Google, LLC.
BoringCrypto

FIPS 140-2 Security Policy

Software Version:

853ca1ea1168dff08011e5d42d94609cc0ca2e27

Date: May 6th, 2022

2400 Research Blvd, Suite 395
Rockville, MD 20850
tel: +1 (703) 375-9820
info@acumensecurity.net
www.acumensecurity.net

Introduction

Federal Information Processing Standards Publication 140-2 — 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.

More information is available on the CMVP website at:
https://csrc.nist.gov/groups/STM/cmvp/index.html.

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 140-2.

The BoringCrypto module may also be referred to as the “module” in this document.

Disclaimer

Notices

Table of Contents

List of Tables

List of Figures

1. Introduction

Google, LLC BoringCrypto module (hereafter referred to as the "module") is an open-source, general-purpose cryptographic library which provides FIPS 140-2 approved cryptographic algorithms to serve BoringSSL and other user-space applications. The validated version of the library is 853ca1ea1168dff08011e5d42d94609cc0ca2e27. For the purposes of the FIPS 140-2 validation, its embodiment type is defined as multi-chip standalone.

The cryptographic module was tested on the following operational environments on the general-purpose computer (GPC) platforms detailed below:

#	Operational Environment	Processor Family
1	Debian Linux 5.10.40 (Rodete) on HPE Z620	Intel Xeon E5-2680 with PAA
2	Debian Linux 5.10.40 (Rodete) on HPE Z620	Intel Xeon E5-2680 without PAA
4	Google prodimage with Linux 4.15.0 on Google Arcadia-Rome	AMD Rome with PAA
5	Google prodimage with Linux 4.15.0 on Google Arcadia-Rome	AMD Rome without PAA
5	Google prodimage with Linux 4.15.0 on Ampere Altra SoC	ARM Neoverse N1 with PAA
6	Google prodimage with Linux 4.15.0 on Ampere Altra SoC	ARM Neoverse N1 without PAA

Table 1 - Tested Operational Environments

The cryptographic module is also supported on the following operating environments for which operational testing and algorithm testing was not performed:

- Linux 4.X executing on x86_64 architecture;
- Linux 4.X executing on Aarch64 architecture;
- Linux 5.X executing on x86_64 architecture;
- Linux 5.X executing on Aarch64 architecture

As per FIPS 140-2 Implementation Guidance G.5, compliance is maintained for other versions of the respective operational environments where the module binary is unchanged. No claim can be made as to the correct operation of the module or the security strengths of the generated keys if any source code is changed and the module binary is reconstructed.

The GPC(s) used during testing met Federal Communications Commission (FCC) Electromagnetic Interference (EMI) and Electromagnetic Compatibility (EMC) requirements for business use as defined by 47 Code of Federal Regulations, Part 15, Subpart B. FIPS 140-2 validation compliance is maintained when the module is operated on other versions of the GPOS running in single user mode, assuming that the requirements outlined in FIPS 140-2 IG G.5 are met.

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

2. FIPS 140-2 Security Levels

The following table lists the level of validation for each area in FIPS 140-2:

FIPS 140-2 Section Title	Validation Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services, and Authentication	1
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
Electromagnetic Interference / Electromagnetic Compatibility	1
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	N/A
Overall Level	1

Table 2 – Validation Level by FIPS 140-2 Section

3. Cryptographic Module Specification

3.1 Cryptographic Boundary

The module is a software library providing a C-language Application Program Interface (API) for use by other processes that require cryptographic functionality. All operations of the module occur via calls from host applications and their respective internal daemons/processes. As such, there are no untrusted services calling the services of the module.

The physical cryptographic boundary is the general-purpose computer on which the module is installed. The logical boundary of the module is a single object file named bcm.o, which is linked into the libcrypto.so shared library. The module performs no communications other than with the calling application (the process that invokes the module services) and the host operating system.

Figure 1 shows the logical relationship of the cryptographic module to the other software and hardware components of the computer:

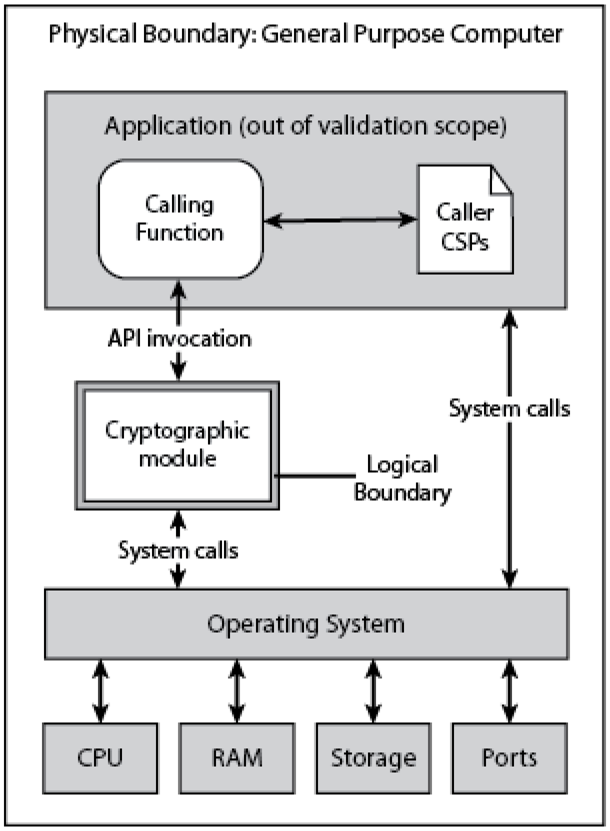

Figure 1 – Logical Boundary

4. Modes of Operation

The module supports two modes of operation: Approved and Non-Approved. The module will be in FIPS-approved mode when all power up Self-Tests have completed successfully, and only Approved or allowed algorithms are invoked. See Table 7 for a list of the supported Approved algorithms and Table 8 for allowed algorithms. The non-Approved mode is entered when a non-Approved algorithm is invoked. See Table 9 for a list of non-Approved algorithms.

5. Cryptographic Module 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.

FIPS Interface	Physical Ports	Logical Interfaces
Data Input	Physical ports of the tested platforms	API input parameters
Data Output	Physical ports of the tested platforms	API output parameters and return values
Control Input	Physical ports of the tested platforms	API input parameters
Status Output	Physical ports of the tested platforms	API return values
Power Input	Physical ports of the tested platforms	N/A

Table 3 – 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.

6. Roles, Authentication and Services

The cryptographic module implements both User and Crypto Officer (CO) roles. The module does not support user authentication. The User and CO roles are implicitly assumed by the entity accessing services implemented by the module. A user is considered the owner of the thread that instantiates the module and, therefore, only one concurrent user is allowed.

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:

Service	Approved security functions	Keys and/or CSPs	Roles	Access rights to keys and/or CSPs
Module Initialization	N/A	N/A	CO	N/A
Symmetric encryption/decryption	AES, Triple-DES	AES, Triple-DES symmetric keys	User, CO	Execute
Keyed hashing	HMAC-SHA	HMAC key	User, CO	Execute
Hashing	SHS	None	User, CO	N/A
Random Bit Generation	CTR_DRBG	DRBG seed, internal state V and Key values	User, CO	Write/Execute
Signature generation/ verification	CTR_DRBG RSA ECDSA	RSA, ECDSA private key	User, CO	Write/Execute
Key Transport	RSA	RSA private key	User, CO	Write/Execute
Key Agreement	KAS ECC	EC DH private key	User, CO	Write/Execute
Key Generation	CTR_DRBG RSA ECDSA	RSA, ECDSA private key	User, CO	Write/Execute
On-Demand Self-test	N/A	N/A	User, CO	Execute
Zeroization	N/A	All keys	User, CO	Write/Execute
Show status	N/A	N/A	User, CO	N/A

Table 4 – Approved Services, Roles and Access Rights

The module provides the following non-Approved services which utilize algorithms listed in Table 9:

Service	Non-Approved Functions	Roles	Keys and/or CSPs
Symmetric encryption/decryption	AES (non-compliant), DES, Triple-DES (non-compliant)	User, CO	N/A
Hashing	MD4, MD5, POLYVAL, GHASH	User, CO	N/A
Signature generation/ verification	RSA (non-compliant) ECDSA (non-compliant)	User, CO	N/A
Key Transport	RSA (non-compliant)	User, CO	N/A
Key Generation	RSA (non-compliant) ECDSA (non-compliant)	User, CO	N/A

Table 5 – non-Approved or non-security relevant services

The module also provides the following non-Approved or non-security relevant services over a non-public interface:

Service	Approved Security Functions	Roles	Access rights to keys and/or CSPs
Large integer operations	None	User, CO	N/A
Disable automatic generation of CTR_DRBG "additional_input" parameter	CTR_DRBG	User, CO	N/A
Wegman-Carter hashing with POLYVAL	None	User, CO	N/A

Table 6 - Non-Security Relevant Services

7. Physical Security

The cryptographic module is comprised of software only and thus does not claim any physical security.

8. Operational Environment

The cryptographic module operates under Debian Linux 5.10.40 and Google prodimage with Linux 4.15.0. The module runs on a GPC or Server running one of the operating systems specified in Table 1. 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.

9. Cryptographic Algorithms & Key Management

9.1 Approved Cryptographic Algorithms

The module implements the following FIPS 140-2 Approved algorithms:

CAVP Cert #	Algorithm	Standard	Mode/Method/Size	Use
A1657	AES	FIPS 197 SP 800-38A	128, 192, 256 CBC, ECB, CTR	Encryption, Decryption
	AES	SP 800-38C	128 CCM	Authenticated Encryption, Authenticated Decryption
	AES	SP 800-38D	128, 192, 256 GCM/GMAC	Authenticated Encryption, Authenticated Decryption
	KTS	SP 800-38F	128, 192, 256 KW, KWP	Key Wrapping, Key Unwrapping
	CVL	SP 800-135rev1	TLS 1.0/1.1 and 1.2 KDF[1]	Key Derivation
	DRBG	SP 800-90Arev1	AES-256 CTR_DRBG	Random Bit Generation
	ECDSA	FIPS 186-4	Signature Generation Component, Key Pair Generation, Signature Generation, Signature Verification, Public Key Validation P-224, P-256, P-384, P-521	Digital Signature Services
	HMAC	FIPS 198-1	HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	Generation, Authentication
	RSA	FIPS 186-4	Key Generation[2], Signature Generation, Signature Verification (1024, 2048, 3072, 4096-bit)[3]	Digital Signature Services
	SHS	FIPS 180-4	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 and SHA-512/256	Digital Signature Generation, Digital Signature Verification, non-Digital Signature Applications

[1] The module supports FIPS 140-2 approved/allowed cryptographic algorithms for TLS 1.0, 1.1 and 1.2.

[2] SHA-1 shall only be utilized for digital signature and general hashing operations.

[3] 1024 Key Size is only used for Signature Verification

	Triple-DES	SP 800-67 SP 800-38A	Three Key TCBC, TECB (provides 112 bits of security strength)	Encryption[4], Decryption
	KAS-SSC	SP 800-56rev3	EC Diffie-Hellman P-224, P-256, P-384 and P-521 Diffie-Hellman FB, FC	Key Agreement Scheme Shared Secret Computation (KAS-SSC) per SP 800-56Arev3
	KAS	SP 800-56rev3 SP 800-135rev1	EC Diffie-Hellman P-224, P-256, P-384 and P-521 Diffie-Hellman FB, FC with TLS 1.0/1.1 or 1.2 KDF	Key Agreement Scheme per SP 800-56Arev3 with Key Derivation per SP 800-135

Table 7 – Approved Algorithms and CAVP Certificates

9.2 Allowed Cryptographic Algorithms

The module supports the following non-FIPS 140-2 Approved but allowed algorithms that may be used in the Approved mode of operation:

Algorithm	Use
RSA Key Transport	Key establishment methodology using PKCS#1-v1.5 per section 8.1 of RFC 2313 provides between 112 and 256 bits of encryption strength
MD5	When used with the TLS protocol version 1.0 and 1.1

Table 8 – Allowed Algorithms

9.3 Non-Approved Cryptographic Algorithms

The module employs the methods listed in Table 9, which are not allowed for use in a FIPS-Approved mode. Their use will result in the module operating in a non-Approved mode.

MD5, MD4	DES
AES-GCM (non-compliant)	AES (non-compliant)
ECDSA (non-compliant)	RSA (non-compliant)
POLYVAL	Triple-DES (non-compliant)
GHASH (non-compliant)	

Table 9 – Non-Approved Algorithms

[4] After December 31st, 2023, three-key TDEA is disallowed for encryption unless specifically allowed by other NIST guidance. Decryption using three-key TDEA is allowed for legacy use.

9.4 Cryptographic Key Management

The table below provides a complete list of Private Keys and CSPs used by the module:

Key/CSP Name	Key Description	Generated/ Input	Output
AES Key	AES (128/192/256) encrypt / decrypt key	Input via API in plaintext	Output via API in plaintext
AES-GCM Key	AES (128/192/256) encrypt / decrypt / generate / verify key	Input via API in plaintext	Output via API in plaintext
AES Wrapping Key	AES (128/192/256) key wrapping key	Input via API in plaintext	Output via API in plaintext
Triple-DES Key	Triple-DES (3-Key) encrypt / decrypt key	Input via API in plaintext	Output via API in plaintext
ECDSA Signing Key	ECDSA (P-224/P-256/P-384/P-521) signature generation key	Internally Generated or input via API in plaintext	Output via API in plaintext
EC DH Private Key	EC DH (P-224/P-256/P-384/P-521) private key	Internally Generated or input via API in plaintext	Output via API in plaintext
HMAC Key	Keyed hash key (160/224/256/384/512)	Input via API in plaintext	Output via API in plaintext
RSA Key (Key Transport)	RSA (2048 to 16384 bits) key decryption (private key transport) key	Internally Generated or input via API in plaintext	Output via API in plaintext
RSA Signature Generation Key	RSA (2048 to 16384 bits) signature generation key	Internally Generated or input via API in plaintext	Output via API in plaintext
TLS Pre-Master Secret	Shared Secret; 48 bytes of pseudo-random data	Internally Generated	Output via API in plaintext
TLS Master Secret	Shared Secret; 48 bytes of pseudo-random data	Internally Derived via key derivation function defined in SP 800-135 KDF (TLS).	Output via API in plaintext
CTR_DRBG V (Seed)	128 bits	Internally Generated	Does not exit the module
CTR_DRBG Key	256 bits	Internally Generated	Does not exit the module
CTR_DRBG Entropy Input	384 bits	Input via API in plaintext	Does not exit the module

Table 10 – Keys and CSPs supported

9.5 Public Keys

The table below provides a complete list of the Public keys used by the module:

Public Key Name	Key Description
ECDSA Verification Key	ECDSA (P-224/P-256/P-384/P-521) signature verification key
EC DH Public Key	EC DH (P-224/P-256/P-384/P-521) public key
RSA Key (Key Transport)	RSA (2048 to 16384 bits) key encryption (public key transport) key

RSA Signature Verification Key	RSA (1024 to 16384 bits) signature verification public key

Table 11 – Public keys supported

9.6 Key Generation

The module supports generation of ECDSA, EC Diffie-Hellman, and RSA key pairs as specified in Section 5 of NIST SP 800-133. The module employs a NIST SP 800-90A random bit generator for creation of the seed for asymmetric key generation. The module receives entropy passively, and a minimum of 112 bits of entropy must be supplied.

The output data path is provided by the data interfaces and is logically disconnected from processes performing key generation or zeroization. No key information will be output through the data output interface when the module zeroizes keys.

9.7 Key Storage

The cryptographic module does not perform persistent storage of keys. Keys and CSPs are passed to the module by the calling application. The keys and CSPs are stored in memory in plaintext. Keys and CSPs residing in internally allocated data structures (during the lifetime of an API call) can only be accessed using the module defined API. The operating system protects memory and process space from unauthorized access.

9.8 Key Zeroization

The module is passed keys as part of a function call from a calling application and does not store keys persistently. The calling application is responsible for parameters passed in and out of the module. The Operating System and the calling application are responsible to clean up temporary or ephemeral keys.

All CSPs can be zeroized by power cycling or by rebooting the host test platform.

10. Self-tests

FIPS 140-2 requires the module to perform self-tests to ensure the integrity of the module and the correctness of the cryptographic functionality at start up. Some functions require conditional tests during normal operation of the module. The supported tests are listed and described in this section.

10.1 Power-On Self-Tests

Power-on self-tests are run upon the initialization of the module and do not require operator intervention to run. If any of the tests fail, the module will not initialize. The module will enter an error state and no services can be accessed.

The module implements the following power-on self-tests:

Type	Test
Integrity Test	HMAC-SHA-256
Known Answer Test	AES KAT (encryption and decryption. Key size: 128-bits)
	AES-GCM KAT (encryption and decryption. Key size: 128-bits)
	Triple-DES KAT (encryption and decryption. Key size: 168-bits)
	ECDSA KAT (signature generation/signature verification. Curve: P-256)
	HMAC KAT (HMAC-SHA-1, HMAC-SHA-512)
	SP 800-90A CTR_DRBG KAT (Key size: 256-bits)
	RSA KAT (signature generation/signature verification and encryption/decryption. Key size: 2048-bit)
	TLS v1.2 KDF KAT
	Primitive "Z" Computation KAT
	KAS-ECC KAT (Curve P-256)
	KAS-FFC KAT (2048-bit)
	SHA KAT (SHA-1, SHA-256, SHA-512)

Table 12 – Power-on Self-tests

By default, all power up self-tests are executed at module initialization. The module can be configured to only run the integrity test on subsequent instantiations by setting the environmental variable BORINGSSL_FIPS_SELF_TEST_FLAG_FILE, as allowed by FIPS 140-2 IG 9.11. If configured, after the self-tests have passed, the module creates a temporary file named after the module's HMAC-SHA-256 integrity value (this value does not persist across power cycles). This file is checked for existence whenever subsequent instantiations of the module are initialized. If it exists, only the integrity test is run. If the environmental variable is not set, the file does not exist, or the file cannot be accessed for any reason, the entire set of power-on self-tests (KATs and integrity test) are run. The power-on self-tests must be passed before a User/Crypto Officer can perform services. The Power-on self-tests can be run on demand by power-cycling the host platform.

10.2 Conditional Self-Tests

Conditional self-tests are run during operation of the module. 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 reinitialized to clear the error and resume FIPS mode of operation. Each module performs the following conditional self-tests:

Type	Test
Pair-wise Consistency Test	ECDSA Key Pair generation RSA Key Pair generation
DRBG Health Tests	Performed on DRBG, per SP 800-90A Section 11.3. Required per IG C.1. Also includes a continuous test.

Table 13 – Conditional Self-tests

Pairwise consistency tests are performed for both possible modes of use, e.g. Sign/Verify and Encrypt/Decrypt.

11. Mitigation of other Attacks

The module is not designed to mitigate against attacks which are outside of the scope of FIPS 140-2.

12. Guidance and Secure Operation

12.1 Installation Instructions

The following steps shall be performed to build, compile and statically link the BoringCrypto module to BoringSSL on the tested Operational Environments.

The below tools are required in order to build and compile the module:

- Clang compiler version 12.0.0 (http://releases.llvm.org/download.html)
- Go programming language version 1.16.5 (https://golang.org/dl/)
- Ninja build system version 1.10.2 (https://github.com/ninja-build/ninja/releases)
- Cmake version 3.20.1 (https://cmake.org/download/)

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 140-2 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-853ca1ea1168dff08011e5d42d94609cc0ca2e27.tar.xz

or by issuing the following command:

wget https://commondatastorage.googleapis.com/chromium-boringssl-fips/boringssl-853ca1ea1168dff08011e5d42d94609cc0ca2e27.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:

a4d069ccef6f3c7bc0c68de82b91414f05cb817494cd1ab483dcf3368883c7c2

By issuing the following command:

- sha256sum boringssl-853ca1ea1168dff08011e5d42d94609cc0ca2e27.tar.xz

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

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

Upon completion of the build process the compilation can be validated by running the following command and observing that no tests fail:

- ./crypto/crypto_test

The module's status can be verified by issuing:

- ./tool/bssl isfips

The module will print "1" if it is in a FIPS 140-2 validated mode of operation.

12.2 Secure Operation

12.2.1 Initialization

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. The module is designed with a default entry point (DEP) which ensures that the power-up tests are initiated automatically when the module is loaded.

12.2.2 Usage of AES OFB, CFB and CFB8

In approved mode, users of the module must not utilize AES OFB, CFB and CFB8.

12.2.3 Usage of AES-GCM

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

AES GCM encryption and decryption are used in the context of the TLS protocol version 1.2 (compliant to Scenario 1 in FIPS 140-2 A.5). 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 and 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 140-2 IG A.5 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 IG A.5, 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.

12.2.4 Usage of Triple-DES

In accordance with CMVP IG A.13, when operating in a FIPS approved mode of operation, the same Triple-DES key shall not be used to encrypt more than 2^{20} or 2^{16} 64-bit data blocks.

The TLS protocol governs the generation of the respective Triple-DES keys. Please refer to IETF RFC 5246 (TLS) for details relevant to the generation of the individual Triple-DES encryption keys. The user is responsible for ensuring that the module limits the number of encrypted blocks with the same key to no more than 2^{20} when utilized as part of a recognized IETF protocol.

For all other uses of Triple-DES, the user is responsible for ensuring that the module limits the number of encrypted blocks with the same key to no more than 2^{16}.

12.2.5 Usage of RSA and ECDSA

The module allows the use of 1024 bit RSA keys for legacy purposes including signature generation, which is disallowed in FIPS 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 implicitly.

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

SHA-1 shall only be utilized for digital signature verification and non-digital signature applications per NIST SP 800-131Ar2.

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.

13. References and Standards

The following Standards are referred to in this Security Policy:

Abbreviation	Full Specification Name
FIPS 140-2	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 140-2 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 800-56Ar3	Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography
SP 800-67	Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher
SP 800-90A	Recommendation for Random Number Generation Using Deterministic Random Bit Generators
SP 800-131Ar2	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

Table 14 – References and Standards

14. Acronyms and Definitions

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
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
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 15 – Acronyms and Definitions