FIPS 140-3 Non-Proprietary Security Policy

Google, LLC.

BoringCrypto

Software Version: 2023042800

Date: January 10, 2025

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 program. The NVLAP accredits independent testing labs to perform FIPS 140 testing; the CMVP validates modules meeting FIPS 140 validation. Validated is the term given to a module that is documented and tested against the FIPS 140 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 BoringCrypto from Google, 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 BoringCrypto module 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. Google, LLC. shall have no liability for any error or damages of any kind resulting from the use of this document.

Notices

This document may be freely reproduced and distributed in its entirety without modification.

Table of Contents

List of Tables

List of Figures

1. General

This document describes the cryptographic module Security Policy (SP) for the Google, LLC. BoringCrypto (Software version: 2023042800) 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 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:

Section	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 Level

2. Cryptographic Module Specification

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.

The boundary of the module is defined as a single object file, bcm.o, and its instantiation in memory. The module version is: 2023042800.

The module was tested on the following operational environments:

#	Operating System	Hardware Platform	Processor	PAA/Acceleration
1	Android 14	Google Pixel 8	Google Tensor G3 64-bit	With and without PAA
2	Android 14	Google Pixel 7	Google Tensor G2 64-bit and 32-bit	With and without PAA
3	Android 14	Google Pixel 6	Google Tensor 64-bit and 32-bit	With and without PAA
4	Android 14	Google Pixel 5a	Qualcomm Snapdragon 765 64-bit and 32-bit	With and without PAA
5	Google Prodimage with Linux 5.15.110	IN762	IN762	With and without PAA
6	Google Prodimage with Linux 5.10.0	Tau t2a	Ampere Altra	With and without PAA
7	Ubuntu 23.04	Gigabyte GA-Z170X-UD5	Intel Core i7-6700K	With and without PAA
8	Debian Linux 6.4.4	n2d	AMD EPYC 7B12	With and 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:

#	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

CAVP Cert	Algorithm and Standard	Mode/Method	Description / Key Size(s) / Key Strength(s)	Use / Function
A4687	AES FIPS 197 SP 800-38A	CBC, ECB, CTR	128, 192, 256-bit keys with 128. 192, 256-bit key strength	Symmetric Encryption, Symmetric Decryption
A4687	AES FIPS 197 SP 800-38D	GCM	128, 192, 256-bit keys with 128. 192, 256-bit key strength	Symmetric Authenticated Encryption, Symmetric Authenticated Decryption
A4687	AES FIPS 197 SP 800-38D	GMAC	128, 192, 256-bit keys with 128. 192, 256-bit key strength	Symmetric message authentication
A4687	AES FIPS 197 SP 800-38C	CCM	128-bit keys with 128-bit key strength	Symmetric Authenticated Encryption, Symmetric Authenticated Decryption
A4687	AES FIPS 197 SP 800-38F	KW, KWP	128, 192, 256-bit keys with 128. 192, 256-bit key strength	Symmetric Key Wrapping, Symmetric Key Unwrapping
A4687	DRBG SP 800-90Arev1	CTR_DRBG	AES-256	Random Bit Generation
A4687	ECDSA FIPS 186-4	Key Pair Generation, Signature Generation, Signature Verification, Public Key Verification	P-224, P-256, P-384, P-521 with 112, 128, 192, 256-bit key strength	Asymmetric Digital Signature Services
A4687	HMAC FIPS 198-1	Generate, Verify HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 HMAC-SHA-512/256	Key Length: 8-524288 Increment 8[1]	Symmetric Generation, Symmetric Authentication
A4687	RSA FIPS 186-4	Key Generation, Signature Generation, Signature Verification	(1024, 2048, 3072, 4096)	Asymmetric Digital Signature Services

[1] HMAC key lengths < 112 bits are disallowed by SP 800-131Ar2.

CAVP Cert	Algorithm and Standard	Mode/Method	Description / Key Size(s) / Key Strength(s)	Use / Function
		PKCS 1.5 and PSS	Note: Key size 1024 is only used for Signature Verification	
A4687	SHS FIPS 180-4	Hashing	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/256	Digital Signature Generation, Digital Signature Verification, non-Digital Signature Applications
A4687	KAS-ECC-SSC SP 800-56Arev3	KAS-ECC-SSC ephemeralUnified, staticUnified	P-224, P-256, P-384 and P-521 with 112, 128, 192, 256-bit key strength	Asymmetric Key Agreement Scheme Shared Secret Computation per SP 800-56Arev3
A4687	KAS-FFC-SSC SP 800-56Arev3	KAS-FFC-SSC dhEphem	2048/244, 2048/256-bit keys with 112-bit key strength	Asymmetric Key Agreement Scheme Shared Secret Computation per SP 800-56Arev3
A4687	KDA HKDF Sp800-56Cr1	KDA HKDF	HMAC SHA2-224, HMAC SHA2-256, HMAC SHA2-384, HMAC SHA2-512, HMAC SHA2-512/256 112-512-bit security strength	Hash Based Key Derivation
A4687	TLS v1.2 KDF RFC7627	CVL TLS v1.2 KDF RFC7627	SHA2-256, SHA2-384, SHA2-512	TLS Key Derivation
A4687	TLS v1.3 KDF	CVL TLS v1.3 KDF DHE, PSK, PKS-DHE	SHA2-256, SHA2-384	TLS Key Derivation

Table 4 – Approved Algorithms

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

Algorithm	Caveat	Use / Function
CKG [IG D.H]		Cryptographic key generation per SP 800-133rev2 and IG D.I * Generation of asymmetric keys for signature generation per [133] section 5.1. * Generation of asymmetric keys for key establishment per [133] section 5.2. * Symmetric key derivation for industry

		standard protocols from a key agreement shared secret per [133] section 6.2.1.

Table 5 – Vendor Affirmed Algorithms

The module does not implement any Non-Approved Algorithms Allowed in the Approved Mode of Operation. (SP 800-140B tables 'Non-Approved Algorithms Allowed in the Approved Mode of Operation' has been omitted)

The module does not implement any Non-Approved Algorithms Allowed in the Approved Mode of Operation with No Security Claimed. (SP 800-140B tables 'Non-Approved Algorithms Allowed in the Approved Mode of Operation with No Security Claimed' has been omitted)

Algorithm/Function	Use/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 6 – Non-Approved Algorithms Not Allowed in the Approved Mode of Operation

Name	Type	Description	SP Properties	Algorithms/CAVP Cert
KAS-ECC-SSC	KAS	SP 800-56Arev3. KAS_ECC_SSC per IG D.F Scenario 2, path (1)	P-224, P-256, P-384, P- 521 curves providing 128, 192, or 256 bits of encryption strength	KAS-ECC-SSC Sp800-56Ar3/A4687
KAS-ECC	KAS	SP 800-56Arev3. KAS_ECC_SSC per IG D.F Scenario 2, path (2). No key confirmation, key derivation per IG 2.4.B. SP 800-135. KDFs (TLS v1.2 RFC6727, TLS v1.3)	P-224, P-256, P-384, P- 521 curves providing 128, 192, or 256 bits of encryption strength	KAS-ECC-SSC Sp800-56Ar3/A4687 TLS v1.2 KDF RFC7627/A4687 TLS v1.3 KDF/A4687
KAS-FFC-SSC	KAS	SP 800-56Arev3. KAS_ECC_SSC per IG D.F Scenario 2, path (1)	2048/244, 2048/256-bit keys with 112-bit of encryption strength	KAS-FFC-SSC Sp800-56Ar3/A4687
KAS-FFC	KAS	SP 800-56Arev3. KAS_FFC_SSC per IG D.F Scenario 2, path (2). No key confirmation, key derivation per IG 2.4.B. SP 800-135. KDFs (TLS v1.2 RFC6727, TLS v1.3)	2048/244, 2048/256-bit keys with 112-bit of encryption strength	KAS-FFC-SSC Sp800-56Ar3/A4687 TLS v1.2 KDF RFC7627/A4687 TLS v1.3 KDF/A4687

KTS	KTS	SP 800-38F. AES-KW, AES-KWP	128, 192, 256-bit keys with 128, 192, or 256-bit encryption strength	AES-KW/A4687 AES-KWP/A4687

Table 7 – Security Function Implementation (SFI)

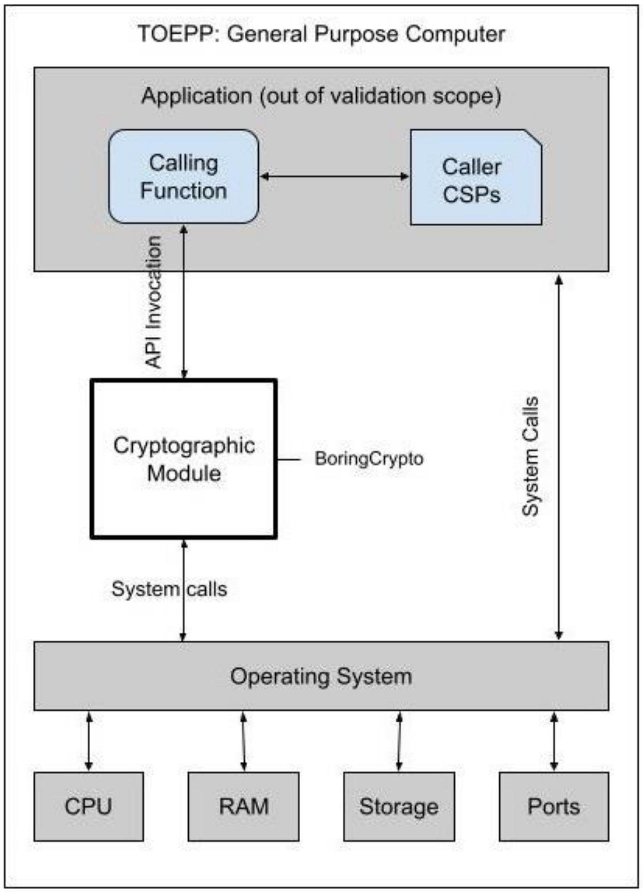

Figure 1 – BoringCrypto cryptographic boundary

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

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 8 – Ports and Interfaces

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 concurrent operator is allowed.

4.2 Authentication

The module does not support operator authentication.

Role	Authentication Method	Authentication Strength
Crypto Officer (CO)	n/a	n/a

Table 9 – Roles and 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 the table below:

Role	Service	Input	Output
CO	Module Initialization	N/A	Return code
CO	Symmetric Encryption	Plaintext, AAD, IV encryption key	Return code, ciphertext, tag
CO	Symmetric Decryption	Ciphertext, AAD, IV, tag, 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 Wrap	API call parameters, unwrapped key, wrapping key	Return code, wrapped key
CO	Key Unwrap	API call parameters, wrapped key	Return code, unwrapped key
CO	Key Agreement	API call parameters	Return code, shared secret
CO	Key Derivation KDA	API call parameters, shared secret	Return code, derived key
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 10 – Roles, Service Commands, Input and Output

Approved services are listed in Table 11 – Approved Services. 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.

The indicator is provided as part of the API and is not a separate call. When a call is made to the module to perform an action using an algorithm, the response contains the indicator as part of the return value automatically. There is not a separate call just to determine the indicator status other than the general status parameter as to whether the entire module is in the approved state or not.

Service	Description	Approved Security Functions	Keys/SSP's	Role	Access rights to Keys/SSP's	Indicator
Module Initialization	Initializes the module	N/A	N/A	CO	N/A	N/A
Symmetric Encryption	Perform symmetric encryption operations	AES CBC, ECB, CTR, GCM, CCM	AES Key, AES-GCM Key	CO	W, E	1
Symmetric Decryption	Perform symmetric decryption operations	AES CBC, ECB, CTR, GCM, CCM	AES Key, AES-GCM Key	CO	W, E	1
Keyed Hashing	Perform keyed hashing operations	HMAC, GMAC	HMAC Key, AES-GCM Key	CO	W, E	1
Hashing	Perform hashing operations	SHS	N/A	CO	N/A	1
Random Bit Generation	Generate random numbers	CTR_DRBG	DRBG Seed, CTR_DRBG V, CTR_DRBG Key	CO	G, E	1
			CTR_DRBG Entropy Input		W, E	
Signature Generation	Perform signing operations	CTR_DRBG, RSA, ECDSA	RSA Signature Generation Key, ECDSA Signing Key	CO	W, E	1
			CTR_DRBG V, CTR_DRBG Key		E	
Signature Verification	Perform verification operations	RSA, ECDSA	RSA Signature Verification Key, ECDSA Verification Key	CO	W, E	1
Key Wrap	Perform key encryption operations	AES KW, KWP	AES Wrapping Key	CO	W, E	1
			Unwrapped Key		W	
			Wrapped Key		G, R	
Key Unwrap	Perform key decryption operations	AES KW, KWP	AES Wrapping Key	CO	W, E	1
			Wrapped Key		W	
			Unwrapped Key		G, R	
Key Agreement	Perform key agreement operations	KAS-ECC-SSC, KAS-FFC-SSC	EC DH Private Key & EC DH Public Key, DH Private Key & DH Public Key	CO	G, E	1

Service	Description	Approved Security Functions	Keys/SSP's	Role	Access rights to Keys/SSP's	Indicator
			Other Party EC DH Public Key, Other Party DH Public Key		W, E	
			Shared Secret		G, R	
Key Derivation KDA	Perform key derivation operations	KDA HKDF Sp800-56Cr2	Shared Secret	CO	W, E	1
			Derived Key		G, R	
TLS Key Derivation	Perform key derivation operations	TLS KDF	TLS Pre-Master Secret	CO	W, E	1
			TLS Master Secret		G, E	
			TLS Key		G, R	
Key Generation	Perform generation operations	CTR_DRBG, RSA, ECDSA	CTR_DRBG V, CTR_DRBG Key	CO	E	1
			RSA Signature Generation Key & RSA Signature Verification Key, ECDSA Signing Key & ECDSA Verification Key		G, E, R	
Key Verification	Perform key pair verification operations	ECDSA	ECDSA Signing Key, ECDSA Verification Key	CO	W, E	1
On-Demand Self-Test	Execute self-tests on demand	N/A	N/A	CO	N/A	1
Zeroization	Zeroize all SSPs	N/A	All keys	CO	Z	N/A
Show Status	Obtain the module status and versioning information	N/A	N/A	CO	N/A	N/A

Table 11 – Approved Services

Non-Approved Services are listed in the Table 12 below:

Service	Description	Algorithms Accessed	Role	Indicator
TLS 1.0/1.1 KDF	Perform hashing operations when used with the TLS protocol version 1.0 and 1.1	MD5 & SHA-1	CO	0
Hashing	Perform hashing operations	MD4, MD5	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 Transport	Perform RSA PKCS #1 v1.5 key transport	RSA	CO	0

Table 12 - Non-Approved Services

5. Software/Firmware Security

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

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 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 Parameter Management

Key/SSP Name/Type	Strength	Security Function Cert Number	Generation	Import/ Export	Establishment	Storage	Zeroisation	Use & related keys
AES Key	128 192 256	A4687	External	Input via API in plaintext (Electronic Entry)	N/A	Plaintext in RAM until function completion	Power-cycle host	AES encrypt / decrypt
AES-GCM Key	128 192 256	A4687	External	Input via API in plaintext (Electronic Entry)	N/A	Plaintext in RAM until function completion	Power-cycle host	AES encrypt / decrypt / generate / verify
AES Wrapping Key	128 192 256	A4687	External	Input via API in plaintext (Electronic Entry)	N/A	Plaintext in RAM until function completion	Power-cycle host	AES key wrapping; wraps Unwrapped Key; unwraps Wrapped Key

Key/SSP Name/Type	Strength	Security Function Cert Number	Generation	Import/ Export	Establishment	Storage	Zeroisation	Use & related keys
Wrapped Key	Any	N/A	External	Input via API wrapped (Electronic Entry) / Output via API wrapped (Electronic Output)	N/A[2]	Wrapped in RAM until function completion	Power-cycle host	Key Transport; Unwrapped by AES Wrapping Key; becoming Unwrapped Key
Unwrapped Key	Any	N/A	External	Input via API in plaintext (Electronic Entry) / Output via API in plaintext (Electronic Output)	N/A	Plaintext in RAM until function completion	Power-cycle host	Key Transport; Wrapped by AES Wrapping Key; becoming Wrapped Key

[2] Module only wraps or unwraps the key, transporting the key would be performed by the calling application.

Key/SSP Name/Type	Strength	Security Function Cert Number	Generation	Import/ Export	Establishment	Storage	Zeroisation	Use & related keys
ECDSA Signing Key	112 128 192 256	A4687	Internally per FIPS 186-4	Input via API in plaintext (Electronic Entry) / Output via API in plaintext (Electronic Output)	N/A	Plaintext in RAM until function completion	Power-cycle host	ECDSA signature generation; Paired with ECDSA Verification Key
ECDSA Verification Key	112 128 192 256	A4687	Internally per FIPS 186-4	Input via API in plaintext (Electronic Entry) / Output via API in plaintext (Electronic Output)	N/A	Plaintext in RAM until function completion	Power-cycle host	ECDSA signature verification; Paired with ECDSA Signing Key

Key/SSP Name/Type	Strength	Security Function Cert Number	Generation	Import/ Export	Establishment	Storage	Zeroisation	Use & related keys
EC DH Private Key	112 128 192 256	A4687	Internally per SP 800-56Arev3	Input via API in plaintext (Electronic Entry) / Output via API in plaintext (Electronic Output)	N/A	Plaintext in RAM until function completion	Power-cycle host	EC DH; Paired with EC DH Public Key; Used with Other Party EC DH Public Key; Establishes Shared Secret, TLS Pre-Master Secret
EC DH Public Key	112 128 192 256	A4687	Internally per SP 800-56Arev3	Input via API in plaintext (Electronic Entry) / Output via API in plaintext (Electronic Output)	N/A	Plaintext in RAM until function completion	Power-cycle host	EC DH; Paired with EC DH Private Key; Establishes Shared Secret, TLS Pre-Master Secret

Key/SSP Name/Type	Strength	Security Function Cert Number	Generation	Import/ Export	Establishment	Storage	Zeroisation	Use & related keys
Other Party EC DH Public Key	112 128 192 256	A4687	External	Input via API in plaintext (Electronic Entry)	N/A	Plaintext in RAM until function completion	Power-cycle host	EC DH; Used with EC DH Private Key; Establishes Shared Secret, TLS Pre-Master Secret
DH Private Key	112	A4687	Internally per SP 800-56Arev3	Input via API in plaintext (Electronic Entry) / Output via API in plaintext (Electronic Output)	N/A	Plaintext in RAM until function completion	Power-cycle host	DH; Paired with DH Public Key; Used with Other Party DH Public Key; Establishes Shared Secret, TLS Pre-Master Secret
DH Public Key	112	A4687	Internally per SP 800-56Arev3	Input via API in plaintext (Electronic Entry) / Output via API in plaintext (Electronic Output)	N/A	Plaintext in RAM until function completion	Power-cycle host	DH; Paired with DH Private Key; Establishes Shared Secret, TLS Pre-Master Secret

Key/SSP Name/Type	Strength	Security Function Cert Number	Generation	Import/ Export	Establishment	Storage	Zeroisation	Use & related keys
Other Party DH Public Key	112	A4687	External	Input via API in plaintext (Electronic Entry)	N/A	Plaintext in RAM until function completion	Power-cycle host	DH; Used with DH Private Key; Establishes Shared Secret, TLS Pre-Master Secret
Shared Secret	At least 112-bit	N/A	External	Input via API in plaintext (Electronic Entry) / Output via API in plaintext (Electronic Output)	KAS-ECC-SSC, KAS-FFC-SSC	Plaintext in RAM until function completion	Power-cycle host	EC DH or DH; Established by EC DH Private Key, EC DH Public Key, Other Party EC DH Public Key, DH Private Key, DH Public Key, Other Party DH Public Key, Derives Derived Key
HMAC Key	At least 112-bit	A4687	External	Input via API in plaintext (Electronic Entry)	N/A	Plaintext in RAM until function completion	Power-cycle host	Keyed hashing

Key/SSP Name/Type	Strength	Security Function Cert Number	Generation	Import/ Export	Establishment	Storage	Zeroisation	Use & related keys
RSA Signature Generation Key	112 128 150	A4687	Internally per FIPS 186-4	Input via API in plaintext (Electronic Entry) / Output via API in plaintext (Electronic Output)	N/A	Plaintext in RAM until function completion	Power-cycle host	RSA signature generation; Paired with RSA Signature Verification Key
RSA Signature Verification Key	80 112 128 150	A4687	Internally per FIPS 186-4	Input via API in plaintext (Electronic Entry) / Output via API in plaintext (Electronic Output)	N/A	Plaintext in RAM until function completion	Power-cycle host	RSA signature verification; Paired with RSA Signature Generation Key
Derived Key	112 – 512	A4687	Internally per SP 800-56Cr2	Output via API in plaintext (Electronic Output)	N/A	Plaintext in RAM until function completion	Power-cycle host	Key Derivation; Derived from Shared Secret
TLS Pre-Master Secret (other SSP)	At least 112-bit	A4687	External	Input via API in plaintext (Electronic Entry)	KAS-ECC, KAS-FFC	Plaintext in RAM until function completion	Power-cycle host	TLS key derivation; Derives TLS Master Secret

Key/SSP Name/Type	Strength	Security Function Cert Number	Generation	Import/ Export	Establishment	Storage	Zeroisation	Use & related keys
TLS Master Secret (other SSP)	At least 112-bit	A4687	Internally Derived via SP 800-135 KDF (TLS)	N/A	N/A	Plaintext in RAM until function completion	Power-cycle host	TLS key derivation; Derived from TLS Pre-Master Secret, Derives TLS Key
TLS Key	At least 112-bit	A4687	Internally Derived via SP 800-135 KDF (TLS)	Output via API in plaintext (Electronic Output)	N/A	Plaintext in RAM until function completion	Power-cycle host	TLS; Derived from TLS Master Secret
DRBG Seed	384 bits	A4687	Internally per SP 800-90Ar1	N/A	N/A	Plaintext in RAM until DRBG uninstantiated or module shutdown	Power-cycle host	DRBG Seeding material
CTR_DRBG V	128 bits	A4687	Internally per SP 800-90Ar1	N/A	N/A	Plaintext in RAM until DRBG uninstantiated or module shutdown	Power-cycle host	DRBG internal state

Key/SSP Name/Type	Strength	Security Function Cert Number	Generation	Import/ Export	Establishment	Storage	Zeroisation	Use & related keys
CTR_DRBG Key	256 bits	A4687	Internally per SP 800-90Ar1	N/A	N/A	Plaintext in RAM until DRBG uninstantiated or module shutdown	Power-cycle host	DRBG internal state
CTR_DRBG Entropy Input	384 bits	A4687	External	Input via API in plaintext (Electronic Entry)	N/A	Plaintext in RAM until function completion	Power-cycle host	DRBG entropy

Table 13 – SSP's

Entropy sources	Minimum number of bits of entropy	Details
Passive Entropy	Shall provide module at least 384 bits of entropy	Use of 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.

Table 14 – Non-Deterministic Random Number Generation Specification

10. Self-tests

FIPS 140-3 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. Self-tests can be requested on demand by power cycling the host platform. The module has a single error state, just called the error state. The failure of a self-test will cause the module to enter the error state. The module indicates this error state by providing the output status “Aborted”. 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. 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:

Type	Test
Software Integrity Test	HMAC-SHA-256

Table 15 – Pre-operational Self-tests

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:

Type	Test
KAT	ECDSA Signature Generation (P-256)
	ECSDA Signature Verification (P-256)
	RSA Signature Generation (2048 bits)
	RSA Signature Verification (2048 bits)
	SP800-56Arev3 KAS-ECC-SSC (P-256)
	SP800-56Arev3 KAS-FFC-SSC (2048 bits)
	AES CBC Encryption (128 bits)
	AES CBC Decryption (128 bits)
	AES-GCM Encryption (128 bits)
	AES-GCM Decryption (128 bits)
	TLS v1.2 KDF
	TLS v1.3 KDF
	HKDF
	SHA-1
	SHA2-256
	SHA2-512

Type	Test
	HMAC-SHA2-256
CAST	performed on DRBG, per SP800-90Arev1 Section 11.3

Table 16 – Conditional Algorithm Self-tests

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 addition conditional self-tests:

Type	Test
Pair-wise Consistency Test	ECDSA Key Pair generation
	SP800-56Arev3 EC DH Key Pair generation
	SP800-56Arev3 DH Key Pair generation
	RSA Key Pair generation

Table 17 – Conditional Self-tests

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.

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).

11.1 Configuration Management

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.

The Android version of the module is also maintained in a git repository. Once the generic version is available, it is imported into the Android repository. As with the generic version, while development on the port is performed, it is handled in an internal git repository. Once it is ready for release it is published publicly to https://ci.android.com. The version number is determined by the Android repository build number (the numeric part of the manifest filename).

Only the Android version of the module is released as a pre-compiled version (as opposed to a self-compiled version. The Android manifest specifies all the configuration information needed to duplicate the build.

Documentation that isn't included in text files stored in git is maintained in Google Docs. All documents (whether spreadsheets, documents, presentations or anything else) are automatically version tracked along with the owner. Like git, Docs uses access control lists to control access to the design documentation for the module.

All internal systems (both git and Google Docs) utilize the Google ID for login and access control over the repositories.

11.2 Installation Instructions

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

Target Platform	Tools
Android	● repo git repository tool 2.4.0 (https://gerrit.googlesource.com/git-repo)
Linux	● clang compiler version 16.0.0 (http://releases.llvm.org/download.html) ● go programming language version 1.21.1 (https://golang.org/dl/)

	● ninja build system version 1.11.1 (https://github.com/ninja-build/ninja/releases) ● cmake version 3.27.4 (https://cmake.org/download/)

Table 18 – Build Tools

11.2.1 Building for Android

The necessary Android build tools that are configured as part of the manifest. Running the envsetup.sh script will ensure that the proper environment is set to build the library for Android.

Download the manifest from https://ci.android.com/builds/submitted/10050109/aosp_cf_arm64_phone-userdebug/latest by clicking the Download button.

Verify the manifest using the following command:

```
sha256sum ~/manifest_10050109.xml
```

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

5f8701016e3c39503e26c81e0facb6ab386319b94f5d2508288907e652060d92 manifest_10050109.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_10050109.xml
~/repo sync -q -c -j 50
```

To build the correct test tools (the test_fips components below, not the module), the following additional steps need to be followed:

```
cd external/boringssl
git fetch https://android.googlesource.com/platform/external/boringssl refs/changes/99/2775199/2 && git
cherry-pick FETCH_HEAD
git fetch https://android.googlesource.com/platform/external/boringssl refs/changes/28/2778328/2 && git
cherry-pick FETCH_HEAD
cd src/util/fipstools
nano break-kat.go

Change the first line of the file to be:
       //go:build ignore

Save and exit
```

Once downloaded, the module and testing components can be built using the following commands:

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

11.2.2 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-a430310d6563c0734ddafca7731570dfb683dc19.tar.xz or by issuing the following command:

```
wget https://commondatastorage.googleapis.com/chromium-boringssl-fips/boringssl-
a430310d6563c0734ddafca7731570dfb683dc19.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:

2d5339b756dbf1ceb4fdc4b1c8f19e32ded055292dc57827a6592f15ca9d359f

By issuing the following command:

```
sha256sum boringssl-a430310d6563c0734ddafca7731570dfb683dc19.tar.xz
```

The tarball can be extracted using the following command:

```
tar xJ < boringssl-a430310d6563c0734ddafca7731570dfb683dc19.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
```

Retrieving Module name and version

The following methods will provide the module name and versions:

- FIPS_module_name() – BoringCrypto
- FIPS_version() – 2023042800

11.3 Crypto Officer Guidance

11.3.1 Usage of 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.

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 SP800-52rev2 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 SP800-52rev2, 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 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.

11.3.2 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 800-131A. Therefore, the 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.

11.3.3 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.

11.3.4 Modes of Operation

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 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.

12. Mitigation of Other Attacks

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

13. 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 800-38D	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 800-56Arev3	Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography
SP 800-90Ar1	Recommendation for Random Number Generation Using Deterministic Random Bit Generators
SP 800-90B	Recommendation for the Entropy Sources Used for Random Bit Generation
SP 800-131Ar2	Transitioning the Use of Cryptographic Algorithms and Key Lengths
SP 800-133r2, [133]	Recommendation for Cryptographic Key Generation
SP 800-135rev1	Recommendation for Existing Application-Specific Key Derivation Functions

Table 19 – References and Standards

14. Acronyms

Acronym	Definition
ADB	Android Debug Bridge
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
CPU	Central Processing Unit
CRNGT	Continuous Random Number Generator Test
CSP	Critical Security Parameter
CTR	Counter-mode
CVL	Component Validation List

Acronym	Definition
DEP	Default Entry Point
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
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FCC	Federal Communications Commission
FIPS	Federal Information Processing Standards
GCM	Galois/Counter Mode
GMAC	Galois Message Authentication Code
GPC	General Purpose Computer
GPOS	General Purpose Operating System
HMAC	Key-Hashed Message Authentication Code
IETF	Internet Engineering Task Force
IG	Implementation Guidance
IV	Initialization Vector
KAS	Key Agreement Scheme
KAT	Known Answer Test
KDF	Key Derivation Function
KTS	Key Transport Scheme
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
NDRNG	Non-Deterministic Random Number Generator
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

Acronym	Definition
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SP	Special Publication
SSL	Secure Socket Layer
TCBC	Triple-DES Cipher-Block Chaining
TDEA	Triple Data Encryption Algorithm
TECB	Triple-DES Electronic Code Book
TLS	Transport Layer Security
Triple-DES	Triple Data Encryption Standard

Table 20 – Acronyms