FIPS 140-3 Non-Proprietary Security Policy

SUSE LLC.

SUSE Rancher Kubernetes Cryptographic Library

Software Version:
2.0

Date: October 14, 2024

Prepared by:

Corsec Security, Inc.
12600 Fair Lakes Circle, Suite 210
Fairfax, VA 22033
United States of America

Phone: +1 703 267 6050
www.corsec.com

Introduction

Federal Information Processing Standards Publication 140-3 — Security Requirements for Cryptographic Modules specifies requirements for cryptographic modules to be deployed in a Sensitive but Unclassified environment. The National Institute of Standards and Technology (NIST) and Canadian Centre for Cyber Security (CCCS) Cryptographic Module Validation Program (CMVP) run the FIPS 140-3 program. The NVLAP accredits independent testing labs to perform FIPS 140-3 testing; the CMVP validates modules meeting FIPS 140-3 validation. Validated is the term given to a module that is documented and tested against the FIPS 140-3 criteria.

Additional information is available on the CMVP website at:
https://csrc.nist.gov/projects/cryptographic-module-validation-program

About this Document

This non-proprietary Cryptographic Module Security Policy for the SUSE Rancher Kubernetes Cryptographic Library from SUSE LLC. provides an overview of the product and a high-level description of how it meets the overall Level 1 security requirements of FIPS 140-3.

The SUSE Rancher Kubernetes Cryptographic Library is also referenced in this document as the “module.”

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design, and manufacturing. SUSE LLC. shall have no liability for any error or damages of any kind resulting from the use of this document.

Notices

Table of Contents

List of Tables

List of Figures

1. General

This document describes SUSE LLC.'s cryptographic module Security Policy (SP) for the SUSE Rancher Kubernetes Cryptographic Library (Software version: 2.0) cryptographic module (also referred to as the "module" hereafter). It contains specification of the security rules under which the cryptographic module operates, including the security rules derived from the requirements of the FIPS 140-3 standard. The module is a software module and has a Multi-Chip Stand Alone embodiment.

The module meets the overall Level 1 security requirements of FIPS 140-3. The following table lists the level of validation for each area in FIPS 140-3:

ISO/IEC 24759 Section 6.	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	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

Table 1 - Security Levels

2. Cryptographic Module Specification

The SUSE Rancher Kubernetes Cryptographic Library from SUSE LLC. is an open source software library that contains cryptography to serve SUSE's Rancher Kubernetes Engine and its ecosystem of supported cloud-native tools written in the Go programming language. The module is intended for use in environments specified in Table 2 below and any general-purpose environment that requires cryptographic primitives. The Tested Operational Environment's Physical Perimeter (TOEPP) of the module is the physical perimeter of the tested environment, which is listed in Table 2 below. The module is a software module and has a Multi-Chip Stand Alone embodiment. The installation instructions are provided in Section 11 of this document.

The boundary of the module is defined as a single object file, bcm.o. The module version is 2.0. The module was tested on the following operational environments:

#	Operating System	Hardware Platform	Processor	PAA/Acceleration
1	Android 13	Google Pixel 7 Pro	Google Tensor G2 64-bit and 32-bit	With PAA
2	Android 13	Google Pixel 7 Pro	Google Tensor G2 64-bit and 32-bit	Without PAA
3	Android 13	Google Pixel 6 Pro	Google Tensor 64-bit and 32-bit	With PAA
4	Android 13	Google Pixel 6 Pro	Google Tensor 64-bit and 32-bit	Without PAA
5	Android 13	Google Pixel 5a	Qualcomm Snapdragon 765 64-bit and 32-bit	With PAA
6	Android 13	Google Pixel 5a	Qualcomm Snapdragon 765 64-bit and 32-bit	Without PAA
7	Android 13	Google Pixel 4a	Qualcomm Snapdragon 730 64-bit and 32-bit	With PAA
8	Android 13	Google Pixel 4a	Qualcomm Snapdragon 730 64-bit and 32-bit	Without PAA
9	Android 13	Google Pixel 4XL	Qualcomm Snapdragon 855 64- bit and 32-bit	With PAA
10	Android 13	Google Pixel 4XL	Qualcomm Snapdragon 855 64- bit and 32-bit	Without PAA
11	Google Prodimage with Linux 5.10.120	IN762	IN762	With PAA
12	Google Prodimage with Linux 5.10.120	IN762	IN762	Without PAA

#	Operating System	Hardware Platform	Processor	PAA/Acceleration
13	Google Prodimage with Linux 4.15.0	Tau t2a	Ampere Altra	With PAA
14	Google Prodimage with Linux 4.15.0	Tau t2a	Ampere Altra	Without PAA
15	Debian Linux 5.17.11 (Rodete)	n2d	AMD EPYC 7B12	With PAA
16	Debian Linux 5.17.11 (Rodete)	n2d	AMD EPYC 7B12	Without PAA
17	Google Prodimage with Linux 4.15.0	n1	Intel Xeon E5 2696 v4	With PAA
18	Google Prodimage with Linux 4.15.0	n1	Intel Xeon E5 2696 v4	Without PAA

Table 2 - Tested Operational Environments

The cryptographic module is also supported on the following operational environments for which operational testing and algorithm testing was not performed. The CMVP makes no statement as to the correct operation of the module on the operational environments for which operational testing was not performed.

#	Operating System	Hardware Platform
1	Linux 4.X	x86_64 architecture ARMv7 architecture ARMv8 architecture
2	Linux 5.X	X86_64 architecture ARMv7 architecture ARMv8 architecture
3	Linux 6.X	x86_64 architecture ARMv7 architecture ARMv8 architecture

Table 3 - Vendor Affirmed Operational Environments

Table 4 below lists all the approved algorithms implemented in the module:

CAVP Cert[1]	Algorithm and Standard	Mode/Method	Description / Key Size(s) / Key Strength(s)	Use / Function
A2811	AES FIPS 197 SP800-38A	CBC, ECB, CTR	Key sizes: 128, 192, 256 bits; Strength: 128, 192, 256 bits	Encryption, Decryption
A2811	AES FIPS 197 SP800-38D	GCM	Key sizes: 128, 192, 256 bits; Strength: 128, 192, 256 bits	Authenticated Encryption, Authenticated Decryption
A2811	AES FIPS 197 SP800-38C	CCM	Key size: 128 bits; Strength: 128 bits	Authenticated Encryption, Authenticated Decryption
A2811	AES, KTS FIPS 197 SP800-38F	KW, KWP	Key sizes: 128, 192, 256 bits; Strength: 128, 192, 256 bits	Key Transport per IG D.G Key establishment methodology provides between 128 and 256 bits of encryption strength
CVL A2811	TLS v1.0/1.1 and v1.2 KDF[2] SP800-135rev1	N/A	SHA2-256, SHA2-384, SHA2-512; Strength: 256, 384, 512 bits	Key Derivation
Vendor Affirmed	CKG	SP800-133rev2	Cryptographic Key Generation: Section 5: Generation of Key Pairs for Asymmetric-Key Algorithms, Section 6.1: The "Direct Generation" of Symmetric Keys	Key Generation Symmetric keys and seeds are generated as the direct output of the DRBG
A2811	DRBG SP800-90Arev1	CTR_DRBG	AES-256; Key size: 256 bits; Strength: 256 bits	Random Bit Generation

[1] There are algorithms that have been CAVP-tested on the same certificate but are not used by any approved service of the module Only the algorithms, modes/methods, and key lengths/curves/moduli shown in this table are used by an approved service of the module.

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

CAVP Cert[1]	Algorithm and Standard	Mode/Method	Description / Key Size(s) / Key Strength(s)	Use / Function
A2811	ECDSA FIPS 186-4	Key Pair Generation, Signature Generation, Signature Verification, Public Key Validation	P-224, P-256, P-384, P-521; Strength: 112, 128, 192, 256 bits	Digital Signature Services
A2811	HMAC FIPS 198-1	Generate, Verify	HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512; Strength: 128, 192, 256, 384, 512 bits	Generation, Authentication
A2811	RSA FIPS 186-4	Key Generation, Signature Generation, Signature Verification PKCS 1.5 and PSS	1024, 2048, 3072, 4096; Strength: 80, 112, 128, 152 bits; Note: Key size 1024 should be only used for Signature Verification	Digital Signature Services
A2811	SHA FIPS 180-4	Hashing	SHA-1[3], SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/256; Strength: 80, 112, 128, 192, 256, 128 bits	Digital Signature Generation, Digital Signature Verification, Non-Digital Signature Applications
A2811	KAS-SSC SP800-56Arev3	KAS-ECC-SSC ephemeralUnified	ECC: P-224, P-256, P-384 and P-521; Strength: 112, 128, 192, 256 bits	Key Agreement Scheme Shared Secret Computation per SP800-56Arev3; Key establishment methodology provides between 112 and 256 bits of security strength

Table 4 - Approved Algorithms

[3] Used for non-digital signature applications or to verify existing digital signatures only

Algorithm	Caveat	Use / Function
MD5	As allowed per SP800-135rev1 (No security claimed)	When used with the TLS protocol version 1.0 and 1.1

Table 5 - Non-Approved, Allowed Algorithms with No Security Claimed

Algorithm/Function	Use/Function
MD5, MD4	Non-Approved hashing
POLYVAL	Non-Approved authenticated encryption
DES, Triple-DES (non-compliant)	Non-Approved encryption/decryption
AES-GCM-SIV (non-compliant)	Non-Approved encryption/decryption
DH (non-compliant)	Non-Approved key agreement

Table 6 - Non-Approved, Not Allowed Algorithms

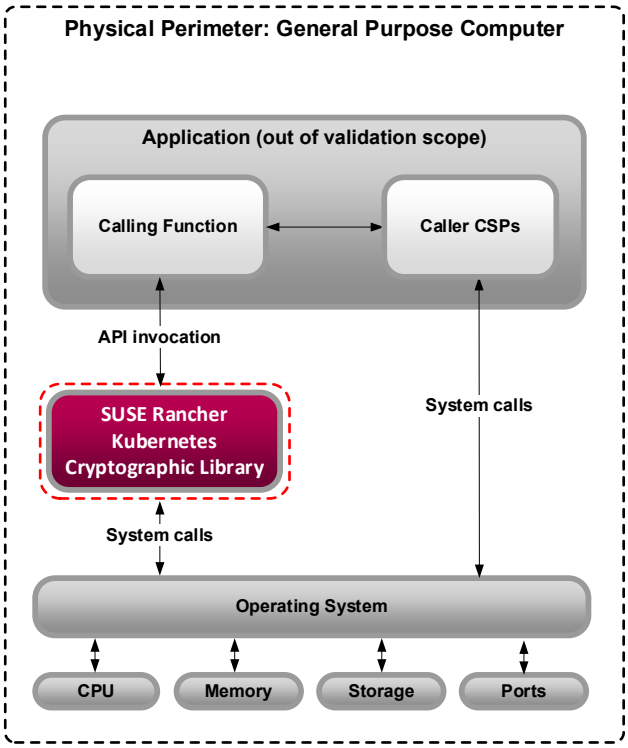

Figure 1 - Module Boundary

2.1 Overall Security Design and Rules of Operation

2.1.1 Usage of AES-GCM

AES GCM encryption and decryption are used in the context of the TLS protocol version 1.2 (compliant to Scenario 1a in *FIPS 140-3 IG C.H*). The module is compliant with *NIST SP 800-52* and the mechanism for IV generation is compliant with *RFC 5288*. The module ensures that it is strictly increasing and thus cannot repeat. When the IV exhausts the maximum number of possible values for a given session key, the first party (client or server) to encounter this condition may either trigger a handshake to establish a new encryption key in accordance with *RFC 5246* or fail. In either case, the module prevents any IV duplication and thus enforces the security property.

The module's IV is generated internally by the module's Approved DRBG, which is internal to the module's boundary. The IV is 96 bits in length per *NIST SP 800-38D*, Section 8.2.2 and *FIPS 140-3 IG C.H* scenario 2.

The selection of the IV construction method is the responsibility of the user of this cryptographic module. In approved mode, users of the module must not utilize GCM with an externally generated IV. Per *FIPS 140-3 IG C.H*, in the event module power is lost and restored, the consuming application must ensure that any of its AES-GCM keys used for encryption or decryption are re-distributed.

The module implements the TLS 1.2 KDF and other cryptographic primitives used in TLS 1.2, but does not implement the TLS 1.2 protocol itself.

2.1.2 RSA and ECDSA Keys

The module allows the use of 1024-bit RSA keys for legacy purposes including signature generation, which is disallowed in Approved mode as per *NIST SP 800-131Arev2*. Therefore, cryptographic operations with the Non-Approved key sizes will result in the module operating in Non-Approved mode.

The elliptic curves utilized shall be the validated NIST-recommended curves and shall provide a minimum of 112 bits of encryption strength.

2.1.3 CSP Sharing

Non-Approved cryptographic algorithms shall not share the same key or CSP as an approved algorithm. As such, Approved algorithms shall not use the keys generated by the module's Non-Approved key generation methods or the converse.

2.1.4 Modes of Operation

The module supports two modes of operation: Approved and Non-approved. The module will be in approved mode when all self-tests have completed successfully, and only Approved algorithms are invoked. See Table 4 above for a list of the supported Approved algorithms. The non-Approved mode is entered when a non-Approved algorithm is invoked. See Table 6 for a list of non-Approved algorithms.

3. Cryptographic Module Interfaces

The Data Input interface consists of the input parameters of the API functions. The Data Output interface consists of the output parameters of the API functions. The Control Input interface consists of the actual API input parameters. The Status Output interface includes the return values of the API functions.

Logical interface	Data that passes over port/interface
Data Input	API input parameters
Data Output	API output parameters and return values
Control Input	API input parameters
Status Output	API return values

Table 7 - Ports and Interfaces

The module does not implement a power input interface or a control output interface. As a software module, control of the physical ports is outside the module scope. However, when the module is performing self-tests, or is in an error state, all output on the module's logical data output interfaces is inhibited.

4. Roles, Services, and Authentication

4.1 Roles

The cryptographic module only implements a Crypto Officer (CO) role. The CO role is implicitly assumed by the entity accessing services implemented by the module. An operator is considered the owner of the thread that instantiates the module and, therefore, only one operator is allowed, and no concurrent operators are allowed.

4.2 Authentication

The module does not support operator authentication.

4.3 Services

The Approved services supported by the module and access rights within services accessible over the module's public interface are listed in Table 8 below.

Role	Service	Input	Output
CO	Symmetric Encryption	Plaintext, encryption key	Return code, ciphertext
CO	Symmetric Decryption	Ciphertext, decryption key	Return code, plaintext
CO	Keyed Hashing	Message, key	Return code, Message Authentication Code
CO	Hashing	Message	Return code, hash
CO	Random Bit Generation	API call parameters	Return code, random bits
CO	Signature Generation	Message, signing key	Return code, signature
CO	Signature Verification	Signature, verification key	Return code
CO	Key Transport	API call parameters, wrapping key	Return code, wrapped key
CO	Key Agreement	API call parameters	Return code, shared secret
CO	TLS Key Derivation	API call parameters, TLS pre-master secret	Return code, TLS Key
CO	Key Generation	API call parameters	Return code, key pair
CO	Key Verification	API call parameters, key pair	Return code
CO	On-Demand Self-Test	N/A	Return code
CO	Zeroization	N/A	N/A
CO	Show Status	API call parameters	Return code, status

Table 8 - Roles, Service Commands, Input and Output

Approved services are listed in Table 9. The SSPs listed in the table indicate the access required using below notation:

G = Generate: The module generates or derives the SSP.
R = Read: The SSP is read from the module (e.g., the SSP is output).

W = Write: The SSP is updated, imported, or written to the module.
E = Execute: The module uses the SSP in performing a cryptographic operation.
Z = Zeroize: The module zeroizes the SSP.

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access Rights to Keys and/or SSPs	Indicator
Symmetric Encryption	Perform symmetric encryption operations	AES CBC, ECB, CTR, CCM (Cert. #A2811) CKG	AES Key, AES-GCM Key	CO	W, E	1
Symmetric Decryption	Perform symmetric decryption operations	AES CBC, ECB, CTR, GCM, CCM (Cert. #A2811) CKG	AES Key, AES-GCM Key, AES-GCM IV	CO	W, E	1
Keyed Hashing	Perform keyed hashing operations	HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512 (Cert. #A2811)	HMAC Key	CO	W, E	1
Hashing	Perform hashing operations	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/256 (Cert. #A2811)	N/A	CO	N/A	1
Random Bit Generation	Generate random numbers	CTR_DRBG (Cert. #A2811) CKG	DRBG Seed, CTR_DRBG V, CTR_DRBG Key	CO	G, E	1
			DRBG output	CO	G, R	
			CTR_DRBG Entropy Input	CO	W, E	

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access Rights to Keys and/or SSPs	Indicator
Signature Generation	Perform signing operations	CTR_DRBG, RSA SigGen, ECDSA SigGen (Cert. #A2811)	RSA Signature Generation Key, ECDSA Signing Key	CO	G, W, E	1
Signature Verification	Perform verification operations	RSA SigVer, ECDSA SigVer (Cert. #A2811)	RSA Signature Verification Key, ECDSA Verification Key	CO	G, W, E	1
Key Transport	Perform key encryption operations; KTS using AES-KW, AES-KWP per IG D.G	AES KW, KWP (Cert. #A2811) CKG	AES Wrapping Key	CO	W, E	1
Key Agreement	Perform key agreement operations	KAS-ECC-SSC (Cert. #A2811)	EC DH Private Key, EC DH Public Key	CO	G, W, E	1
			Shared Secret		G	
TLS Key Derivation	Perform key derivation operations	TLS KDF (Cert. #A2811)	TLS Pre-Master Secret	CO	W, E	1
			TLS Master Secret		G, E	
Key Generation	Perform generation operations	CTR_DRBG, RSA KeyGen, ECDSA KeyGen (Cert. #A2811) CKG	RSA Signature Generation Key, ECDSA Signing Key	CO	G, W, E	1
Key Verification	Perform key pair verification operations	ECDSA KeyVer (Cert. #A2811)	ECDSA Signing Key, ECDSA Verification Key	CO	G, W, E	1
On-Demand Self-Test	Execute self-tests on demand	N/A	N/A	CO	N/A	1

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access Rights to Keys and/or SSPs	Indicator
Zeroization	Zeroize all SSPs	N/A	All SSPs	CO	Z	N/A
Show Status	Obtain the module status and versioning information	N/A	N/A	CO	N/A	N/A

Table 9 - Approved Services

Non-Approved Services are listed in the Table 10 below:

Service	Description	Algorithms Accessed	Role	Indicator
Hashing (as allowed per SP800-135rev1)	Perform hashing operations when used with the TLS protocol version 1.0 and 1.1	MD5	CO	0
Hashing	Perform hashing operations	MD4	CO	0
Hashing	Used as part of AES-GCM-SIV	POLYVAL	CO	0
Symmetric encryption/decryption	Perform symmetric encryption and/or decryption operations	DES Triple-DES AES	CO	0
Key Generation	Perform generation operations	DH	CO	0
RSA Primitives (RSADP, RSAEP, RSASP, RSAVP)	Perform RSA related primitive operations (decrypt, encrypt, sign, verify)	RSA	CO	0

Table 10 - Non-Approved Services

5. Software/Firmware Security

The pre-operational integrity test is performed using HMAC-SHA2-256. The integrity test can be executed on demand by power-cycling the host platform and reloading the module. The module does not support software loading. Please refer to Section 11.1 for instructions on compiling the source code into executable.

5.1 Module Format

The form of the module is a single object file, bcm.o.

6. Operational Environment

The module runs on a GPC, which is a modifiable operational environment, running one of the operating systems specified in Table 2. Each approved operating system manages processes and threads in a logically separated manner. The module's user is considered the owner of the calling application that instantiates the module.

No specific security rules, settings or restrictions to the configuration of the operational environment applies to the module. The module is designed to ensure that all the self-tests are initiated automatically when the module is loaded.

7. Physical Security

As a software module, the physical security requirements are not applicable.

8. Non-invasive Security

The module does not claim any non-invasive security measures.

9. Sensitive Security Parameter Management

All the SSPs are zeroized implicitly when host platform is restarted. The various SSPs used by the module are listed in Table 11 below.

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import/ Export	Establishment	Storage	Zeroisation	Use & Related Keys
AES Key (CSP)	128/192/256 bits	AES-CBC, ECB, CTR, CCM A2811	External	Input via API in plaintext (Electronic Entry)	N/A	Plaintext in RAM	Power-cycle host	AES encrypt / decrypt
AES-GCM Key (CSP)	128/192/256 bits	AES-GCM A2811	External	Input via API in plaintext (Electronic Entry)	N/A	Plaintext in RAM	Power-cycle host	AES decrypt / verify
AES-GCM IV[4] (CSP)	96 bits	AES-GCM A2811	External	Input via API in plaintext (Electronic Entry)	N/A	Plaintext in RAM	Power-cycle host	AES decrypt / verify
AES Wrapping Key (CSP)	128/192/256 bits	AES-KW, AES-KWP A2811	External	Input via API in plaintext (Electronic Entry)	N/A	Plaintext in RAM	Power-cycle Host	AES key wrapping

[4] As specified in Section 2.1.1, usage of externally generated IV is only allowed for AES-GCM decryption in the approved mode of operation.

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import/ Export	Establishment	Storage	Zeroisation	Use & Related Keys
ECDSA Signing Key (CSP)	112/128/192/256 bits	ECDSA SigGen A2811	Internally Generated	Input via API in plaintext (Electronic Entry); Output via API in plaintext (Electronic Entry)	N/A	Plaintext in RAM	Power-cycle host	ECDSA signature generation
ECDSA Verification Key (PSP)	112/128/192/256 bits	ECDSA SigVer A2811	Internally Generated	Input via API in plaintext (Electronic Entry); Output via API in plaintext (Electronic Entry)	N/A	Plaintext in RAM	Power-cycle Host	ECDSA signature verification
EC DH Private Key (CSP)	112/128/192/256 bits	ECDSA KeyGen A2811	Internally Generated	Input via API in plaintext (Electronic Entry); Output via API in plaintext (Electronic Entry)	N/A	Plaintext in RAM	Power-cycle host	Key Agreement

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import/ Export	Establishment	Storage	Zeroisation	Use & Related Keys
EC DH Public Key (PSP)	112/128/192/256 bits	ECDSA KeyGen A2811	Internally Generated	Input via API in plaintext (Electronic Entry); Output via API in plaintext (Electronic Entry)	N/A	Plaintext in RAM	Power-cycle host	Key Agreement
HMAC Key (CSP)	128/192/256/384/512 bits	HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512 A2811	External	Input via API in plaintext (Electronic Entry)	N/A	Plaintext in RAM	Power-cycle host	Keyed hashing
Shared Secret (CSP)	112/128/192/256 bits	KAS-ECC-SSC A2811	Internally Generated	N/A	SP800-56Arev3	Plaintext in RAM	Power-cycle host	Key Agreement
RSA Signature Generation Key (CSP)	112, 128, 152 bits	RSA SigGen A2811	Internally Generated	Input via API in plaintext (Electronic Entry); Output via API in plaintext (Electronic Entry)	N/A	Plaintext in RAM	Power-cycle host	RSA signature generation

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import/ Export	Establishment	Storage	Zeroisation	Use & Related Keys
RSA Signature Verification Key (PSP)	80, 112, 128, 152 bits	RSA SigVer A2811	Internally Generated	Input via API in plaintext (Electronic Entry); Output via API in plaintext (Electronic Entry)	N/A	Plaintext in RAM	Power-cycle host	RSA signature verification
TLS Master Secret (CSP)	384 bits	TLS KDF A2811	Internally Derived via key derivation function defined in SP800-135rev1 KDF (TLS)	N/A	N/A	Plaintext in RAM	Power-cycle host	TLS key derivation
TLS Pre-Master Secret (CSP)	112-256 bits	TLS KDF A2811	External	Input via API in plaintext (Electronic Entry)	N/A	Plaintext in RAM	Power-cycle host	TLS key derivation
DRBG Seed (CSP)	384 bits	CTR_DRBG A2811	Internally Generated	N/A	N/A	Plaintext in RAM	Power-cycle host	DRBG Seeding material
CTR_DRBG V (CSP)	128 bits	CTR_DRBG A2811	Internally Generated	N/A	N/A	Plaintext in RAM	Power-cycle host	DRBG internal state
CTR_DRBG Key (CSP)	256 bits	CTR_DRBG A2811	Internally Generated	N/A	N/A	Plaintext in RAM	Power-cycle host	DRBG internal state

Key/SSP Name/Type	Strength	Security Function and Cert. Number	Generation	Import/ Export	Establishment	Storage	Zeroisation	Use & Related Keys
CTR_DRBG Entropy Input (CSP)	384 bits used as seed, quality of entropy at least 112 bits	CTR_DRBG A2811	External	Input via API in plaintext (Electronic Entry)	N/A	Plaintext in RAM	Power-cycle host	DRBG entropy
DRBG output	2048 bits	CTR_DRBG A2811	Internally Generated	N/A	N/A	Plaintext in RAM	Power-cycle host	Random bits provided for the calling application

Table 11 - SSPs

Entropy sources	Minimum number of bits of entropy	Details
Passive Entropy	112 bits and above	Use of a [SP800-90B] compliant entropy source with at least 256 bits of security strength. Entropy is supplied to the Module via callback functions. The callback functions shall return an error if the minimum entropy strength cannot be met. The caveat “No assurance of the minimum strength of generated SSPs (e.g., keys)” is applicable

Table 12 - Non-Deterministic Random Number Generation Specification

10. Self-Tests

ISO/IEC 19790 requires the module to perform self-tests to ensure the integrity of the module and the correctness of the cryptographic functionality. Some functions also require conditional tests during normal operation of the module. The self-tests can be requested on demand by power cycling the host platform. The module has a single error state, which is called the error state. This state is entered upon failure of a self-test. The module indicates this error state by providing the output status “*** KAT failed” where *** is the algorithm name (example: ECDSA-sign KAT failed). The module can be recovered by terminating execution of the host program and reclamation by the host operating system. The supported tests are listed and described in this section.

10.1 Pre-Operational Self-Tests

Pre-operational self-tests are run upon the initialization of the module and further reboots of the host platform. The CAST (Cryptographic Algorithm Self-Test) for HMAC-SHA2-256 is performed before the integrity test. Self-tests do not require operator intervention to run. If any of the tests fail, the module will not initialize and enter an error state where no services can be accessed.

The module implements the following pre-operational self-tests:

- Software Integrity Test (HMAC-SHA2-256)

10.2 Conditional Self-Tests

Conditional Cryptographic Algorithm Self-Tests (CAST) are run prior to the first use of the cryptographic algorithm. CASTs do not require operator intervention to run. If any of the tests fail, the module will enter an error state, and no services can be accessed.

The module implements the following CASTs:

- ECDSA Signature Generation KAT (P-256)
- ECSDA Signature Verification KAT (P-256)
- RSA Signature Generation KAT (2048 bits, PKCSv1.5)
- RSA Signature Verification KAT (2048 bits, PKCSv1.5)
- SP800-56Arev3 KAS-ECC KAT (P-256)
- AES CBC Encryption KAT (128 bits)
- AES CBC Decryption KAT (128 bits)
- AES-GCM Encryption KAT (128 bits)
- AES-GCM Decryption KAT (128 bits)
- CAST, performed on DRBG, per *NIST SP800-90Arev1* Section 11.3 (Instantiate, Generate, Reseed)
- TLS v1.2 KDF KAT
- SHA-1 KAT
- SHA2-256 KAT
- SHA2-512 KAT
- HMAC-SHA2-256 KAT

Conditional self-tests are run during the module's operation. If any of these tests fail, the module will enter an error state, where no services can be accessed by the operators. The module can be re- initialized to clear the error and resume approved mode of operation.

The module implements the following conditional pairwise consistency tests. The below PCTs are executed whenever a new key pair is generated:

- ECDSA Pairwise Consistency Test
- EC DH Pairwise Consistency Test
- RSA Pairwise Consistency Test

11. Life-Cycle Assurance

The cryptographic module is initialized by loading the module before any cryptographic functionality is available. In User Space, the operating system is responsible for the initialization process and loading of the library. There are no maintenance requirements applicable.

General guidance about the module can be found at https://boringssl.googlesource.com/boringssl. This includes information about the APIs, building and specific information related to FIPS can be found at https://boringssl.googlesource.com/boringssl.git/+/refs/heads/fips-20220613/crypto/fipsmodule/FIPS.md (note this still mentions 140-2, but the information there is the same).

11.1 Installation Instructions

The module is open source. A Linux workstation with the following tools is required to build and compile the module:

Android 13	git 2.23 or later (https://git-scm.com/download/linux) base64, curl, sha256sum (these should come with the Linux installation)
Linux	Clang compiler version 14.0.0 (http://releases.llvm.org/download.html) Go programming language version 1.18.1 (https://golang.org/dl/) Ninja build system version 1.10.2 (https://github.com/ninja- build/ninja/releases) Cmake version 3.22.1 (https://cmake.org/download/)

11.1.1 Building for Android

Once a Linux workstation with the above tools has been obtained, issue the following commands to download and verify repo:

```
curl 'https://gerrit.googlesource.com/git-repo/+/e778e57f11/repo?format=TEXT' | base64 -d > ~/repo
chmod u+x ~/repo
gpg --recv-key 8BB9AD793E8E6153AF0F9A4416530D5E920F5C65
curl https://storage.googleapis.com/git-repo-downloads/repo.asc | gpg --verify - ~/repo
```

Download the manifest from https://ci.android.com/builds/submitted/8918218/aosp_arm64-userdebug/latest/manifest_8918218.xml by clicking the Download button.

Verify the manifest using the following command:

```
Sha256sum ~/manifest_8918218.xml
```

Manually validate that the output from the final command indicates the following expected hash values for this file:

fae7a587167b3b3ebdf5b2c53335a1d1827beddcf23d2788d07f3bbbe9ff7182 manifest_8918218.xml

The module can be obtained by issuing the following commands:

```
mkdir aosp cd
aosp
~/repo init -u https://android.googlesource.com/platform/manifest --depth 1
~/repo init -m ~/manifest_8918218.xml
~/repo sync -q -c -j 20
```

Once downloaded, the module can be built using the following command:

```
. build/envsetup.sh
lunch aosp_arm64-eng
m clean
m test_fips
```

11.1.2 Building for Linux

Once the above tools have been obtained, issue the following command to create a Cmake toolchain file to specify the use of Clang:

```
printf "set(CMAKE_C_COMPILER \"clang\")\nset(CMAKE_CXX_COMPILER \"clang++\")\n" > ${HOME}/toolchain
```

The FIPS 140-3 validated release of the module can be obtained by downloading the tarball containing the source code at the following location:

https://commondatastorage.googleapis.com/chromium-boringssl-fips/boringssl-0c6f40132b828e92ba365c6b7680e32820c63fa7.tar.xz or by issuing the following command:

```
wget https://commondatastorage.googleapis.com/chromium-boringssl-fips/boringssl-
0c6f40132b828e92ba365c6b7680e32820c63fa7.tar.xz
```

The set of files specified in the archive constitutes the complete set of source files of the validated module. There shall be no additions, deletions, or alterations of this set as used during module build.

The downloaded tarball file can be verified using the below SHA2-256 digest value:

62f733289f2d677c2723f556aa58034c438f3a7bbca6c12b156538a88e38da8a

By issuing the following command:

```
sha256sum boringssl-0c6f40132b828e92ba365c6b7680e32820c63fa7.tar.xz
```

The tarball can be extracted using the following command:

```
tar xJ < boringssl-0c6f40132b828e92ba365c6b7680e32820c63fa7.tar.xz
```

After the tarball has been extracted, the following commands will compile the module:

```
cd boringssl
mkdir build && cd build && cmake -GNinja -DCMAKE_TOOLCHAIN_FILE=${HOME}/toolchain -DFIPS=1 -
DCMAKE_BUILD_TYPE=Release ..
ninja && ninja run_tests
```

11.1.3 Retrieving Module Name and Version

The following methods will provide the module name and versions:

- FIPS_module_name() – BoringCrypto (correlating to "**SUSE Rancher Kubernetes Cryptographic Library**" on the validation certificate)
- FIPS_version() – 2022061300 (correlating to "**2.0**" on the validation certificate)

12. Mitigation of Other Attacks

The module is not designed to mitigate attacks which are outside of the scope of FIPS 140-3.

References and Standards

The following Standards are referenced in this Security Policy:

Abbreviation	Full Specification Name
FIPS 140-3	Security Requirements for Cryptographic modules
FIPS 180-4	Secure Hash Standard (SHS)
FIPS 186-4	Digital Signature Standard (DSS)
FIPS 197	Advanced Encryption Standard
FIPS 198-1	The Keyed-Hash Message Authentication Code (HMAC)
IG	Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program
SP 800-38A	Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode
SP 800-38C	Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality
SP 800-38D	Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC
SP 800-38F	Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping
SP 800-52	Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations
SP 800-56A	Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography
SP 800-90A	Recommendation for Random Number Generation Using Deterministic Random Bit Generators
SP 800-131A	Transitioning the Use of Cryptographic Algorithms and Key Lengths
SP 800-133	Recommendation for Cryptographic Key Generation
SP 800-135	Recommendation for Existing Application-Specific Key Derivation Functions

Acronyms

Acronym	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher-Block Chaining
CCCS	Canadian Centre for Cyber Security
CFB	Cipher Feedback
CKG	Cooperative Key Generation
CMVP	Crypto Module Validation Program
CO	Cryptographic Officer
CRNGT	Continuous Random Number Generator Test
CSP	Critical Security Parameter
CTR	Counter-mode
DES	Data Encryption Standard
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
DSS	Digital Signature Standard
EC	Elliptic Curve
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
EC DH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Authority
FIPS	Federal Information Processing Standards
GCM	Galois/Counter Mode
GMAC	Galois Message Authentication Code
GPC	General Purpose Computer
HMAC	Key-Hashed Message Authentication Code
IG	Implementation Guidance
IV	Initialization Vector
KAS	Key Agreement Scheme
KAT	Known Answer Test
KDF	Key Derivation Function
KW	Key Wrap
KWP	Key Wrap with Padding
LLC	Limited Liability Company
MAC	Message Authentication Code
MD4	Message Digest algorithm MD4
MD5	Message Digest algorithm MD5
N/A	Not-Applicable
NIST	National Institute of Standards and Technology
NVLAP	National Voluntary Lab Accreditation Program

OFB	Output Feedback
PAA	Processor Algorithm Accelerator
RAM	Random Access Memory
RFC	Request For Comment
RSA	Rivest Shamir Adleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SP	Special Publication
SSL	Secure Socket Layer
TLS	Transport Layer Security
Triple-DES	Triple Data Encryption Standard