2) To activate the Ubuntu Pro subscription run:

$ sudo pro attach <your_pro_token>

(3) To enable the FIPS validated operational environment run:

$ sudo pro enable fips

(4) To verify that the FIPS validated operational environment is enabled run:

$ sudo pro status

The pro client will install the necessary packages that are part of the FIPS validated operational environment, including the kernel and the bootloader. After this step you MUST reboot to enter the FIPS validated operational environment. The reboot will boot into the kernel of the FIPS validated operational environment and create the /proc/sys/crypto/fips_enabled entry which tells the FIPS certified modules to run in the approved mode of operation. If you do not reboot after installing and configuring the bootloader, you will not be in the FIPS validated operational environment.

To verify that the FIPS validated operational environment is enabled after the reboot check the /proc/sys/crypto/fips_enabled file and ensure it is set to 1. If it is set to 0, the FIPS modules will not run in the approved mode of operation. If the file is missing, the correct kernel (which is part of the FIPS validated operational environment) is not installed. You can verify that the FIPS validated operational environment has been properly enabled with the pro status command.

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

If the module is not installed, initialized, and configured according to this section, the module is in a non-compliant state. If the module is in a non-compliant state, it can be placed into the compliant state by un-initializing and uninstalling the module and then installing, initializing, and configuring the module according to this section.

11.1.2 Delivery of the Module

On the Supermicro SYS-1019P-WTR hardware platform with the Intel Xeon Gold 6226 processor, the module is delivered through the following Ubuntu packages:

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

libgnutls30_3.7.3-4ubuntu1.2+Fips1.1_amd64.deb
libnettle8_3.7.3-1ubuntu0.1~Fips1_amd64.deb
libhogweed6_3.7.3-1ubuntu0.1~Fips1_amd64.deb
libgmp10_6.2.1+dfsg-3ubuntu1+Fips1_amd64.deb

On the Amazon Web Services (AWS) c6g.metal hardware platform with the AWS Graviton2 processor, the module is delivered through the following Ubuntu packages:

libgnutls30_3.7.3-4ubuntu1.2+Fips1.1_arm64.deb
libnettle8_3.7.3-1ubuntu0.1~Fips1_arm64.deb
libhogweed6_3.7.3-1ubuntu0.1~Fips1_arm64.deb
libgmp10_6.2.1+dfsg-3ubuntu1+Fips1_arm64.deb

On the IBM z15 hardware platform with the z15 processor, the module is delivered through the following Ubuntu packages:

libgnutls30_3.7.3-4ubuntu1.2+Fips1.1_s390x.deb
libnettle8_3.7.3-1ubuntu0.1~Fips1_s390x.deb
libhogweed6_3.7.3-1ubuntu0.1~Fips1_s390x.deb
libgmp10_6.2.1+dfsg-3ubuntu1+Fips1_s390x.deb

11.1.3 Installation of the Module

After the operating environment has been configured according to the instructions of section 11.1.1, the Crypto Officer can install the Ubuntu packages containing the module using the Advanced Package Tool (APT) with the following commands:

$ sudo apt-get install libgnutls30=3.7.3-4ubuntu1.2+Fips1.1

$ sudo apt-get install libgmp10=2:6.2.1+dfsg-3ubuntu1+Fips1

$ sudo apt-get install libhogweed6=3.7.3-1ubuntu0.1~Fips1

$ sudo apt-get install libnettle8=3.7.3-1ubuntu0.1~Fips1

All the Ubuntu packages are associated with hashes for integrity check. The integrity of the Ubuntu package is automatically verified by the packing tool during the installation of the module. The Crypto Officer shall not install the package if the integrity fails.

The module cannot use the following environment variables:

GNUTLS_NO_EXPLICIT_INIT
GNUTLS_SKIP_FIPS_INTEGRITY_CHECKS

The module can only be used with the cryptographic algorithms provided. Therefore, the following API functions are forbidden in the approved mode of operation:

gnutls_crypto_register_cipher
gnutls_crypto_register_aead_cipher
gnutls_crypto_register_mac
gnutls_crypto_register_digest
gnutls_privkey_import_ext4

11.2 Administrator Guidance

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

The Crypto Officer shall follow this Security Policy to configure the operational environment and install the module to be operated in the approved mode.

The output of the “Show module name and version” service is:

“Canonical Ltd. Ubuntu 22.04 GnuTLS Cryptographic Module” and "3.7.3-4ubuntu1.2+Fips1.1".

11.3 Non-Administrator Guidance

The approved security functions are listed in section 2.6 of this security policy.

The logical interfaces available to the users of the cryptographic module are listed in section 3.1.

For the secure operation of the module, the operator must follow the instructions in section 11.1 of this security policy.

11.4 Design and Rules

N/A

11.5 Maintenance Requirements

N/A

11.6 End of Life

For secure sanitization of the cryptographic module, the module needs first to be powered off, which will zeroize all keys and CSPs in volatile memory. The module does not possess persistent storage of SSPs, so further sanitization steps are not needed.

11.7 Additional Information

N/A

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

12 Mitigation of Other Attacks

The module does not mitigate other attacks.

12.1 Attack List

N/A

12.2 Mitigation Effectiveness

N/A

12.3 Guidance and Constraints

N/A

12.4 Additional Information

N/A

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Appendix A: TLS Cipher Suites

The module supports the following cipher suites for the TLS protocol version 1.0, 1.1, 1.2 and 1.3, compliant with section 3.3.1 of SP800-52rev2. Each cipher suite defines the key exchange algorithm, the bulk encryption algorithm (including the symmetric key size) and the MAC algorithm.

Cipher Suite	ID	Reference
TLS_DH_RSA_WITH_AES_128_CBC_SHA	{ 0x00, 0x31 }	RFC3268
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	{ 0x00, 0x33 }	RFC3268
TLS_DH_RSA_WITH_AES_256_CBC_SHA	{ 0x00, 0x37 }	RFC3268
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	{ 0x00, 0x39 }	RFC3268
TLS_DH_RSA_WITH_AES_128_CBC_SHA256	{ 0x00,0x3F }	RFC5246
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256	{ 0x00,0x67 }	RFC5246
TLS_DH_RSA_WITH_AES_256_CBC_SHA256	{ 0x00,0x69 }	RFC5246
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	{ 0x00,0x6B }	RFC5246
TLS_PSK_WITH_AES_128_CBC_SHA	{ 0x00, 0x8C }	RFC4279
TLS_PSK_WITH_AES_256_CBC_SHA	{ 0x00, 0x8D }	RFC4279
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	{ 0x00, 0x9E }	RFC5288
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	{ 0x00, 0x9F }	RFC5288
TLS_DH_RSA_WITH_AES_128_GCM_SHA256	{ 0x00, 0xA0 }	RFC5288
TLS_DH_RSA_WITH_AES_256_GCM_SHA384	{ 0x00, 0xA1 }	RFC5288
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA	{ 0xC0, 0x04 }	RFC4492
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA	{ 0xC0, 0x05 }	RFC4492
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	{ 0xC0, 0x09 }	RFC4492
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	{ 0xC0, 0x0A }	RFC4492
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA	{ 0xC0, 0x0E }	RFC4492
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA	{ 0xC0, 0x0F }	RFC4492
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	{ 0xC0, 0x13 }	RFC4492
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	{ 0xC0, 0x14 }	RFC4492

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Cipher Suite	ID	Reference
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	{ 0xC0, 0x23 }	RFC5289
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	{ 0xC0, 0x24 }	RFC5289
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256	{ 0xC0, 0x25 }	RFC5289
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384	{ 0xC0, 0x26 }	RFC5289
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	{ 0xC0, 0x27 }	RFC5289
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	{ 0xC0, 0x28 }	RFC5289
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256	{ 0xC0, 0x29 }	RFC5289
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384	{ 0xC0, 0x2A }	RFC5289
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	{ 0xC0, 0x2B }	RFC5289
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	{ 0xC0, 0x2C }	RFC5289
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256	{ 0xC0, 0x2D }	RFC5289
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384	{ 0xC0, 0x2E }	RFC5289
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	{ 0xC0, 0x2F }	RFC5289
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	{ 0xC0, 0x30 }	RFC5289
TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256	{ 0xC0, 0x31 }	RFC5289
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384	{ 0xC0, 0x32 }	RFC5289
TLS_DHE_RSA_WITH_AES_128_CCM	{ 0xC0, 0x9E }	RFC6655
TLS_DHE_RSA_WITH_AES_256_CCM	{ 0xC0, 0x9F }	RFC6655
TLS_DHE_RSA_WITH_AES_128_CCM_8	{ 0xC0, 0xA2 }	RFC6655
TLS_DHE_RSA_WITH_AES_256_CCM_8	{ 0xC0, 0xA3 }	RFC6655
TLS_AES_128_GCM_SHA256	{ 0x13, 0x01 }	RFC8446
TLS_AES_256_GCM_SHA384	{ 0x13, 0x02 }	RFC8446
TLS_AES_128_CCM_SHA256	{ 0x13, 0x04 }	RFC8446
TLS_AES_128_CCM_8_SHA256	{ 0x13, 0x05 }	RFC8446

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Appendix B. Glossary and Abbreviations

AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cipher Feedback
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CPACF	Central Processor Assist for Cryptographic Function
CSP	Critical Security Parameter
CTR	Counter Mode
DF	Derivation Function
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standards Publication
FSM	Finite State Model
GCM	Galois Counter Mode
HMAC	Hash Message Authentication Code
KAS	Key Agreement Schema
KAT	Known Answer Test
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
OFB	Output Feedback
OS	Operating System
PAA	Processor Algorithm Acceleration
PAI	Processor Algorithm Implementation
PR	Prediction Resistance
PSS	Probabilistic Signature Scheme
RNG	Random Number Generator
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

SSH	Secure Shell
SSP	Sensitive Security Parameter
XTS	XEX-based Tweaked-codebook mode with cipher text Stealing

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

Appendix C. References

FIPS140-3	**FIPS PUB 140-3 - Security Requirements For Cryptographic Modules** March 2019 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-3.pdf
FIPS140-3_IG	**Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program** March 2024 https://csrc.nist.gov/csrc/media/Projects/cryptographic-module-validation-program/documents/fips%20140-3/FIPS%20140-3%20IG.pdf
FIPS180-4	**Secure Hash Standard (SHS)** August 2015 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
FIPS186-4	**Digital Signature Standard (DSS)** July 2013 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
FIPS197	**Advanced Encryption Standard** November 2001 https://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
FIPS198-1	**The Keyed Hash Message Authentication Code (HMAC)** July 2008 https://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf
FIPS202	**SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions** August 2015 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf
PKCS#1	**Public Key Cryptography Standards (PKCS) #1: RSA Cryptography** Specifications Version 2.1 February 2003 https://www.ietf.org/rfc/rfc3447.txt
RFC3394	**Advanced Encryption Standard (AES) Key Wrap Algorithm** September 2002 https://www.ietf.org/rfc/rfc3394.txt
RFC5649	**Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm** September 2009 https://www.ietf.org/rfc/rfc5649.txt
SP800-38A	**NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques** December 2001 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

SP800-38B	**NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication** May 2005 https://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** May 2004 https://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** November 2007 https://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** January 2010 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38e.pdfhttps://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf
SP800-38F	**NIST Special Publication 800-38F - Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping** December 2012 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf
SP800-38G	**NIST Special Publication 800-38G - Recommendation for Block Cipher Modes of Operation: Methods for Format - Preserving Encryption** March 2016 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38G.pdf
SP800-52rev2	**NIST Special Publication 800-52 Revision 2 - Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations** August 2019 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r2.pdf
SP800-56Arev3	**NIST Special Publication 800-56A Revision 3 - Recommendation for Pair Wise Key Establishment Schemes Using Discrete Logarithm Cryptography** April 2018 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf
SP800-56Crev2	**NIST Special Publication 800-56C Revision 2 - Recommendation for Key Derivation through Extraction-then-Expansion** August 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr2.pdf
SP800-57rev5	**NIST Special Publication 800-57 Part 1 Revision 5 - Recommendation for Key Management Part 1: General** May 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.

SP800-90Arev1	**NIST Special Publication 800-90A Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf
SP800-90B	**NIST Special Publication 800-90B - Recommendation for the Entropy Sources Used for Random Bit Generation** January 2018 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf
SP800-108rev1	**NIST Special Publication 800-108 Revision 1 - Recommendation for Key Derivation Using Pseudorandom Functions (Revised)** August 2022 https://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf
SP800-131Arev2	**NIST Special Publication 800-131 Revision 2 - Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths** March 2019 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf
SP800-132	**NIST Special Publication 800-132 - Recommendation for Password-Based Key Derivation - Part 1: Storage Applications** December 2010 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf
SP800-133rev2	**NIST Special Publication 800-133 Revision 2 - Recommendation for Cryptographic Key Generation** June 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r2.pdf
SP800-135rev1	**NIST Special Publication 800-135 Revision 1 - Recommendation for Existing Application-Specific Key Derivation Functions** December 2011 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-135r1.pdf
SP800-140B	**NIST Special Publication 800-140B - CMVP Security Policy Requirements** March 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-140B.pdf

© 2024 Canonical Ltd. / atsec information security.
This document can be reproduced and distributed only whole and intact, including this copyright notice.