Google, LLC

BoringCrypto

FIPS 140-3 Non-Proprietary Security Policy

Date: September 10. 2025

Table of Contents

List of Tables

List of Figures

1 General

1.1 Overview

This document describes the cryptographic module Security Policy (SP) for the Google, LLC BoringCrypto (software version: 20240407) cryptographic module (also referred to as the “module” hereafter). It contains a 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 meets the overall Level 1 security requirements of FIPS 140-3.

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

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

The Google, LLC BoringCrypto module is an open-source, general-purpose cryptographic library which provides FIPS 140-3 approved cryptographic algorithms to serve BoringSSL and other user-space applications.

Module Type: Software

Module Embodiment: MultiChipStand

Module Characteristics:

Cryptographic Boundary:

The boundary of the module is defined as a single object file, bcm.o, and its instantiation in memory.

Tested Operational Environment's Physical Perimeter (TOEPP):

The TOEPP is the enclosure of the general purpose computer the module is running on.

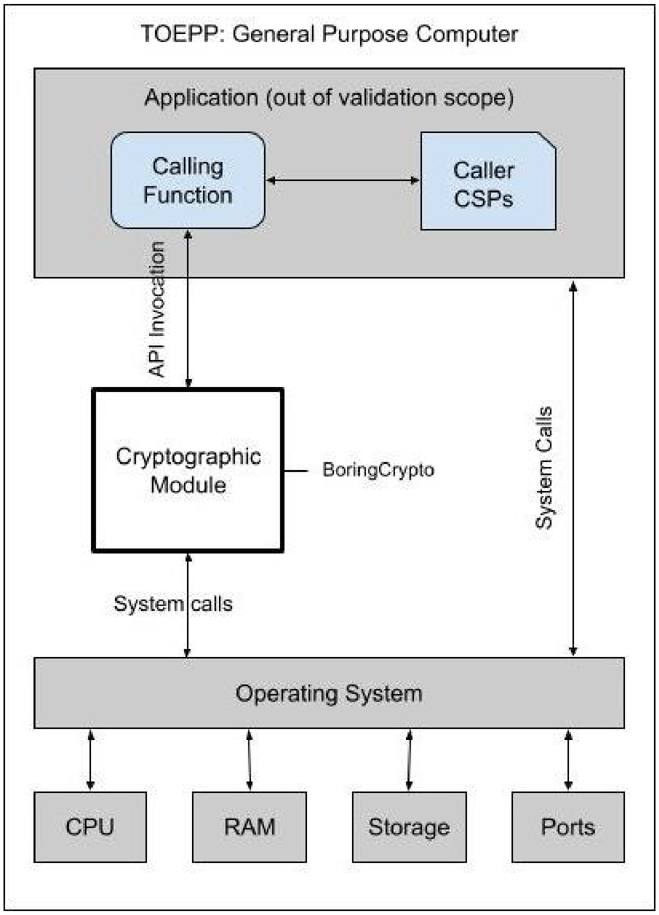

Figure 1: Block Diagram

2.2 Tested and Vendor Affirmed Module Version and Identification

Tested Module Identification – Software:

Package or File Name	Software/ Firmware Version	Features	Integrity Test
bcm.o	20240407		Single encompassing HMAC

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

Tested Operational Environments - Software:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Google Prodimage with Linux 5.10.0	APIF-824	AMD EPYC 7B12	Yes		20240407
Google Prodimage with Linux 5.10.0	APIF-824	AMD EPYC 7B12	No		20240407
Google Prodimage with Linux 5.10.0	APIF-091	ARM Neoverse-N1	Yes		20240407
Google Prodimage with Linux 5.10.0	APIF-091	ARM Neoverse-N1	No		20240407
Google Prodimage with Linux 5.10.0	APIF-738	Intel Xeon 8273CL	Yes		20240407
Google Prodimage with Linux 5.10.0	APIF-738	Intel Xeon 8273CL	No		20240407

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

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

2.3 Excluded Components

The module contains no excluded components.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved	When all self-tests pass and only Approved algorithms are invoked	Approved	Per service indication
Non-Approved	When a non-Approved algorithm is invoked	Non-Approved	Per service indication

Table 4: Modes List and Description

The module supports two modes of operation: Approved and Non-approved. The module will be in approved mode when all power up self-tests have completed successfully, and only Approved algorithms are invoked (see table below). The non-Approved mode is entered when a non-Approved algorithm is invoked (see table below).

Mode Change Instructions and Status:

The module does not enforce a general Approved mode, use the service indicators to determine whether a given service is operated in an Approved mode.

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A5370	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A5370	Key Length - 128	SP 800-38C
AES-CTR	A5370	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A5370	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A5370	Direction - Decrypt, Encrypt IV Generation - External, Internal IV Generation Mode - 8.2.2 Key Length - 128, 192, 256	SP 800-38D
AES-GMAC	A5370	Direction - Decrypt, Encrypt IV Generation - External Key Length - 128, 192, 256	SP 800-38D
AES-KW	A5370	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
AES-KWP	A5370	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
Counter DRBG	A5370	Prediction Resistance - No Mode - AES-256 Derivation Function Enabled - No	SP 800-90A Rev. 1
ECDSA KeyGen (FIPS186-5)	A5370	Curve - P-224, P-256, P-384, P-521 Secret Generation Mode - testing candidates	FIPS 186-5
ECDSA KeyVer (FIPS186-5)	A5370	Curve - P-224, P-256, P-384, P-521	FIPS 186-5
ECDSA SigGen (FIPS186-5)	A5370	Curve - P-224, P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/256	FIPS 186-5
ECDSA SigVer (FIPS186-5)	A5370	Curve - P-224, P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/256	FIPS 186-5
HMAC-SHA-1	A5370	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A5370	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A5370	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A5370	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A5370	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512/256	A5370	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A5370	Domain Parameter Generation Methods - P-224, P-256, P-384, P-521	SP 800-56A Rev. 3

Algorithm	CAVP Cert	Properties	Reference
		Scheme - ephemeralUnified - KAS Role - initiator, responder staticUnified - KAS Role - initiator, responder	
KAS-FFC-SSC Sp800-56Ar3	A5370	Domain Parameter Generation Methods - FB, FC Scheme - dhEphem - KAS Role - initiator	SP 800-56A Rev. 3
KDA HKDF Sp800-56Cr1	A5370	Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-65336 Increment 8 HMAC Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/256	SP 800-56C Rev. 2
RSA KeyGen (FIPS186-5)	A5370	Key Generation Mode - probable Modulo - 2048, 3072, 4096 Primality Tests - 2powSecStr Private Key Format - standard	FIPS 186-5
RSA SigGen (FIPS186-5)	A5370	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
RSA SigVer (FIPS186-5)	A5370	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
SHA-1	A5370	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-224	A5370	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-256	A5370	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-384	A5370	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-512	A5370	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-512/256	A5370	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
TLS v1.2 KDF RFC7627 (CVL)	A5370	Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1
TLS v1.3 KDF (CVL)	A5370	HMAC Algorithm - SHA2-256, SHA2-384 KDF Running Modes - DHE, PSK, PSK-DHE	SP 800-135 Rev. 1

Table 5: Approved Algorithms

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
CKG	Key Type:Asymmetric	N/A	Section 4, example 1: U is directly output without XORing V

Table 6: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms with No Security Claimed:

N/A for this module.

Non-Approved, Not Allowed Algorithms:

Name	Use and Function
MD5, MD4	Non-Approved Hashing
POLYVAL	Non-Approved authenticated encryption
DES, Triple-DES (non-compliant)	Non-Approved encryption/decryption
AES (non-compliant)	Non-Approved encryption/decryption
DH (non-compliant)	Non-Approved key agreement
RSA PKCS #1 v1.5 key wrapping (non-compliant)	Non-Approved key wrapping
TLS 1.0/1.1 KDF (non-compliant)	Non-Approved TLS key derivation

Table 7: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
Authenticated Decryption	BC-Auth	Symmetric authenticated decryption of calling application data		AES-CCM: (A5370) AES-GCM: (A5370)
Authenticated Encryption	BC-Auth	Symmetric authenticated encryption of calling application data		AES-CCM: (A5370) AES-GCM: (A5370)
Decryption	BC-UnAuth	Symmetric decryption of calling application data		AES-CBC: (A5370) AES-CTR: (A5370) AES-ECB: (A5370)
Encryption	BC-UnAuth	Symmetric encryption of calling application data		AES-CBC: (A5370) AES-CTR: (A5370) AES-ECB: (A5370)

Name	Type	Description	Properties	Algorithms
Hashing	SHA	Hashing of calling application data		SHA-1: (A5370) SHA2-224: (A5370) SHA2-256: (A5370) SHA2-384: (A5370) SHA2-512: (A5370) SHA2-512/256: (A5370)
KAS-ECC-SSC	KAS-SSC	SP 800-56Arev3. KAS_ECC_SSC per IG D.F Scenario 2, path (1)	Caveat:providing 128, 192, or 256 bits of encryption strength	KAS-ECC-SSC Sp800-56Ar3: (A5370) CKG: () Key Type: Asymmetric Counter DRBG: (A5370) AES-ECB: (A5370)
KAS-FFC-SSC	KAS-SSC	SP 800-56Arev3. KAS_ECC_SSC per IG D.F Scenario 2, path (1)	Caveat:providing 112 bits of encryption strength	KAS-FFC-SSC Sp800-56Ar3: (A5370) CKG: () Key Type: Asymmetric Counter DRBG: (A5370) AES-ECB: (A5370)
Key Derivation Hash Based	KAS-56CKDF	Hash Based Key Derivation to support calling application		KDA HKDF Sp800-56Cr1: (A5370) SHA2-224: (A5370) SHA2-256: (A5370) SHA2-384: (A5370) SHA2-512: (A5370) SHA2-512/256: (A5370)
Key Derivation TLS 1.2	KAS-135KDF	Key derivation to support calling application's TLS implement		TLS v1.2 KDF RFC7627: (A5370) HMAC-SHA2-256: (A5370)

Name	Type	Description	Properties	Algorithms
				SHA2-256: (A5370) HMAC-SHA2-384: (A5370) SHA2-384: (A5370) HMAC-SHA2-512: (A5370) SHA2-512: (A5370)
Key Derivation TLS 1.3	KAS-135KDF	Key derivation to support calling application's TLS implement		TLS v1.3 KDF: (A5370) HMAC-SHA2-256: (A5370) SHA2-256: (A5370) HMAC-SHA2-384: (A5370) SHA2-384: (A5370)
AES-KeyWrap	BC-Auth	Symmetric key wrapping to support calling application's key transport	Caveat:providing 128, 192, or 256 bits of encryption strength	AES-KW: (A5370) AES-KWP: (A5370)
Message Authentication	MAC	Message authentication of calling application data		AES-GMAC: (A5370) HMAC-SHA-1: (A5370) Key Size: 112-bit or greater HMAC-SHA2-224: (A5370) Key Size: 112-bit or greater HMAC-SHA2-256: (A5370) Key Size: 112-bit or greater HMAC-SHA2-384: (A5370) Key Size: 112-bit or greater HMAC-SHA2-512: (A5370) Key Size: 112-bit or greater HMAC-SHA2-512/256:

Name	Type	Description	Properties	Algorithms
				(A5370) Key Size: 112-bit or greater SHA-1: (A5370) SHA2-224: (A5370) SHA2-256: (A5370) SHA2-384: (A5370) SHA2-512: (A5370) SHA2-512/256: (A5370)
Random Bit Generation	DRBG	Random Bit Generation		Counter DRBG: (A5370) AES-ECB: (A5370)
Signature Generation	DigSig-SigGen	Digital signature generation to support calling application		ECDSA SigGen (FIPS186-5): (A5370) RSA SigGen (FIPS186-5): (A5370) SHA2-224: (A5370) SHA2-256: (A5370) SHA2-384: (A5370) SHA2-512: (A5370) SHA2-512/256: (A5370)
Signature Key Generation	AsymKeyPair-KeyGen	Generation of digital signature key pairs		ECDSA KeyGen (FIPS186-5): (A5370) RSA KeyGen (FIPS186-5): (A5370) CKG: () Key Type: Asymmetric Counter DRBG: (A5370) AES-ECB: (A5370)

Name	Type	Description	Properties	Algorithms
Signature Key Validation	AsymKeyPair-KeyVer	Verification of ECDSA digital signature key pair		ECDSA KeyVer (FIPS186-5): (A5370)
Signature Verification	DigSig-SigVer	Verification of digital signature to support calling application		ECDSA SigVer (FIPS186-5): (A5370) RSA SigVer (FIPS186-5): (A5370) SHA2-224: (A5370) SHA2-256: (A5370) SHA2-384: (A5370) SHA2-512: (A5370) SHA2-512/256: (A5370)

Table 8: Security Function Implementations

2.7 Algorithm Specific Information

AES-CTR

Reuse of a counter value under the same AES key in AES-CTR is a serious cryptographic vulnerability. The developer integrating the module must prevent this vulnerability by never providing a counter start value that is the same or earlier than the last counter value used under that key for AES-CTR encryption.

AES-GCM

In the case of AES-GCM, the IV generation method is user-selectable, and the value can be computed in more than one manner.

The module does not implement the TLS protocol but offers cryptographic primitives that can be used by an external operator/application to implement TLS. The following restrictions must be followed when an external operator/application uses the module's AES-GCM within a TLS implementation.

In the context of the TLS protocol version 1.3, AES-GCM encryption and decryption is used compliant to Scenario 5 in FIPS 140-3 IG C.H. The module is compliant with NIST SP 800-52rev2 and the mechanism for IV generation is compliant with RFC 8446. 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 send a TLS 1.3 KeyUpdate message to establish a new encryption key, or fail. In either case, the module prevents any IV duplication and thus enforces the security property.

In the context of the TLS protocol version 1.2, AES-GCM encryption and decryption is used compliant to Scenario 1 in FIPS 140-3 IG C.H. The module is compatible with TLS protocol version 1.2 using AES-GCM ciphersuites as specified in NIST SP 800-52rev2, Section 3.3.1, 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, only internally generated IVs, or the TLS modes described above, are considered compliant for use.

Per 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.

AES-KW / AES-KWP

The module does not establish SSPs using an approved key transport scheme (KTS). However, it does offer approved authenticated algorithms that can be used by an external operator/application as part of an approved KTS.

Counter DRBG

The CTR_DRBG is used without a derivation function. IG D.L requires that a CTR_DRNG used without a derivation function shall be seeded from an entropy source producing full-entropy outputs, and the entropy source shall be located within the TOEPP. The module relies on passively provided entropy and it is the responsibility of the developer integrating the module to ensure that the entropy used to seed the DRBG comes from a source located within the TOEPP which CMVP has evaluated as producing full entropy output.

KAS-ECC-SSC and KAS-FFC-SSC

The module does not establish SSPs using an approved key agreement scheme (KAS). However, it does offer some or all of the underlying KAS cryptographic functionality to be used by an external operator/application as part of an approved KAS.

Hashing

The module does not perform truncation of hash outputs except as part of the approved SHA2-224, SHA2-512/256, and SHA2-384 algorithms. These algorithms inherently produce hash outputs of 224, 256, and 384 bits, respectively, as defined by FIPS 180-4. The module complies with IG C.L because no additional or manual truncation of hash outputs is implemented by the module in any other context.

Legacy Use Algorithms

SHA-1 is categorized as Legacy Use when using as part of digital signature verification. Algorithms designated as "Legacy" can only be used on data that was generated prior to the Legacy Date specified in FIPS 140-3 IG C.M. It is the responsibility of the developer integrating the module to ensure this restriction is met.

2.8 RBG and Entropy

N/A for this module.

N/A for this module.

The module passively receives entropy, per IG 9.3.A case 2(b), and shall be provided at least 384 bits of entropy. Use a SP 800-90B 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.

2.9 Key Generation

The module provides several key generation methods.

- Generation of asymmetric keys for key generation per SP 800-133r2 section 5.1 CKG and FIPS 186-5.
- Generation of asymmetric keys for key establishment per SP 800-133r2 section 5.2 CKG and SP 800-56Arev3.
- Derivation of symmetric keys for industry standard protocols from a key agreement shared secret per SP 800-133r2 section 6.2.1 and SP 800-135rev1 TLS v1.2 KDF / RFC 8446 TLS v1.3 KDF.
- Derivation of symmetric keys from a key agreement shared secret per SP 800-133r2 section 6.2.1 and SP 800-56Cr2 KDA HKDF.

The module does not have a key generation service for symmetric keys; however, a calling application may create one using output from the module’s Approved DRBG.

2.10 Key Establishment

The module provides the cryptographic building blocks for key agreement in its SP 800-56Arev3 KAS-ECC-SSC and KAS-FFC-SSC algorithms. A calling application may link these to the module’s SP 800-135rev1 TLS v1.2 KDF or RFC 8446 TLS v1.3 KDF to form a complete key agreement scheme.

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

2.11 Industry Protocols

The module does not implement any complete industry protocols, however it provides the key agreement and key derivation cryptographic algorithm building blocks to allow calling applications to implement the industry standard TLS v1.2 RFC 7627 or TLS v1.3 protocols using FIPS approved key establishment. The key establishment and generation primitives are described in sections 2.9 and 2.10, and the use of AES-GCM within them described in section 2.7.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
n/a	Data Input	API input parameters
n/a	Data Output	API output parameters and return values
n/a	Control Input	API input parameters
n/a	Status Output	API return values

Table 9: Ports and 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.

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.

The module does not implement a power interface or a control output interface.

4 Roles, Services, and Authentication

4.1 Authentication Methods

N/A for this module.

The module does not support operator authentication.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer (CO)	Role	Crypto Officer	None

Table 10: 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 concurrent operator is allowed.

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Module Initialization	Module Initialization	N/A	N/A	Return Code	None	Crypto Officer (CO)
Symmetric Encryption	Symmetric encryption of calling application data	fips_service_indicator set to 1	Plaintext, AAD, IV, encryption key	Return code, ciphertext, tag	Authenticated Encryption Encryption	Crypto Officer (CO) - AES Key: W,E - AES-GCM Key: W,E
Symmetric Decryption	Symmetric decryption of calling application data	fips_service_indicator set to 1	Ciphertext, AAD, IV, tag, decryption key	Return code, plaintext	Authenticated Decryption Decryption	Crypto Officer (CO) - AES Key: W,E - AES-GCM Key: W,E
Keyed Hashing	Symmetric message authentication of calling application data	fips_service_indicator set to 1	Message, key	Return code, Message Authentication Code	Message Authentication	Crypto Officer (CO) - HMAC Key: W,E - AES-GCM Key: W,E
Hashing	Hashing of calling application data	fips_service_indicator set to 1	Message	Return code, hash	Hashing	Crypto Officer (CO)
Random Bit Generation	Generation of random bits for calling application	fips_service_indicator set to 1	API call parameters	Return code, random bits	Random Bit Generation	Crypto Officer (CO) - CTR_DRBG Entropy Input: W,E - CTR_DRBG Seed: G,E - CTR_DRBG V: G,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- CTR_DRBG Key: G,E
Signature Generation	Digital signature of calling application data	fips_service_indicator set to 1	Message, signing key	Return code, signature	Signature Generation	Crypto Officer (CO) - ECDSA Signing Key: W,E - RSA Signature Generation Key: W,E - CTR_DRBG V: E - CTR_DRBG Key: E
Signature Verification	Digital signature verification of calling application data	fips_service_indicator set to 1	Signature, verification key	Return code	Signature Verification	Crypto Officer (CO) - ECDSA Verification Key: W,E - RSA Signature Verification Key: W,E
Key Wrap Service	Symmetric key wrapping of calling application key	fips_service_indicator set to 1	API call parameters, unwrapped key, wrapping key	Return code, wrapped key	AES-KeyWrap	Crypto Officer (CO) - AES Wrapping Key: W,E - Unwrapped Key: W - Wrapped Key: G,R

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Key Unwrap Service	Symmetric key unwrapping of calling application key	fips_service_indicator set to 1	API call parameters, wrapped key	Return code, unwrapped key	AES-KeyWrap	Crypto Officer (CO) - AES Wrapping Key: W,E - Wrapped Key: W - Unwrapped Key: G,R
Key Agreement Service	API call parameters	fips_service_indicator set to 1	Return code, shared secret	Return code, shared secret	KAS-ECC-SSC KAS-FFC-SSC	Crypto Officer (CO) - EC DH Private Key: G,E - EC DH Public Key: G,R - Other Party EC DH Public Key: W,E - DH Private Key: G,E - DH Public Key: W,E - Other Party DH Public Key: W,E - KAS Shared Secret: G,R
Key Derivation KDA	Hash based key derivation for calling application	fips_service_indicator set to 1	API call parameters, shared secret	Return code, derived keying material	Key Derivation Hash Based	Crypto Officer (CO) - KDA Shared Secret: W,E - Derived

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Keying Material: G,R
TLS Key Derivation	TLS Key derivation for calling application	fips_service_indicator set to 1	API call parameters, TLS KDF input	Return code, TLS keying material	Key Derivation TLS 1.2 Key Derivation TLS 1.3	Crypto Officer (CO) - TLS KDF Input: G,E - TLS Keying Material: G,R
Key Generation	Asymmetric key generation	fips_service_indicator set to 1	API call parameters	Return code, key pair	Signature Key Generation	Crypto Officer (CO) - ECDSA Signing Key: G,R - ECDSA Verification Key: G,R - RSA Signature Generation Key: G,R - RSA Signature Verification Key: G,R - CTR_DRBG V: E - CTR_DRBG Key: E
Key Verification	Asymmetric key pair validation	fips_service_indicator set to 1	API call parameters, key pair	Return code	Signature Key Validation	Crypto Officer (CO) - ECDSA Signing Key: W,E - ECDSA

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Verification Key: W,E
On-Demand Self-Test	On-Demand Self-Test	fips_service_indicator set to 1	N/A	Return Code	Authenticated Decryption Authenticated Encryption Decryption Encryption Hashing KAS-ECC-SSC KAS-FFC-SSC Key Derivation Hash Based Key Derivation TLS 1.2 Key Derivation TLS 1.3 AES-KeyWrap Message Authentication Random Bit Generation Signature Generation Signature Verification	Crypto Officer (CO)
Zeroisation	Zeroisation	fips_service_indicator set to 1	N/A	N/A	None	Crypto Officer (CO) - AES Key: Z - AES-GCM Key: Z - AES Wrapping Key: Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- Wrapped Key: Z - Unwrapped Key: Z - ECDSA Signing Key: Z - ECDSA Verification Key: Z - EC DH Private Key: Z - EC DH Public Key: Z - Other Party EC DH Public Key: Z - DH Private Key: Z - DH Public Key: Z - Other Party DH Public Key: Z - KAS Shared Secret: Z - KDA Shared Secret: Z - HMAC Key: Z - RSA Signature Generation Key: Z - RSA Signature Verification Key: Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- TLS KDF Input: Z - TLS Keying Material: Z - CTR_DRBG Entropy Input: Z - CTR_DRBG Seed: Z - CTR_DRBG V: Z - CTR_DRBG Key: Z - Derived Keying Material: Z
Show Status	Show Status	fips_service_indicator set to 1	API call parameters	Return code, status	None	Crypto Officer (CO)
Show Name	Show module name	fips_service_indicator set to 1	API call parameters	Return code, string of module name	None	Crypto Officer (CO)
Show Version	Show module version	fips_service_indicator set to 1	API call parameters	Return code, string of module version	None	Crypto Officer (CO)

Table 11: Approved Services

The Approved services supported by the module and access rights within services accessible over the module’s public interface are listed in the table above.

The corresponding FIPS_service_indicator_xxx function can be called to query the approved algorithm status of the proceeding service call, or the macro CALL_SERVICE_AND_CHECK_APPROVED can be used to automatically check the approved service indicator and inform the calling application whether the service called through the macro was FIPSStatus::APPROVED or FIPSStatus::NOT_APPROVED.

4.4 Non-Approved Services

Name	Description	Algorithms	Role
TLS 1.0/1.1 KDF	Perform hashing operations when used with the TLS protocol version 1.0 and 1.1	TLS 1.0/1.1 KDF (non-compliant)	Crypto Officer (CO)
Hashing	Perform hashing operations	MD5, MD4	Crypto Officer (CO)
Hashing for GCM-SIV	Used as part of AES-GCM-SIV	POLYVAL	Crypto Officer (CO)
Symmetric encryption/decryption	Perform symmetric encryption and/or decryption operations	DES, Triple-DES (non-compliant) AES (non-compliant)	Crypto Officer (CO)
Key Transport	Perform RSA PKCS #1 v1.5 key transport	RSA PKCS #1 v1.5 key wrapping (non-compliant)	Crypto Officer (CO)
Key Agreement	Perform non-compliant DH key agreement	DH (non-compliant)	Crypto Officer (CO)

Table 12: Non-Approved Services

Non-Approved Services are listed in the table above.

4.5 External Software/Firmware Loaded

The module does not support external software loading.

5 Software/Firmware Security

5.1 Integrity Techniques

The pre-operational integrity test is performed using HMAC-SHA-256.

5.2 Initiate on Demand

The integrity test can be executed on demand by power-cycling the host platform and reloading the module.

5.3 Open-Source Parameters

The module is open-source. To build the approved version of the module the following tools are required to build and compile the module.

Target Platform	Tools
Linux	- clang compiler version 17.0.6 (http://releases.llvm.org/download.html) - go programming language version 1.22.3 (https://golang.org/dl/) - ninja build system version 1.12.1 (https://github.com/ninja-build/ninja/releases) ● cmake version 3.29.3 (https://cmake.org/download/)

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

The module runs on a GPC, which is a modifiable operational environment, running one of the operating systems specified in Table 2. Each tested 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 special configuration of the operating system is required. The module is designed to ensure that the power-up 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 Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
RAM	Ephemeral storage in RAM	Dynamic

Table 13: Storage Areas

The module has no persistent SSP storage, all SSPs are stored ephemerally in RAM and are zeroised once no longer needed.

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
PT Input	Calling Application	RAM	Plaintext	N/A	Electronic	
PT Output	RAM	Calling Application	Plaintext	N/A	Electronic	

Table 14: SSP Input-Output Methods

The software module inputs and outputs SSP only as part of its API, and normally they are input or output in plaintext. The exceptions are the services that provide key wrapping or key unwrapping, where the wrapped key is entered or output encrypted.

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Power Cycle Host	Turn off or power cycle the host computer to clear all RAM contents	All module SSPs are held ephemerally in RAM, so turning off or power cycling the computer will cause them to be irretrievably lost.	Procedural - turn off or power cycle host

Table 15: SSP Zeroization Methods

The software module has no persistent SSP storage and the zeroisation method for all SSP is procedural. The operator may power-cycle the host computer to zeroise all SSPs the module currently holds.

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES Key	AES Key	128 192 256 - 128 192 256	Symmetric Key - CSP			Decryption Encryption
AES-GCM Key	AES-GCM Key	128 192 256 - 128 192 256	Symmetric Key - CSP			Authenticated Decryption Authenticated Encryption Message Authentication
AES Wrapping Key	AES Wrapping Key	128 192 256 - 128 192 256	Symmetric Key - CSP			AES-KeyWrap
Wrapped Key	Any calling application	Any - Any	Any - CSP			

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	key the module AES Key Wraps					
Unwrapped Key	Any calling application key the module AES Key Unwraps	Any - Any	Any - CSP			
ECDSA Signing Key	ECDSA Signing Key	P-224 P-256 P-384 P-521 - 112 128 192 256	Private - CSP	Signature Key Generation		Signature Generation Signature Key Validation
ECDSA Verification Key	ECDSA Verification Key	P-224 P-256 P-384 P-521 - 112 128 192 256	Public - PSP	Signature Key Generation		Signature Key Validation Signature Verification
EC DH Private Key	EC DH Private Key	P-256 P-384 P-521 - 128 192 256	Private - CSP	KAS-ECC-SSC		KAS-ECC-SSC
EC DH Public Key	EC DH Private Key	P-256 P-384 P-521 - 128 192 256	Public - PSP	KAS-ECC-SSC		KAS-ECC-SSC
Other Party EC DH Public Key	Other Party EC DH Public Key	P-256 P-384 P-521 - 128 192 256	Public - PSP			KAS-ECC-SSC
DH Private Key	DH Private Key	2048 - 112	Private - CSP	KAS-FFC-SSC		KAS-FFC-SSC
DH Public Key	DH Public Key	2048 - 112	Public - PSP	KAS-FFC-SSC		KAS-FFC-SSC
Other Party DH Public Key	Other Party DH Public Key	2048 - 112	Public - PSP			KAS-FFC-SSC
KAS Shared Secret	KAS Shared Secret	At least 112-bit -	Shared Secret - CSP		KAS-ECC-SSC	

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
		At least 112-bit			KAS-FFC-SSC	
KDA Shared Secret	KDA Shared Secret	At least 112-bit - At least 112-bit	Shared Secret - CSP			Key Derivation Hash Based
Derived Keying Material	Derived Keying Material	112 - 512 - 112 - 512	Symmetric Key - CSP	Key Derivation Hash Based		
HMAC Key	HMAC Key	At least 112-bit - At least 112-bit	Symmetric Key - CSP			Message Authentication
RSA Signature Generation Key	RSA Signature Generation Key	2048 3072 4096 - 112 128 150	Private - CSP	Signature Key Generation		Signature Generation
RSA Signature Verification Key	RSA Signature Verification Key	2048 3072 4096 - 112 128 150	Public - PSP	Signature Key Generation		Signature Verification
TLS KDF Input	TLS KDF input keying material	At least 112-bit - At least 112-bit	Shared Secret - CSP			Key Derivation TLS 1.2 Key Derivation TLS 1.3
TLS Keying Material	TLS Keying material from TLS KDF, for use by calling application TLS protocol	At least 112-bit - At least 112-bit	Symmetric Key - CSP	Key Derivation TLS 1.2 Key Derivation TLS 1.3		
CTR_DRBG Entropy Input	CTR_DRBG Entropy Input	384 - 384	Entropy Input - CSP			
CTR_DRBG Seed	CTR_DRBG Seed	384 - 384	DRBG CSP - CSP	Random Bit Generation		

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
CTR_DRBG V	CTR_DRBG V	128 - 128	DRBG CSP - CSP	Random Bit Generation		
CTR_DRBG Key	CTR_DRBG Key	256 - 256	DRBG CSP - CSP	Random Bit Generation		

Table 16: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES Key	PT Input	RAM:Plaintext	Until function completion	Power Cycle Host	
AES-GCM Key	PT Input	RAM:Plaintext	Until function completion	Power Cycle Host	
AES Wrapping Key	PT Input	RAM:Plaintext	Until function completion	Power Cycle Host	Wrapped Key:Encrypts Unwrapped Key:Decrypts
Wrapped Key	PT Input PT Output	RAM:Plaintext	Until function completion	Power Cycle Host	AES Wrapping Key:Wrapped By
Unwrapped Key	PT Input PT Output	RAM:Plaintext	Until function completion	Power Cycle Host	AES Wrapping Key:Unwrapped By
ECDSA Signing Key	PT Input PT Output	RAM:Plaintext	Until function completion	Power Cycle Host	ECDSA Verification Key:Paired With
ECDSA Verification Key	PT Input PT Output	RAM:Plaintext	Until function completion	Power Cycle Host	ECDSA Signing Key:Paired With
EC DH Private Key	PT Input PT Output	RAM:Plaintext	Until function completion	Power Cycle Host	EC DH Public Key:Paired With Other Party EC DH Public Key:Used With KAS Shared Secret:Establishes
EC DH Public Key	PT Input	RAM:Plaintext	Until function completion	Power Cycle Host	EC DH Private Key:Paired With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
	PT Output				
Other Party EC DH Public Key	PT Input	RAM:Plaintext	Until function completion	Power Cycle Host	EC DH Private Key:Used With KAS Shared Secret:Establishes
DH Private Key	PT Input PT Output	RAM:Plaintext	Until function completion	Power Cycle Host	DH Public Key:Paired With Other Party DH Public Key:Used With KAS Shared Secret:Establishes
DH Public Key	PT Input PT Output	RAM:Plaintext	Until function completion	Power Cycle Host	DH Private Key:Paired With
Other Party DH Public Key	PT Input	RAM:Plaintext	Until function completion	Power Cycle Host	DH Private Key:Used With KAS Shared Secret:Establishes
KAS Shared Secret	PT Input PT Output	RAM:Plaintext	Until function completion	Power Cycle Host	EC DH Private Key:Established By Other Party EC DH Public Key:Established By DH Private Key:Established By Other Party DH Public Key:Established By
KDA Shared Secret	PT Input	RAM:Plaintext	Until function completion	Power Cycle Host	Derived Keying Material:Derives
Derived Keying Material	PT Output	RAM:Plaintext	Until function completion	Power Cycle Host	KDA Shared Secret:Derived From
HMAC Key	PT Input	RAM:Plaintext	Until function completion	Power Cycle Host	
RSA Signature Generation Key	PT Input PT Output	RAM:Plaintext	Until function completion	Power Cycle Host	RSA Signature Verification Key:Paired With
RSA Signature Verification Key	PT Input PT Output	RAM:Plaintext	Until function completion	Power Cycle Host	RSA Signature Generation Key:Paired With
TLS KDF Input	PT Input	RAM:Plaintext	Until function completion	Power Cycle Host	TLS Keying Material:Derives

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
TLS Keying Material	PT Output	RAM:Plaintext	Until function completion	Power Cycle Host	TLS KDF Input:Derived From
CTR_DRBG Entropy Input	PT Input	RAM:Plaintext	Until function completion	Power Cycle Host	CTR_DRBG Seed:Derives
CTR_DRBG Seed		RAM:Plaintext	Until DRBG iunnstantiation	Power Cycle Host	CTR_DRBG Entropy Input:Derived From CTR_DRBG V:Derives CTR_DRBG Key:Derives
CTR_DRBG V		RAM:Plaintext	Until DRBG iunnstantiation	Power Cycle Host	CTR_DRBG Seed:Derived From
CTR_DRBG Key		RAM:Plaintext	Until DRBG iunnstantiation	Power Cycle Host	CTR_DRBG Seed:Derived From

Table 17: SSP Table 2

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A5370)	Hardcoded 512 bit key	Integrity	SW/FW Integrity	return code of 1 for success, 0 for failure	Single HMAC over entire module

Table 18: Pre-Operational Self-Tests

Pre-operational self-tests are run upon the initialization of the module. 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.

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC Encrypt KAT	128	KAT	CAST	None on success. "AES-CBC-encrypt KAT failed" on stderr on failure.	Encrypt	module power-up

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC Decrypt KAT	128	KAT	CAST	None on success. "AES-CBC-decrypt KAT failed" on stderr on failure.	Decrypt	module power-up
AES-GCM Encrypt KAT	128	KAT	CAST	None on success. "AES-GCM-encrypt KAT failed" on stderr on failure.	Encrypt	module power-up
AES-GCM Decrypt KAT	128	KAT	CAST	None on success. "AES-GCM-decrypt KAT failed" on stderr on failure.	Decrypt	module power-up
Counter DRBG KAT	initialize, reseed, generate tests, per SP 800-90Arev1 Section 11.3	KAT	CAST	None on success. "CTR-DRBG failed" on stderr on failure.	Instantiate, Reseed, Generate	module power-up
ECDSA SigGen KAT	P-256	KAT	CAST	None on success. "ECDSA-sign KAT failed" on stderr on failure.	Sign	On first use
ECDSA SigVer KAT	P-256	KAT	CAST	None on success. "ECDSA-verify KAT failed" on stderr on failure.	Verify	On first use
HMAC-SHA2-256 KAT	128	KAT	CAST	None on success. "HMAC-SHA-256 KAT failed" on stderr on failure.	MAC	module power-up
KAS-ECC-SSC Sp800-56Ar3 KAT	P-256	KAT	CAST	None on success. "Z-computation KAT failed." on stderr on failure.	SSC	On first use
KAS-FFC-SSC Sp800-56Ar3 KAT	2048	KAT	CAST	None on success. "FFDH failed" on stderr on failure.	SSC	On first use
KDA HKDF Sp800-	HMAC SHA-256	KAT	CAST	None on success. "HKDF failed" on stderr on failure.	KDF	module power-up

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
56Cr1 KAT						
RSA SigGen KAT	2048	KAT	CAST	None on success. "RSA-sign KAT failed" on stderr on failure.	Sign	On first use
RSA SigVer KAT	2048	KAT	CAST	None on success. "RSA-verify KAT failed" on stderr on failure.	Verify	On first use
SHA-1 KAT	n/a	KAT	CAST	None on success. "SHA-1 KAT failed" on stderr on failure.	Hash	module power-up
SHA2-256 KAT	n/a	KAT	CAST	None on success. "SHA-256 KAT failed" on stderr on failure.	Hash	module power-up
SHA2-512 KAT	n/a	KAT	CAST	None on success. "SHA-512 KAT failed" on stderr on failure.	Hash	module power-up
TLS v1.2 KDF RFC7627 KAT	SHA-256	KAT	CAST	None on success. "TLS12-KDF KAT failed" on stderr on failure.	KDF	module power-up
TLS v1.3 KDF KAT	SHA-256	KAT	CAST	None on success. "TLS13-KDF KAT failed" on stderr on failure.	KDF	module power-up
ECDSA KeyGen PCT	Generated key-pair	Sign/Verify PCT	PCT	None on success. "EC_KEY_generate_key_fips failed" on stderr on failure.	Sign/Verify PCT	Keypair generated
RSA KeyGen PCT	Generated key-pair	Sign/Verify PCT	PCT	None on success. "RSA_generate_key_fips failed" on stderr on failure.	Sign/Verify PCT	Keypair generated
KAS-ECC-SSC Sp800-56Ar3 PCT	Generated key-pair	SP 800-56Arev3 validity tests	PCT	None on success. Module aborted on failure.	SP 800-56Arev3 validity tests	Keypair generated
KAS-FFC-SSC Sp800-56Ar3 PCT	Generated key-pair	SP 800-56Arev3 validity tests	PCT	None on success. Module aborted on failure.	SP 800-56Arev3 validity tests	Keypair generated

Table 19: 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.

Pair-wise consistency tests (PCT) are run during the module's operation when a new asymmetric keypair is generated. 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.

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A5370)	Integrity	SW/FW Integrity	On startup	Manually call On-Demand Self-Test service or restart module

Table 20: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CBC Encrypt KAT	KAT	CAST	Operator chosen	Manually call On-Demand Self-Test service or restart module
AES-CBC Decrypt KAT	KAT	CAST	Operator chosen	Manually call On-Demand Self-Test service or restart module
AES-GCM Encrypt KAT	KAT	CAST	Operator chosen	Manually call On-Demand Self-Test service or restart module
AES-GCM Decrypt KAT	KAT	CAST	Operator chosen	Manually call On-Demand Self-Test service or restart module
Counter DRBG KAT	KAT	CAST	Operator chosen	Manually call On-Demand Self-Test service or restart module

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
ECDSA SigGen KAT	KAT	CAST	Operator chosen	Manually call On-Demand Self-Test service or restart module
ECDSA SigVer KAT	KAT	CAST	Operator chosen	Manually call On-Demand Self-Test service or restart module
HMAC-SHA2-256 KAT	KAT	CAST	Operator chosen	Manually call On-Demand Self-Test service or restart module
KAS-ECC-SSC Sp800-56Ar3 KAT	KAT	CAST	Operator chosen	Manually call On-Demand Self-Test service or restart module
KAS-FFC-SSC Sp800-56Ar3 KAT	KAT	CAST	Operator chosen	Manually call On-Demand Self-Test service or restart module
KDA HKDF Sp800-56Cr1 KAT	KAT	CAST	Operator chosen	Manually call On-Demand Self-Test service or restart module
RSA SigGen KAT	KAT	CAST	Operator chosen	Manually call On-Demand Self-Test service or restart module
RSA SigVer KAT	KAT	CAST	Operator chosen	Manually call On-Demand Self-Test service or restart module
SHA-1 KAT	KAT	CAST	Operator chosen	Manually call On-Demand Self-Test service or restart module
SHA2-256 KAT	KAT	CAST	Operator chosen	Manually call On-Demand Self-Test service

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
				or restart module
SHA2-512 KAT	KAT	CAST	Operator chosen	Manually call On-Demand Self-Test service or restart module
TLS v1.2 KDF RFC7627 KAT	KAT	CAST	Operator chosen	Manually call On-Demand Self-Test service or restart module
TLS v1.3 KDF KAT	KAT	CAST	Operator chosen	Manually call On-Demand Self-Test service or restart module
ECDSA KeyGen PCT	Sign/Verify PCT	PCT	n/a	n/a
RSA KeyGen PCT	Sign/Verify PCT	PCT	n/a	n/a
KAS-ECC-SSC Sp800-56Ar3 PCT	SP 800-56Arev3 validity tests	PCT	n/a	n/a
KAS-FFC-SSC Sp800-56Ar3 PCT	SP 800-56Arev3 validity tests	PCT	n/a	n/a

Table 21: Conditional Periodic Information

A level 1 module does not require automatic periodic self-testing, however self-tests can be rerun by On-Demand Self-Test service (BORINGSSL_self_test) or restarting the module.

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Error state	The module's error state	Failure of any FIPS self-test	Restart the module	Possible indication of failed test on stderr. Module aborts.

Table 22: Error States

The module's single error state is shows above.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

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.

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-20230428/crypto/fipsmodule/FIPS.md (note this still mentions 140-2, but the information there is the same).

The module is open source and must be built on a Linux workstation using the tools in section 5.3 *Open Source Parameters*.

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-85897d07196b7bf164dbd4673fc78b762aff3e8b.tar.xz or by issuing the following command:

```
wget https://commondatastorage.googleapis.com/chromium-boringssl-fips/boringssl-
85897d07196b7bf164dbd4673fc78b762aff3e8b.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 SHA-256 digest value:

b1c87a2746e831dd51448038d8ec7d0ba256d949e73dace0c9a1484889d82d1a

By issuing the following command:

```
sha256sum boringssl-85897d07196b7bf164dbd4673fc78b762aff3e8b.tar.xz
```

The tarball can be extracted using the following command:

```
tar xJ < boringssl-85897d07196b7bf164dbd4673fc78b762aff3e8b.tar.xz
```

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

```
cd boringssl
mkdir build && cd build
ninja bcm.o
```

Retrieving Module name and version

The following methods will provide the module name and versions:

- FIPS_module_name() – BoringCrypto
- FIPS_version() – 20240407

11.2 Administrator Guidance

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.

11.3 Non-Administrator Guidance

The module does not support a non-administrator role.

11.4 Additional Information

The source code for the module is maintained in a git repository. While in development, work on the code is maintained internally, before eventually being released externally. BoringCrypto is released publicly to https://boringssl.googlesource.com/boringssl (this is the generic version, available under the Building for Linux instructions). The version number is determined by the developer releasing the version, though git attaches hashes to every single file and branch in the repository.

12 Mitigation of Other Attacks

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