Canonical Ltd. Ubuntu 22.04 Strongswan Cryptographic Module

Version: 5.9.5-2ubuntu2.1+Fips1

FIPS 140-3 Non-Proprietary Security Policy

Document Version: 1.1

Last Update: 2024-11-27

Prepared by:
atsec information security corporation
4516 Seton Center Parkway, Suite 250
Austin, TX 78759
www.atsec.com

Prepared for:
Canonical Ltd.
110 Southwark Street, Blue Fin Building, 5th Floor
London, SE1 0SU
www.canonical.com

Table of Contents

List of Tables

List of Figures

1. General Information

1.1. Overview

This document is the non-proprietary FIPS 140-3 Security Policy for version 5.9.5-2ubuntu2.1+Fips1 of the Canonical Ltd. Ubuntu 22.04 Strongswan Cryptographic Module. It contains the security rules under which the module must operate and describes how this module meets the requirements as specified in FIPS PUB 140-3 (Federal Information Processing Standards Publication 140-3) for an overall Security Level 1 module. This Non-Proprietary Security Policy may be reproduced and distributed, but only whole and intact and including this notice. Other documentation is proprietary to their authors.

1.2. How this Security Policy was Prepared

In preparing the Security Policy document, the laboratory formatted the vendor-supplied documentation for consolidation without altering the technical statements therein contained. The further refining of the Security Policy document was conducted iteratively throughout the conformance testing, wherein the Security Policy was submitted to the vendor, who would then edit, modify, and add technical contents. The vendor would also supply additional documentation, which the laboratory formatted into the existing Security Policy, and resubmitted to the vendor for their final editing.

1.3. Security Levels

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

ISO/IEC 24759 Section 6. [Number Below]	FIPS 140-3 Section Title	Security Level
1	General	1
2	Cryptographic Module Specification	1
3	Cryptographic Module Interfaces	1
4	Roles, Services, and Authentication	1
5	Software/Firmware Security	1
6	Operational Environment	1
7	Physical Security	Not Applicable
8	Non-invasive Security	Not Applicable
9	Sensitive Security Parameter Management	1
10	Self-tests	1
11	Life-cycle Assurance	1
12	Mitigation of Other Attacks	Not Applicable

Table 1 - Security Levels

2. Cryptographic Module Specification

2.1. Description

Purpose and Use: The Canonical Ltd. Ubuntu 22.04 Strongswan Cryptographic Module (hereafter referred to as “the module”) provides cryptographic services for the Internet Key Exchange (IKE) protocol in the Ubuntu Operating System user space.

The module uses the Canonical Ltd. Ubuntu 22.04 OpenSSL Cryptographic Module as a bound module (also referred to as “the bound OpenSSL module”), which provides the underlying cryptographic algorithms necessary for establishing and maintaining IKE sessions. The Canonical Ltd. Ubuntu 22.04 OpenSSL Cryptographic Module is a FIPS-validated module with certificate #4794.

The module also uses the Canonical Ltd. Ubuntu 22.04 Kernel Crypto API Cryptographic Module as a bound module (also referred to as “the bound Kernel Crypto API module”) for performing integrity tests. The Canonical Ltd. Ubuntu 22.04 Kernel Crypto API Cryptographic Module is a FIPS-validated module with certificate #4894.

Module Type: Software

Module Embodiment: Multi-chip standalone

Module Characteristics: N/A

Cryptographic Boundary: The cryptographic boundary of the module is defined as the IKEv2 daemon, the libraries and plugins, and the *ipsec* command. In addition, the cryptographic boundary contains the .hmac files which store the expected integrity values for each of the software components.

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

Picture or Block Diagram

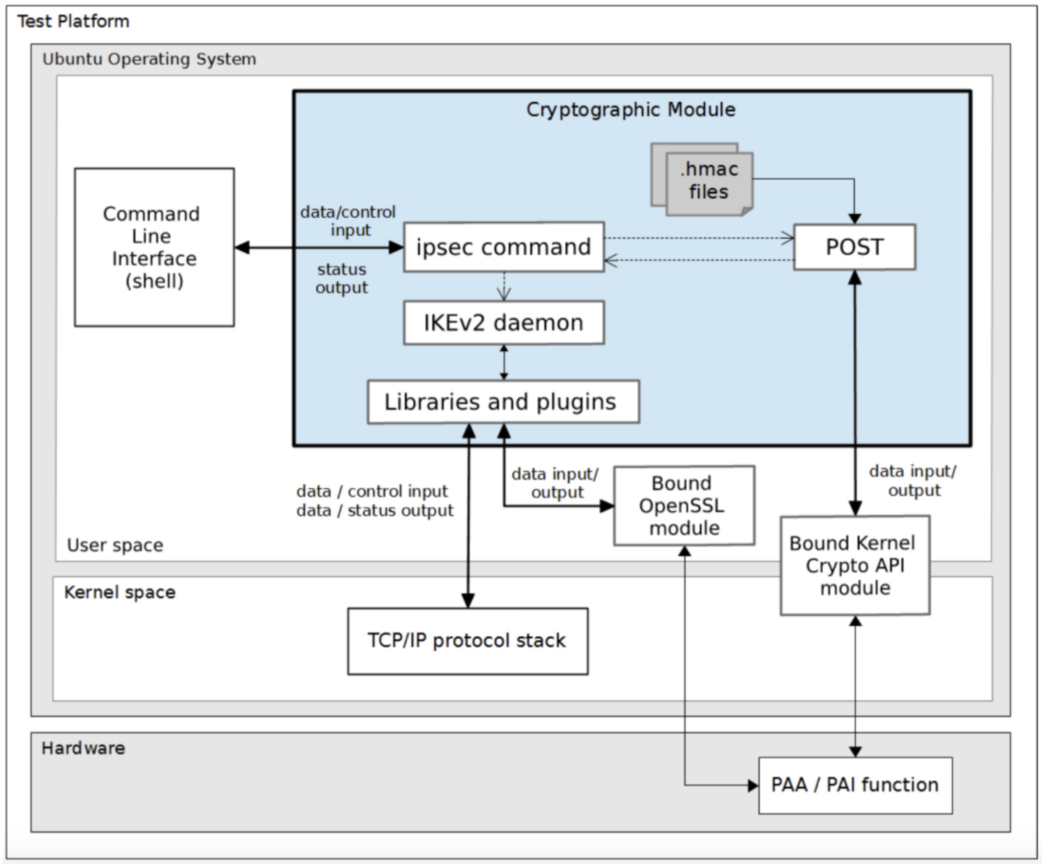

Figure 1 - Block Diagram

2.2. Version Information

Hardware Versions: N/A

Software Versions: 5.9.5-2ubuntu2.1+Fips1

Firmware Versions: N/A

2.3. Operating Environments

Hardware Operating Environments: N/A

Software, Firmware, Hybrid Tested Operating Environments:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS
Ubuntu 22.04	Supermicro SYS-1019P-WTR	Intel Xeon Gold 6226	AES-NI, SHA extensions	N/A
Ubuntu 22.04	Amazon Web Services (AWS) c6g.metal	AWS Graviton2	NEON, Crypto Extensions	N/A
Ubuntu 22.04	IBM z15	IBM z15	CPACF	N/A
Ubuntu 22.04	Supermicro SYS-1019P-WTR	Intel Xeon Gold 6226	None	N/A
Ubuntu 22.04	Amazon Web Services (AWS) c6g.metal	AWS Graviton2	None	N/A

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS
Ubuntu 22.04	IBM z15	IBM z15	None	N/A

Table 2 - Software, Firmware, Hybrid Tested Operating Environments

Executable Code Sets:

<table>
<tr><th>Package or File Name</th><th>Software/ Firmware Version</th><th>Features</th><th>Hybrid Hardware Version</th><th>Integrity Test</th></tr>
<tr><td>/usr/sbin/ipsec
/usr/lib/ipsec/stroke
/usr/lib/ipsec/starter
/usr/lib/ipsec/charon
/usr/lib/ipsec/pool
/usr/lib/ipsec/_updown</td><td rowspan="4">5.9.5-2ubuntu2.1+Fips1</td><td rowspan="4">N/A</td><td rowspan="4">N/A</td><td rowspan="4">HMAC SHA-256</td></tr>
<tr><td>/usr/lib/ipsec/_fipscheck
/usr/lib/ipsec/ikev2-kdf-selftest</td></tr>
<tr><td>/usr/lib/ipsec/libstrongswan.so.0.0.0
/usr/lib/ipsec/plugins/libstrongswan-openssl.so
/usr/lib/ipsec/libcharon.so.0.0.0</td></tr>
<tr><td>/usr/lib/ipsec/plugins/libstrongswan-fips-prf.so
/usr/lib/ipsec/plugins/libstrongswan-nonce.so
/usr/lib/ipsec/plugins/libstrongswan-dnskey.so
/usr/lib/ipsec/plugins/libstrongswan-pem.so
/usr/lib/ipsec/plugins/libstrongswan-pgp.so
/usr/lib/ipsec/plugins/libstrongswan-pkcs1.so
/usr/lib/ipsec/plugins/libstrongswan-pkcs7.so
/usr/lib/ipsec/plugins/libstrongswan-pkcs8.so
/usr/lib/ipsec/plugins/libstrongswan-pkcs12.so
/usr/lib/ipsec/plugins/libstrongswan-pubkey.so
/usr/lib/ipsec/plugins/libstrongswan-sshkey.so
/usr/lib/ipsec/plugins/libstrongswan-x509.so
/usr/lib/ipsec/plugins/libstrongswan-constraints.so
/usr/lib/ipsec/plugins/libstrongswan-revocation.so
/usr/lib/ipsec/plugins/libstrongswan-kernel-netlink.so
/usr/lib/ipsec/plugins/libstrongswan-socket-default.so</td></tr>
</table>

Package or File Name	Software/ Firmware Version	Features	Hybrid Hardware Version	Integrity Test
/usr/lib/ipsec/plugins/libstrongswan-stroke.so /usr/lib/ipsec/plugins/libstrongswan-attr.so /usr/lib/ipsec/plugins/libstrongswan-resolve.so /usr/lib/ipsec/plugins/libstrongswan-updown.so				
Kernel Bound Module				
/boot/vmlinuz-5.15.0-73-fips	5.15.0-73-fips	N/A	N/A	HMAC SHA-512
*.ko files in /usr/lib/modules/5.15.0-73-fips/kernel/crypto/ *.ko files in /usr/lib/modules/5.15.0-73-fips/kernel/arch/x86/crypto/ *.ko files in /usr/lib/modules/5.15.0-73-fips/kernel/arch/arm64/crypto/ *.ko files in /usr/lib/modules/5.15.0-73-fips/kernel/arch/s390/crypto/				RSA signature verification
/usr/lib/*-linux-gnu/libkcapi.so.1.4.0 /usr/bin/sha512hmac	1.4.0-1ubuntu0.1~Fips1			HMAC SHA-512
OpenSSL Bound Module				
/usr/lib/x86_64-linux-gnu/ossl-modules-3/fips.so	3.0.5-0ubuntu0.1+Fips2.1	N/A	N/A	HMAC-SHA-256

Table 3 - Executable Code Sets

Vendor Affirmed Operating Environments: N/A

2.4. Excluded Components

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

2.5. Modes of Operation

Modes List and Description:

Name	Description	Type	Status Indicator
Approved mode	Automatically entered when the module is operational.	Approved	Equivalent to the indicator of the requested service.

Table 4 - Modes List and Description

The module enters Approved mode after passing all pre-operational self-tests and cryptographic algorithm self-tests executed on start-up. The approved mode of operation is assumed once the module is operational.

Mode change instructions and status indicators: The Canonical Ltd. Ubuntu 22.04 Strongswan Cryptographic Module implements the approved service indicator relying on a global service indicator (In compliance with IG 2.4.C - Table) which is the successful establishment of the IKE connection.

Degraded Mode Description: The module does not implement a degraded mode of operation.

2.6. Algorithms

Approved Algorithms:

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strengths	Use / Function
A4017[1]	KDF IKEv2 (CVL)	HMAC with SHA-1, SHA-256, SHA-384, SHA-512	112-256 bits	Key derivation in the IKEv2 protocol
OpenSSL Bound Module				
A3958 A3959 A3960 A3973 A3980 A3981 A3982	AES [FIPS 197, SP 800-38A, SP 800-38A Addendum, SP 800-38F]	CBC, CCM	128, 192, 256 bits	Authenticated Encryption Authenticated Decryption
A3961 A3974 A3975 A3976 A3988 A3989 A3990 A3994 A3995 A3996 A3997 A3998 A3999 A4000 A4001 A4002	AES [FIPS 197, SP 800-38D]	GCM (internal IV)	128, 192, 256 bits	Authenticated Encryption
A3961 A3974 A3975 A3976 A3988 A3989 A3990 A3994 A3995 A3996 A3997 A3998 A3999 A4000 A4001 A4002	AES [FIPS 197, SP 800-38D]	GCM (external IV)	128, 192, 256 bits	Authenticated Decryption
A3970	CTR_DRBG [SP 800-90Ar1]	AES-128, AES-192, AES-256, with/without derivation function, with/without prediction resistance	128, 192, 256 bits	Random number generation
A3962 A3977 A3983 A3993 A4003 A4004 A4005	ECDSA [FIPS 186-4]	SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256	P-224, P-256, P-384, P-521 (112-256 bits)	Signature generation
A3964 A3972 A3979		SHA3-224, SHA3-256, SHA3-384, SHA3-512		
A3962 A3977 A3983 A3993 A4003 A4004 A4005		SHA-1, SHA-224, SHA-256, SHA-384, SHA- 512, SHA-512/224, SHA-512/256	P-224, P-256, P-384, P-521 (112-256 bits)	Signature verification
A3964 A3972 A3979		SHA3- 224, SHA3-256, SHA3-384, SHA3-512		

[1] Although HMAC and SHA are implemented by the bound OpenSSL module, the CAVP certificate still contains HMAC and SHA as pre-requisite algorithms.

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strengths	Use / Function
A3962 A3977 A3983 A3993 A4003 A4004 A4005		Appendix B.4.2 Testing Candidates	P-224, P-256, P-384, P-521 (112-256 bits)	Key pair generation
A3962 A3977 A3983 A3993 A4003 A4004 A4005		N/A	P-224, P-256, P-384, P-521 (112-256 bits)	Key pair verification
A3962 A3977 A3983 A3993 A4003 A4004 A4005	RSA [FIPS 186-4]	PKCS#1 v1.5 and PSS with SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256	2048-16384 bits (112-256 bits)	Signature generation
A3962 A3977 A3983 A3993 A4003 A4004 A4005		PKCS#1 v1.5 and PSS with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256	2048-16384 bits (112-256 bits)	Signature verification
A3962 A3977 A3983 A3993 A4003 A4004 A4005	HMAC [FIPS 198-1]	SHA-1, SHA-224, SHA-256, SHA-384, SHA- 512, SHA-512/224, SHA-512/256	112-524288 bits (112-256 bits)	Message authentication
A3962 A3977 A3983 A3993 A4003 A4004 A4005	SHA [FIPS 180-4]	SHA-1, SHA-224, SHA-256, SHA-384, SHA- 512, SHA-512/224, SHA-512/256	N/A	Message digest
A3963	HMAC [FIPS 198-1]	SHA-256	112-524288 bits (112-256 bits)	Message authentication
	SHA [FIPS 180-4]	SHA-256	N/A	Message digest
A3992	KAS-FFC-SSC [SP 800-56Ar3]	dhEphem (initiator/responder)	MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 (112-200 bits)	Shared secret computation
A3962 A3977 A3983 A3993 A4003 A4004 A4005	KAS-ECC-SSC [SP 800-56Ar3]	Ephemeral Unified Model (initiator/responder)	P-224, P-256, P-384, P-521 (112-256 bits)	Shared secret computation
A3992	Safe primes [SP 800-56Ar3]	SP 800-56Ar3 Section 5.6.1.1.4 Testing Candidates	MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 (112-200 bits)	Key pair generation Key pair verification

CAVP Cert	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strengths	Use / Function
Kernel Bound Module				
A3812 A3813 A3814 A3832 A3850 A3851 A3852 A3853 A3857 A3858	SHA [FIPS 180-4]	SHA-256	N/A	Message digest
A3812 A3813 A3814 A3832 A3850 A3851 A3852 A3853 A3857 A3858	HMAC [FIPS 198-1]	SHA-256	128-524288 bits (128 bits)	Message authentication

Table 5 - Approved Algorithms

Vendor Affirmed Algorithms: N/A

Non-Approved, Allowed Algorithms:

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

Non-Approved, Allowed Algorithms with No Security Claimed:

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

Non-Approved, Not Allowed Algorithms:

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

2.7. RNG and Entropy

Entropy Information:

Name	Type	Operational Environment	Sample Size	Entropy Per Sample	Conditioning Component
OpenSSL CPU Time Jitter RNG Entropy Source (Cert. #E62)	Non-physical	See Table 2	64 bits	64 bits	Linear-Feedback Shift Register (LFSR)

Table 6 - Entropy

RNG Information: The module does not implement any random number generator. Instead, it uses the Random Number Generation (RNG) service provided by the bound Canonical Ltd. Ubuntu 22.04 OpenSSL Cryptographic Module, which implements a Deterministic Random Bit Generator (DRBG) based on [SP800-90Ar1]. The DRBG is seeded with 384 bits of entropy and 256 bits of entropy are used to reseed the DRBG. The highest SSP security strength generated by the module is 256 bits.

2.8. SSP Generation

The module implements the key derivation portion of the DH and ECDH key agreement, using the NIST SP 800-135 IKEv2 (CVL) in compliance with Section 6.2 of SP 800-132r2. The DH and ECDH key pairs to be used in the IKEv2 protocol are generated by the bound OpenSSL module. Below are listed the SSP generation methods provided by the bound OpenSSL module:

Name	Type	Properties
Safe primes key pair generation	CKG	Key type: Diffie-Hellman key pair Groups: MODP-2048, MODP-3072, MODP4096, MODP-6144, MODP-8192 Security strength: 112-200 bits Method: SP 800-56Ar3 (safe primes) Section 5.6.1.1.4 Testing Candidates Compliant to SP 800-133r2, Section 5.2. Random seeds are obtained directly from an SP 800-90Arev1 compliant DRBG in compliance with SP 800-133rev2 section 4 (without the use of V, as described in the additional comment 2 of IG D.H).
EC key pair generation	CKG	Key type: EC Diffie-Hellman key pair Curves: P-224, P-256, P-384, P-521 Security strength: 112, 128, 192, 256 bits Method: FIPS 186-4 Appendix B.4.2 Testing Candidates Compliant to SP 800-133r2, Section 5.1 and 5.2. Random seeds are obtained directly from an SP 800-90Arev1 compliant DRBG in compliance with SP 800-133rev2 section 4 (without the use of V, as described in the additional comment 2 of IG D.H).

Table 7 - SSP Generation

2.9. SSP Establishment

The Canonical Ltd. Ubuntu 22.04 Strongswan Cryptographic Module and the bound OpenSSL module together provide the Diffie Hellman and EC Diffie Hellman key agreement. The Canonical Ltd. Ubuntu 22.04 Strongswan Cryptographic Module only implements the NIST SP 800-135 IKEv2 KDF (CVL) part of the key agreement using the HMAC portion of the SSP agreement and the bound OpenSSL module provides the shared secret computation. Below are listed the SSP agreement methods provided by the bound OpenSSL module:

Name	Type	Properties
Diffie-Hellman key agreement with IKE KDF	KAS	Groups: MODP-2048, MODP-3072, MODP4096, MODP-6144, MODP-8192 Security strength: 112-200 bits Compliant with Scenario 2 (2) of FIPS 140-3 IG D.F
EC Diffie-Hellman key agreement with IKE KDF	KAS	Curves: P-224, P-256, P-384, P-521 Security strength: 112, 128, 192, 256 bits Compliant with Scenario 2 (2) of FIPS 140-3 IG D.F

Table 8 - SSP Agreement

The module does not implement any key transport method.

2.10. Design and Rules

The module performs pre-operational self-test and cryptographic algorithm self-tests when it is loaded into memory without operator intervention. Pre-operational self-tests ensure that the module is not corrupted and that the cryptographic algorithms work as expected. While the module is executing the self-tests, services are not available, and input and output are inhibited. The module is not available for use until the self-tests complete successfully.

If any pre-operational self-test fails, the module will return the error message listed in Table 19, enter the error state and terminate. Therefore, no cryptographic operations or data output are possible.

Note: The bound Canonical Ltd. Ubuntu 22.04 OpenSSL Cryptographic Module and the Canonical Ltd. Ubuntu 22.04 Kernel Crypto API Cryptographic Module perform their own pre-operational and cryptographic algorithm self-tests automatically when they are loaded into memory. The Canonical Ltd. Ubuntu 22.04 Strongswan Cryptographic Module ensures that both bound modules complete their pre-operational self-tests successfully.

2.11. Initialisation

There are no specific initialization requirements.

3. Cryptographic Module Interfaces

3.1. Description

Physical Port	Logical Interface	Data that passes over the port/interface
As a software-only module, the module does not have physical ports. Physical Ports are interpreted to be the physical ports of the hardware platform on which it runs.	Data Input	/etc/ipsec.secrets file, private key file, certificate files under the /etc/ipsec.d directory, input data received from the network (IKEv2 protocol), input data received from the bound OpenSSL module via its API parameters.
	Data Output	Output data sent through the network (IKEv2 protocol),output data sent to the bound OpenSSL module via its API parameters.
	Control Input	Invocation of the *ipsec* command on the command line, control parameters via the *ipsec* command and the /etc/ipsec.conf file, IKEv2 protocol message requests received from the network.
	Status Output	Status messages returned after execution of the *ipsec* command, status of processing IKEv2 protocol message requests sent through the network.
	Power Input	N/A

Table 9 - Ports and Interfaces

The logical interfaces are the APIs through which the applications request services. These logical interfaces are logically separated from each other by the API design.

3.2. Trusted Channel Specification

The module does not implement a trusted channel.

3.3. Control Interface Not Inhibited

The module does not implement a control output interface.

4. Roles, Services, and Authentication

4.1. Authentication Methods

The module does not implement operator authentication.

4.2. Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	N/A

Table 10 - Roles

The module supports the Crypto Officer role only. This sole role is implicitly and always assumed by the operator of the module. No support is provided for multiple concurrent operators.

4.3. Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Start IKEv2 daemon	Start IKE daemon	Successful establishment of the IKE connection.	N/A	Success/Fail	N/A	N/A
Configure IKEv2 daemon	Configure IKEv2 daemon	Successful establishment of the IKE connection.	Pre-shared Key or Post-Quantum Pre-shared Key RSA public key RSA private key EC public key EC private key	Success/Fail	N/A	Pre-shared Key or Post-Quantum Pre-shared Key; RSA public key; RSA private key; EC public key; EC private key: R
IKE_SA_INIT Exchange	Key exchange	Successful establishment of the IKE connection.	Private and public key	Shared secret	Diffie-Hellman (modp2048, modp3072, modp4096, modp6144, modp8192) with key size between 2048 and 8192 bits EC Diffie-Hellman with NIST curves P-224, P-256, P-384, P-521	DH public key, DH private key, EC public key, EC private key, Shared secret: G, R

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	Key derivation	Successful establishment of the IKE connection.	N/A	Success/Fail	SP800-135r1 IKEv2 KDF (CVL) using HMAC with SHA-1, SHA-256, SHA-384 and SHA-512	Derivation key (SK_d), Encryption key (SK_ei, SK_er), Authentication key (SK_ai, SK_ar), Authentication payload key (SK_pi, SK_pr), Shared secret: G
IKE_AUTH Exchange	Signature generation Signature verification	Successful establishment of the IKE connection.	Private and public key	Success/Fail	RSA PKCS#1 v1.5 and PSS with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256 ECDSA (P-224, P-256, P-384, P-521) with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256	RSA public key, RSA private key, EC public key, EC private key, RSA Peer's public key, Peer's EC public key: W
	Authenticated Encryption Authenticated Decryption	Successful establishment of the IKE connection.	Encryption key Authentication key	Success/Fail	AES-CBC + HMAC with SHA-1, SHA-256, SHA-384, SHA-512 AES-GCM AES-CCM	Encryption key (SK_ei, SK_er): W Authentication key (SL_ai, SK_ar): W
CREATE_CHILD_SA Exchange	Authenticated Encryption Authenticated Decryption	Successful establishment of the IKE connection.	Encryption key Authentication key	Success/Fail	AES-CBC + HMAC with SHA-1, SHA-256, SHA-384, SHA-	Encryption key (SK_ei, SK_er): W Authenticat

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					512 AES-GCM AES-CCM	ion key (SK_ai, SK_ar): W
CREATE_CHILD_SA Exchange	Key exchange	Successful establishment of the IKE connection.	N/A	Shared secret	Diffie-Hellman (modp2048, modp3072, modp4096, modp6144, modp8192) with key size between 2048 and 8192 bits EC Diffie-Hellman with NIST curves P-224, P-256, P-384, P-521	DH public key, DH private key, EC public key, EC private key, Shared secret: G
CREATE_CHILD_SA Exchange	Key derivation	Successful establishment of the IKE connection.	Derivation key	Derived key	SP800-135r1 IKEv2 KDF (CVL) using HMAC with SHA-1, SHA-256, SHA-384 and SHA-512	Derivation key (SK_d): W; New derivation key (SK_d), New encryption key (SK_ei, SK_er), New authentication key (SK_ai, SK_ar), New authentication payload key (SK_pi, SK_pr): G
INFORMATIONAL Exchange	Authenticated Encryption Authenticated Decryption	Successful establishment of the IKE connection.	Encryption key Authentication key	Success/Fail	AES-CBC + HMAC with SHA-1, SHA-256, SHA-384, SHA-512 AES-GCM AES-CCM	Encryption key (SK_ei, SK_er): W Authentication key (SK_ai, SK_ar): W
Show version	Return the module name and version information	None	N/A	Module name and version	N/A	N/A

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Show status	Return the module status	None	N/A	Module status	N/A	N/A
Self-test	Perform the CASTs and integrity tests	None	N/A	Success/fail	See Section 10	N/A
Zeroization	Close Security Association	None	Any SSP	Success/Fail	N/A	All SSPs: Z
	Terminate IKEv2 daemon	None	Any SSP	Success/Fail	N/A	All SSPs: Z

Table 11 - Approved Services

Table 11 lists the approved services. The following convention is used to specify access rights to SSPs:

- **Generate (G):** The module generates or derives the SSP.
- **Read (R):** The SSP is read from the module (e.g. the SSP is output).
- **Write (W):** The SSP is updated, imported, or written to the module.
- **Execute (E):** The module uses the SSP in performing a cryptographic operation.
- **Zeroize (Z):** The module zeroizes the SSP.

4.4. Non-Approved Services

The module does not implement any non-approved services.

4.5. External Software/Firmware Loaded

The module does not load external software or firmware.

4.6. Bypass Actions and Status

The module does not implement a bypass capability.

4.7. Cryptographic Output Actions and Status

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

5. Software/Firmware Security

5.1. Integrity Techniques

The integrity of the module is verified by comparing an HMAC-SHA-256 value calculated at run time with the HMAC value stored in the .hmac file that was computed at build time, for each of the components that comprise the module. The HMAC-SHA-256 algorithm for integrity test is provided by the bound Canonical Ltd. Ubuntu 22.04 Kernel Crypto API Cryptographic Module. If the HMAC values do not match, the test fails and the module enters the error state.

5.2. Initiate on Demand

Integrity test is performed as part of the pre-operational self-tests, which are executed when the module is initialized. The integrity test can be invoked on demand by unloading and subsequently re-initializing the module, which will perform (among others) the software integrity tests.

6. Operational Environment

6.1. Operational Environment Type and Requirements

Type of Operating Environment: modifiable; the module executes as part of a general-purpose operating system (Canonical Ubuntu 22.04), which allows modification, loading, and execution of software that is not part of the validated module.

How Requirements are Satisfied: If properly installed, 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. Processes that are spawned by the cryptographic module are owned by the module and are not owned by external processes/operators.

6.2. Configuration Settings and Restrictions

The module shall be installed as stated in Section 11.1.

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

7. Physical Security

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

8. Non-Invasive Security

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

9. Sensitive Security Parameters Management

9.1. Storage Areas

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

Table 12 - Storage Areas

The module does not perform persistent storage of SSPs. The SSPs are temporarily stored in the RAM in plaintext form. SSPs are provided to the module by the calling process and are destroyed when released by the appropriate zeroization function calls. Public and private keys for IKEv2 authentication are stored in the /etc/ipsec.d/certs and /etc/ipsec.d/private directories, which are within the module's physical perimeter, but outside its cryptographic boundary.

9.2. SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	Related SFI
Key files	Operator within TOEPP	Cryptographic Module	Plaintext	Manual	Electronic	N/A
IKE protocol messages (input)	Operator within TOEPP	Cryptographic Module	Plaintext	Manual	Electronic	N/A
API parameters (input)	OpenSSL bound module	Cryptographic Module	Plaintext	Manual	Electronic	N/A
IKE protocol messages (output)	Cryptographic Module	Operator within TOEPP	Plaintext	Manual	Electronic	N/A
API parameters (output)	Cryptographic Module	OpenSSL bound module Operator	Plaintext	Manual	Electronic	N/A

Table 13 - SSP Input-Output

The module does not support manual key entry or intermediate key generation key output. The keys are entered from or output to the module electronically.

9.3. SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
IKE SA Close	Close of the IKEv2 Security Association (SA)	Memory occupied by SSPs is overwritten with zeroes, which renders the SSP values irretrievable.	By closing an IKE connection (i.e., the command *ipsec down*).
IKEv2 Daemon Terminate	Termination of the IKEv2 daemon.	Memory occupied by SSPs is overwritten with zeroes, which renders the SSP values irretrievable.	The command *ipsec stop*.
Remove power from the module	De-allocates the volatile memory used to store SSPs.	Volatile memory used by the module is overwritten within nanoseconds when power is removed.	By removing power.

Table 14 - SSP Zeroization Methods

The memory occupied by SSPs is allocated by regular memory allocation operating system calls. The module calls appropriate key zeroization functions provided by the bound OpenSSL module and calls its own appropriate key zeroization functions. In both cases, these functions overwrite the memory with zeroes and deallocate the memory with the regular memory deallocation operating system call. All data output is inhibited during zeroization.

9.4. SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
RSA private key	RSA private key.	2048-16384 bits (112-256 bits)	Private Key	N/A	N/A	IKE_AUTH Exchange Configure IKEv2 daemon
RSA public key	RSA public key.	2048-16384 bits (112-256 bits)	Public Key	N/A	N/A	IKE_AUTH Exchange Configure IKEv2 daemon
RSA Peer's public key	RSA Peer's public keys.	2048-16384 bits (112-256 bits)	Public Key	N/A	N/A	IKE_AUTH Exchange
DH public key	DH public key.	MODP-2048, MODP-3072, MODP4096, MODP-6144, MODP-8192 (112-200 bits)	Public Key	N/A (Generated by the bound module OpenSSL)	N/A	IKE_SA_INIT Exchange CREATE_CHILD_SA Exchange
DH private key	DH private key.	MODP-2048, MODP-3072, MODP4096, MODP-6144, MODP-8192 (112-200 bits)	Private Key	N/A (Generated by the bound module OpenSSL)	N/A	IKE_SA_INIT Exchange CREATE_CHILD_SA Exchange
Peer's DH public key	Peer's DH public key.	MODP-2048, MODP-3072, MODP4096, MODP-6144, MODP-8192 (112-200 bits)	Public Key	N/A	N/A	IKE_SA_INIT Exchange CREATE_CHILD_SA Exchange
EC private key	Private key for ECDSA or ECDH.	P-224, P-256, P-384, P-521 (112, 128, 192, 256 bits)	Private Key	N/A (Generated by the bound module OpenSSL)	N/A	IKE_SA_INIT Exchange CREATE_CHILD_SA Exchange IKE_AUTH Exchange Configure IKEv2 daemon
EC public key	Public key for ECDSA or ECDH.	P-224, P-256, P-384, P-521 (112, 128, 192, 256 bits)	Public Key	N/A (Generated by the bound	N/A	IKE_SA_INIT Exchange CREATE_CHILD_SA Exchange

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
				module OpenSSL)		IKE_AUTH Exchange Configure IKEv2 daemon
Peer's EC public key	Peer's public keys for ECDSA or ECDH.	P-224, P-256, P-384, P-521 (112, 128, 192, 256 bits)	Public Key	N/A	N/A	IKE_SA_INIT Exchange CREATE_CHILD_SA Exchange IKE_AUTH Exchange
Shared secret	Shared secret for DH or ECDH.	112-256 bits	Shared Secret	N/A	N/A (Established by the bound module OpenSSL)	IKE_SA_INIT Exchange CREATE_CHILD_SA Exchange
Derivation key (SK_d)	Derivation key (SK_d) from IKE_SA	112-256 bits	Symmetric Key	NIST SP800-135r1 IKEv2 KDF (CVL)	N/A	IKE_SA_INIT Exchange CREATE_CHILD_SA Exchange
Encryption key (SK_ei, SK_er)	Encryption keys from IKE_SA (SK_ei, SK_er) (AES)	112-256 bits	Symmetric Key	NIST SP800-135r1 IKEv2 KDF (CVL)	N/A	IKE_SA_INIT Exchange CREATE_CHILD_SA Exchange IKE_AUTH Exchange INFORMATIONAL Exchange
Authentication key (SK_ai, SK_ar)	Authentication keys from IKE_SA (SK_ai, SK_ar) (HMAC)	112-256 bits	Symmetric Key	NIST SP800-135r1 IKEv2 KDF (CVL)	N/A	IKE_SA_INIT Exchange CREATE_CHILD_SA Exchange IKE_AUTH Exchange INFORMATIONAL Exchange
Authentication payload key (SK_pi, SK_pr)	Authentication payload keys from IKE_SA (SK_pi, SK_pr) (HMAC)	112-256 bits	Symmetric Key	NIST SP800-135r1 IKEv2 KDF (CVL)	N/A	IKE_SA_INIT Exchange
New Derivation key (SK_d)	New Derivation Key from CHILD_SA (SK_d) (HMAC)	112-256 bits	Symmetric Key	NIST SP800-135r1 IKEv2 KDF (CVL)	N/A	CREATE_CHILD_SA Exchange
New Encryption key (SK_ei, SK_er)	New Encryption keys from CHILD_SA (SK_ei, SK_er) (HMAC)	112-256 bits	Symmetric Key	NIST SP800-135r1 IKEv2 KDF (CVL)	N/A	CREATE_CHILD_SA Exchange

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
New Authentication key (SK_ai, SK_ar)	New Authentication keys from CHILD_SA (SK_ai, SK_ar) (HMAC)	112-256 bits	Symmetric Key	NIST SP800-135r1 IKEv2 KDF (CVL)	N/A	CREATE_CHILD_SA Exchange
New Authentication payload key (SK_pi, SK_pr)	New Authentication payload keys from CHILD_SA (SK_pi, SK_pr) (HMAC)	112-256 bits	Symmetric Key	NIST SP800-135r1 IKEv2 KDF (CVL)	N/A	CREATE_CHILD_SA Exchange
Pre-shared Key or Post-Quantum Pre-shared Key	Pre-shared Key or Post-Quantum Pre-shared Key	112-256 bits	Symmetric Key	N/A	N/A	Configure IKEv2 daemon

Table 15 - SSP Information First

Name	Input - Output	Storage	Storage Duration	Type	Related SSPs
RSA private key	Input: Read from the private key files. Output: To the bound OpenSSL module via API parameters.	RAM	For the duration of the service.	CSP	RSA public key
RSA public key	Input: Read from the host key files. Output: To the network peer via IKE_AUTH exchange message.	RAM	For the duration of the service.	PSP	RSA private key
RSA Peer's public key	Input: From the network peer via IKE_AUTH exchange message. Output: To the bound OpenSSL module via API parameters.	RAM	For the duration of the service.	PSP	None
DH public key	Input: From the bound OpenSSL module via API parameters. Output: To the network peer via IKE_SA_INIT or CREATE_CHILD_SA exchange messages.	RAM	For the duration of the service.	PSP	DH private key
DH private key	Input: From the bound OpenSSL module via API parameters. Output: To the bound OpenSSL module via API parameters.	RAM	For the duration of the service.	CSP	DH public key
Peer's DH public key	Input: From the network peer IKE_SA_INIT or CREATE_CHILD_SA exchange messages Output: To the bound OpenSSL module via API parameters.	RAM	For the duration of the service.	PSP	None
EC private key	Input ECDSA: Read from the private key files. Output ECDSA: To the bound OpenSSL module via API parameters.	RAM	For the duration of the service.	CSP	EC public key
	Input ECDH: From the bound OpenSSL module via API parameters.	RAM	For the duration of the service.	CSP	EC public key

Name	Input - Output	Storage	Storage Duration	Type	Related SSPs
	Output ECDH: To the bound OpenSSL module via API parameters.				
EC public key	Input ECDSA: Read from the host key files. Output ECDSA: To the network peer via IKE_AUTH exchange message.	RAM	For the duration of the service.	PSP	EC private key
	Input ECDH: From the bound OpenSSL module via API parameters. Output ECDH: to the network peer via IKE_SA_INIT or CREATE_CHILD_SA exchange messages.	RAM	For the duration of the service.	PSP	EC private key
Peer's EC public key	Input ECDSA: From the network peer via IKE_AUTH exchange message. Output ECDSA: To the bound OpenSSL module via API parameters.	RAM	For the duration of the service.	PSP	None
	Input ECDH: From the network peer IKE_SA_INIT or CREATE_CHILD_SA exchange messages. Output ECDH: To the bound OpenSSL module via API parameters.	RAM	For the duration of the service.	PSP	None
Shared secret	Input: From the bound OpenSSL module via API parameters. Output: N/A.	RAM	For the duration of the service.	CSP	DH public key DH private key EC public key EC private key Peer's DH public key Peer's EC public key
Derivation key (SK_d)	Input: N/A Output: N/A	RAM	For the duration of the service.	CSP	None
Encryption key (SK_ei, SK_er)	Input: N/A Output: To the bound OpenSSL module via API parameters.	RAM	For the duration of the service.	CSP	None
Authentication key (SK_ai, SK_ar)	Input: N/A Output: To the bound OpenSSL module via API parameters.	RAM	For the duration of the service.	CSP	None
Authentication payload key (SK_pi, SK_pr)	Input: N/A Output: To the bound OpenSSL module via API parameters.	RAM	For the duration of the service.	CSP	None
New Derivation key (SK_d)	Input: N/A Output: N/A	RAM	For the duration of the service.	CSP	None
New Encryption key (SK_ei, SK_er)	Input: N/A Output: To the bound Kernel Crypto API module via API parameters.	RAM	For the duration of the service.	CSP	None

Name	Input - Output	Storage	Storage Duration	Type	Related SSPs
New Authentication key (SK_ai, SK_ar)	Input: N/A Output: To the bound Kernel Crypto API module via API parameters.	RAM	For the duration of the service.	CSP	None
New Authentication payload key (SK_pi, SK_pr)	Input: N/A Output: To the bound Kernel Crypto API module via API parameters.	RAM	For the duration of the service.	CSP	None
Pre-shared Key or Post-Quantum Pre-shared Key	Input: Read from the private key files. Output: N/A.	RAM	For the duration of the service.	CSP	None

Table 16 - SSP Information Second

9.5. Transitions

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

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

10. Self-Tests

10.1. Pre-Operational Self-Tests

Algorithm	Implementation	Test Properties	Test Method	Test Type	Indicator	Details
HMAC SHA-256	C	256-bit keys	HMAC SHA-256 value calculated at run time is compared with the precomputed HMAC SHA-256 value.	Software integrity	Module becomes operational	Integrity test performed by the bound Kernel Crypto API module to check the fipscheck tool and the Strongswan executables.

Table 17 - Pre-Operational Self-Tests

The pre-operational software integrity tests are performed automatically when the module is powered on, before the module transitions into the operational state. While the module is executing the self-tests, services are not available, and data output (via the data output interface) is inhibited until the tests are successfully completed. The module transitions to the operational state only after the pre-operational self-tests are passed successfully.

10.2. Conditional Self-Tests

The module performs the self-test on the following Approved cryptographic algorithm supported in Approved mode using the Known Answer Test (KAT) shown below:

Algorithm	Implementation	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
SP800-135r1 IKEv2 KDF (CVL)	C	HMAC-SHA-1	KAT	CAST	Module is operational	KDF used for IKEv2	Module initialization
OpenSSL Bound Module							
HMAC SHA-256	C	256 bit keys	KAT	CAST	Module is operational	Message authentication	Module initialization
SHA-1	C	0-8184 bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
SHA-512	C	0-8184 bit messages	KAT	CAST	Module is operational	Message digest	Module initialization
AES-GCM	C	256-bit key	KAT	CAST	Module is operational	Encryption, Decryption (separately)	Module initialization
CTR_DRBG	C	128 bit key with derivation function and prediction resistance	KAT	CAST	Module is operational	DRBG generation and reseed Health tests according to section 11.3 of [SP800- 90Ar1]	Module initialization
KAS-FFC-SSC	C	ffdhe2048	KAT	CAST	Module is operational	Shared secret computation	Module initialization

Algorithm	Implementation	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
KAS-ECC-SSC	C	P-256	KAT	CAST	Module is operational	Shared secret computation	Module initialization
RSA SigGen/Sigver	C	PKCS#1 v1.5 with SHA-256 and 2048-bit key	KAT	CAST	Module is operational	Signature generation and Signature verification	Module initialization
ECDSA SigGen/Sigver	C	SHA-256 and P-224, B-233	KAT	CAST	Module is operational	Signature generation and Signature verification	Module initialization
DH KeyGen (Safe Primes)	C	ffdhe2048	PCT	PCT	Module is operational	SP800-56Ar3 Section 5.6.2.1.4	Key pair generation
RSA KeyGen	C	PKCS#1 v1.5 padding SHA-256	PCT	PCT	Module is operational	Signature generation and Signature verification	Key pair generation
ECDSA KeyGen	C	SHA-256	PCT	PCT	Module is operational	Signature generation and Signature verification	Key pair generation
Kernel Bound Module							
HMAC SHA-256	C	32, 160, 1048 bit keys	KAT	CAST	Module becomes operational	Self-test for the algorithm provided by the bound Kernel Crypto API module used for the integrity test	Module initialization

Table 18 - Conditional Self-Tests

For the KAT, the module calculates the result and compares it with the known answer. If the calculated value does not match the known answer, the KAT fails and the module enters the error state.

The module performs self-tests on all approved cryptographic algorithms as part of the approved services supported in the approved mode of operation, using the tests shown in Table 18. Services are not available, and data output (via the data output interface) is inhibited during the self-tests. If any of these tests fails, the module transitions to the error state.

10.3. Periodic Self-Tests

The module does not implement any periodic self-tests.

10.4. Error States

If the module fails any of the self-tests, it will return an error message to indicate the error and then enter the error state. In the error state, the module immediately stops functioning and ends the application process. Consequently, the data output interface is inhibited, and the module accepts no more inputs or requests (as the module is no longer running). The following table shows the list of error messages when the module fails any self-test.

Name	Description	Conditions	Recovery Method	Indicator
Error State	Strongswan stops executing	Integrity Test failure	Restart of the module	ipsec: strongswan fips file integrity check failed
		KAT failure		ipsec: strongswan fips ikev2 kdf self-test failed
		Self-test failure in Canonical Ltd. Ubuntu 22.04 OpenSSL Cryptographic Module		*ipsec start* command returns error code 2 with the following cause 'unable to load OpenSSL FIPS provider'. PCT Failure: *ipsec status* command returns error code 1 or 2 and the OpenSSL bound module's OSSL_PROV_PARAM_STATUS is set to 0.
		Self-test failure in Canonical Ltd. Ubuntu 22.04 Kernel Crypto API Cryptographic Module		Kernel panics with message failure “alg: <algo_name>: test failed”.

Table 19 - Error States

To recover from the error state, the module must be restarted and perform self-tests again. If the failure persists, the module must be reinstalled.

Note: Self-test failures in the bound Canonical Ltd. Ubuntu 22.04 Kernel Crypto API Cryptographic Module or Canonical Ltd. Ubuntu 22.04 OpenSSL Cryptographic Module will prevent the Canonical Ltd. Ubuntu 22.04 Strongswan Cryptographic Module from operating.

10.5. Operator Initiation

The software integrity tests, cryptographic algorithm self-tests, and entropy source start-up tests can be invoked on demand by unloading and subsequently re-initializing the module. On-demand self-tests can be invoked by powering off and reloading the module, which cause the module to run the self-tests again. During the execution of the on-demand self-tests, services are not available and no data output or input is possible.

11. Life-Cycle Assurance

11.1. Startup Procedures

The module is distributed as a part of Ubuntu 22.04 in the form of the following deb packages:

- strongswan=5.9.5-2ubuntu2.1+Fips1
- strongswan-charon=5.9.5-2ubuntu2.1+Fips1
- strongswan-hmac=5.9.5-2ubuntu2.1+Fips1
- strongswan-libcharon=5.9.5-2ubuntu2.1+Fips1
- strongswan-starter=5.9.5-2ubuntu2.1+Fips1
- libstrongswan=5.9.5-2ubuntu2.1+Fips1
- libstrongswan-standard-plugins=5.9.5-2ubuntu2.1+Fips1
- libcharon-extauth-plugins_5.9.5-2ubuntu2.1+Fips1

In order to configure the operating environment, the following bound modules must be installed:

- Canonical Ltd. Ubuntu 22.04 Kernel Crypto API Cryptographic Module
- Canonical Ltd. Ubuntu 22.04 OpenSSL Cryptographic Module

Please follow the instructions provided in the respective security policies to install and configure both modules in the Approved mode of operation.

Once these modules are installed and configured properly, the operating environment is configured to support Approved operation. The Crypto Officer should check the existence of the file /proc/sys/crypto/fips_enabled, which content should be the character “1”. If the file does not exist or does not contain “1”, the operating environment is not configured to support Approved mode and the module will not operate in the Approved mode properly.

11.1.1. Operating Environment Configurations

The module needs to be set to run in the Approved mode configuration. This can be enabled automatically via the Ubuntu Advantage tool after attaching your subscription.

- To install the tool type the following commands:

$ sudo apt update

$ sudo apt install ubuntu-advantage-tools

- To activate the Ubuntu Pro subscription run:

$ sudo pro attach 'your_pro_token'

- To enable Approved mode run:

$ sudo pro enable fips

- To verify that Approved mode is enabled run:

$ sudo pro status

The pro client will install the necessary packages for the Approved mode, including the kernel and the bootloader. After this step you MUST reboot to put the system into Approved mode. The reboot will boot into the FIPS supported kernel and create the /proc/sys/crypto/fips_enabled entry which tells the FIPS certified modules to run in Approved mode. If you do not reboot after installing and configuring the bootloader, Approved mode is not yet enabled.

To verify that Approved mode 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 modules will not run in Approved mode. If the file is missing, the FIPS kernel is not installed, you can verify that FIPS has been properly enabled with the pro status command.

The module can only operate in Approved mode as stated in section 3.2. With the operational environment setup as stated in the above section, the following restrictions are applicable. No more cipher addition is possible by configuration or command line options.

Configure Charon as specified in ipsec.conf(5), and ipsec.secrets(5) man pages

- To start and stop the module, use the *ipsec* command.
- IKEv2 must be used In order to run the module in Approved mode of operation, the following setting must be included in the ipsec.conf file:

  keyexchange = ikev2
- ikelifetime should not be larger than 1 hour:

  ikelifetime = 1h
- salifetime should not be larger than 1 hour:

  salifetime = 1h
- Galois Counter Mode (GCM) should be used with their full tag lengths.
- Aggressive mode should not be used.
- Stopping the module will zeroize the SSP (e.g., ephemeral key).
- To check module status, read the Charon debug data using the *ipsec statusall* command line and the logs in /var/log/charon.log.

NOTE: Encryption and decryption of data is done implicitly when the kernel triggers Charon to set up a new Security Association.

11.1.2. Module Installation

Once the operating environment is configured following the instructions provided in Section 11.1.1, and configuration to access the PPA is complete, the Crypto Officer can install the Ubuntu packages containing the module using the Advanced Package Tool (APT), for example with the following command line:

$ sudo apt-get install strongswan strongswan-charon strongswan-hmac strongswan-libcharon strongswan-starter libstrongswan libstrongswan-standard-plugins

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.

After the packages are installed, the Crypto Officer must execute the *ipsec version* command. The Crypto Officer must ensure that the proper name and version are listed in the output as follows:

Linux strongSwan U5.9.5/K5.15.0-70-fips

University of Applied Sciences Rapperswil, Switzerland

FIPS module name: Canonical Ltd. Ubuntu 22.04 Strongswan Cryptographic Module

FIPS module version: 5.9.5-2ubuntu2.1+Fips1

See 'ipsec --copyright' for copyright information.

11.2. Administrator Guidance

11.2.1. Managing the IKEv2 Daemon

To start the IKEv2 daemon, use the following command:

 # ipsec start

To stop the IKEv2 daemon, use the following command:

 # ipsec stop

To start the IKEv2 daemon automatically at the system boot time, use the following command:

 # systemctl enable strongswan

To prevent the IKEv2 daemon from automatically starting, use the following command:

 # systemctl disable strongswan

See the ipsec(8), ipsec.conf(5) and ipsec.secrets(5) man pages for more information about how to operate the module.

11.2.2. AES GCM IV

The Canonical Ltd. Ubuntu 22.04 Strongswan Cryptographic Module is bound to the Canonical Ltd. Ubuntu 22.04 OpenSSL Cryptographic Module which implements AES-GCM. This module, Strongswan, generates the IV for AES-GCM in OpenSSL through the IKEv2 key establishment protocol which is compliant with IG C.H provision 1) (b), "IPSec protocol IV generation".

The AES-GCM IV generation is in compliance with [RFC5282]. The module uses the [RFC7296] compliant IKEv2 protocol to establish the shared secret SKEYSEED from which the AES-GCM encryption keys are derived.

By the virtue of the lifetime limit (see above Section 11.1.1), the IV is renegotiated before reaching 2^64. The IV does not get stored permanently. In case or normal or abnormal termination of the IKE connection, the SA has to be renegotiated by the module.

In the event the module's power is lost and restored, the operator must ensure that a new key for use with the AES GCM key encryption or decryption under this scenario shall be established.

11.2.3. RSA Signatures

To meet the requirement stated in IG C.F, the bound OpenSSL module implements only the FIPS 186-4 approved modulus sizes of 2048, 3072, and 4096 bits for signature generation. For signature verification, the bound OpenSSL module implements only the FIPS 186-4 approved modulus sizes of 1024, 2048, 3072, and 4096 bits. Each modulus size was tested, and corresponding certificates can be found detailed in Section 2 for the bound OpenSSL module.

11.2.4. Compliance to SP 800-56ARev3 Assurances

The bound OpenSSL module offers DH and ECDH shared secret computation services compliant to the SP 800-56ARev3. The Canonical Ltd. Ubuntu 22.04 Strongswan Cryptographic Module implements the NIST SP 800-135 IKEv2 KDF (CVL) part of the key agreement using the HMAC portion of the SSP agreement, while the bound OpenSSL module provides the SP 800-56Arev3-compliant DH and ECDH shared secret computation. Therefore, the module meets the requirements of IG D.F scenario 2 (2). In order to meet the required assurances listed in section 5.6 of SP 800-56ARev3, the following steps are performed.

1. The entity using the bound OpenSSL module, must use the bound OpenSSL module's "Key pair generation" service for generating DH/ECDH ephemeral keys. This meets the assurances required by key pair owner defined in the section 5.6.2.1 of SP 800-56ARev3.
2. As part of the bound OpenSSL module's shared secret computation (SSC) service, the bound OpenSSL module internally performs the public key validation on the peer's public key passed in as input to the SSC function. This meets the public key validity assurance required by the sections 5.6.2.2.1/5.6.2.2.2 of SP 800-56ARev3.
3. The bound OpenSSL module does not support static keys therefore the "assurance of peer's possession of private key" is not applicable.

11.2.5. Legacy Algorithms

The module utilizes the following legacy algorithms from the bound OpenSSL module, as defined in SP 800-131Arev2:

- SHA-1 for RSA Signature Verification and ECDSA Signature Verification purposes.

11.3. Non-Administrator Guidance

There is no non-administrator guidance.

11.4. Maintenance Requirements

There are no maintenance requirements.

11.5. End of Life

As the module does not persistently store SSPs, secure sanitization of the module consists of unloading the module. This will zeroize all SSPs in volatile memory. Then, the Ubuntu packages can be uninstalled from the Ubuntu 22.04 system.

12. Mitigation of Other Attacks

The module does not implement security mechanisms to mitigate other attacks.

Appendix A. Glossary and Abbreviations

AES	Advanced Encryption Standard
API	Application Programming Interface
CAST	Cryptographic Algorithm Self-Test
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CKG	Cryptographic Key Generation
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
ENT (NP)	Non-physical Entropy Source
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standards
GCM	Galois Counter Mode
GMAC	Galois Counter Mode Message Authentication Code
HMAC	Keyed-Hash Message Authentication Code
IG	Implementation Guidance
IKE	Internet Key Exchange
IPSEC	Internet Protocol Security
KAS	Key Agreement Scheme
KAT	Known Answer Test
NIST	National Institute of Science and Technology
PAA	Processor Algorithm Acceleration
PCT	Pair-wise Consistency Test
PKCS	Public-Key Cryptography Standards
PSS	Probabilistic Signature Scheme
RAM	Random Access Memory
RNG	Random Number Generator
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
SSC	Shared Secret Computation
SSH	Secure Shell
SSP	Sensitive Security Parameter
TLS	Transport Layer Security
TOEPP	Tested Operational Environment's Physical Perimeter

Appendix B. References

FIPS 140-3	**FIPS PUB 140-3 - Security Requirements For Cryptographic Modules** March 2019 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-3.pdf
FIPS 140-3 IG	**Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program** https://csrc.nist.gov/Projects/cryptographic-module-validation-program/fips-140-3-ig-announcements
FIPS 180-4	**Secure Hash Standard (SHS)** March 2012 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
FIPS 186-4	**Digital Signature Standard (DSS)** February 2013 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
FIPS 186-5	**Digital Signature Standard (DSS)** February 2023 https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf
FIPS 197	**Advanced Encryption Standard** November 2001 https://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
FIPS 198-1	**The Keyed Hash Message Authentication Code (HMAC)** July 2008 https://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf
PKCS#1	**Public Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1** February 2003 http://www.ietf.org/rfc/rfc3447.txt
RFC 3526	**More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)** May 2003 https://www.ietf.org/rfc/rfc3526.txt
SP 800-38A	**Recommendation for Block Cipher Modes of Operation Methods and Techniques** December 2001 https://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf
SP 800-38A Addendum	**Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode** October 2010 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a-add.pdf
SP 800-38C	**Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality** May 2004 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf

SP 800-38D	**Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC** November 2007 https://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf
SP 800-38F	**Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping** December 2012 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf
SP 800-56Ar3	**Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography** April 2018 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf
SP 800-90Ar1	**Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf
SP 800-90B	**Recommendation for the Entropy Sources Used for Random Bit Generation** January 2018 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf
SP 800-133r2	**Recommendation for Cryptographic Key Generation** June 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133r2.pdf
SP 800-135r1	**Recommendation for Existing Application-Specific Key Derivation Functions** December 2011 https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-135r1.pdf
SP 800-140B	CMVP Security Policy Requirements March 2020 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-140B.pdf