Eclypses, Inc.

Eclypses Cryptographic Library

Library Version 1.0.0

FIPS 140-3 Security Policy

Document Version 1.0

Table of Contents

1 General: Security Level

The module meets the overall requirements of FIPS 140-3 Level 1.

Table 1: Module Security Levels

ISO/IEC 24759 Section 6 Subsection	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

2 Cryptographic Module Specification

2.1 Overview

The cryptographic boundary is defined as the ‘Eclypses Cryptographic Library’. The ‘Eclypses Cryptographic Library’ (hereafter referred to as the "Module") version 1.0.0 is a software library supporting FIPS 140-3 approved cryptographic algorithms. The Module is a Software Module with the ability to use PAA when present. The Module is a proprietary implementation, general- purpose cryptographic library with an API targeted toward high performance and minimal footprint. It was designed to have no dependencies or interactions with the operating system or any other libraries; the only interactions are to perform cryptographic algorithms when requested by application code. It was designed to support a wide range of operating environments with as much common code as possible. For this reason, the Software Module designation was chosen.

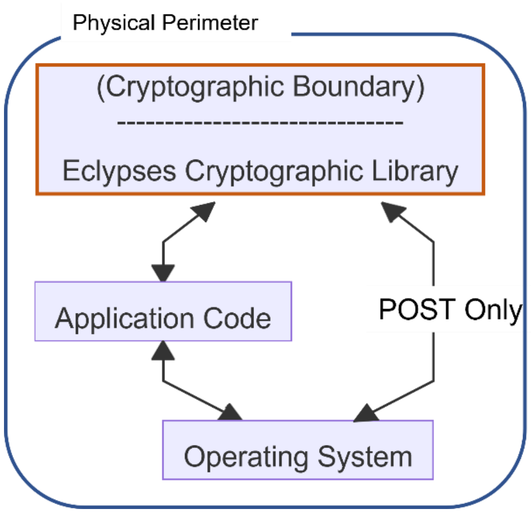

Figure 1: Cryptographic Boundary and Physical Perimeter

The Eclypses Cryptographic Library Module consists of the following files, depending on operational environment (*refer to Table 2 for OE numbers in brackets*):

- libecl.so, libecl.so.sha256
    - Google Android 9 (32-bit) [7]
    - Google Android 9 (64-bit) [8, 9]
    - OpenWRT Linux 19.07 (32-bit) [12]
    - Raspbian Linux 5.10 (32-bit) [13]
    - Ubuntu Linux 20.04 (64-bit) [14, 15]

- libecl.dylib, libecl.dylib.sha256
    - Apple iOS 13.5 (64-bit) [1, 2]
    - Apple macOS Big Sur 11.3 (64-bit) [3, 4, 5, 6]
- ecl.dll, ecl.dll.sha256
    - Microsoft Windows 10 Pro 20H2 (64-bit) [10, 11]

Table 2: Tested Operational Environments (OE)

#	Operating System	Hardware Platform	Processor	PAA / Acceleration
1	Apple iOS 13.5 (64-bit)	iPhone SE	Apple A9 (APL1022) with Cryptography Extensions (ARM v8)	Yes
2	Apple iOS 13.5 (64-bit)	iPhone SE	Apple A9 (APL1022) (ARM v8)	No
3	Apple macOS Big Sur 11.3 (64-bit)	MacBook Pro	Intel Core i9 (I9-9880H) with AES-NI, AVX, SSE2, and SSSE3	Yes
4	Apple macOS Big Sur 11.3 (64-bit)	MacBook Pro	Intel Core i9 (I9-9880H)	No
5	Apple macOS Big Sur 11.3 (64-bit)	Mac mini	Apple M1 (APL1102) with Cryptography Extensions (ARM v8)	Yes
6	Apple macOS Big Sur 11.3 (64-bit)	Mac mini	Apple M1 (APL1102) (ARM v8)	No
7	Google Android 9 (32-bit)	Oukitel C16	MediaTek MT6580P (ARM v7)	No
8	Google Android 9 (64-bit)	Samsung Galaxy S8	Qualcomm Snapdragon 835 with Cryptography Extensions (ARM v8)	Yes
9	Google Android 9 (64-bit)	Samsung Galaxy S8	Qualcomm Snapdragon 835 (ARM v8)	No
10	Microsoft Windows 10 Pro 20H2 (64-bit)	HP ZBook	Intel Core i7 (i7-7500U) with AES-NI, AVX, SSE2, and SSSE3	Yes
11	Microsoft Windows 10 Pro 20H2 (64-bit)	HP ZBook	Intel Core i7 (i7-7500U)	No
12	OpenWRT Linux 19.07 (32-bit)	GL.iNet GL-AR750S	Qualcomm QCA9563 (MIPS 74Kc)	No

13	Raspbian Linux 5.10 (32-bit)	Raspberry Pi 4 Model B Rev 1.1	Broadcom BCM2711 (ARM v8)	No
14	Ubuntu Linux 20.04 (64-bit)	HP ZBook	Intel Core i7 (i7-7500U) with AES-NI, AVX, SSE2, and SSSE3	Yes
15	Ubuntu Linux 20.04 (64-bit)	HP ZBook	Intel Core i7 (i7-7500U)	No

Refer to Table 1: Module Security Levels for the security rating of the Module and security levels of individual areas.

The cryptographic boundary is defined by the library and API. All implementation is inside the boundary. The library contains only cryptographic functions, and the library performs all cryptographic functions itself without dependency on any other library, application, or operating system functionality.

2.2 Modes Of Operation

There is only one mode of operation: Normal Approved Mode.

The software development kit (SDK) package clearly indicates the compliance mode by including "FIPS" in the package name to indicate the compliance. There is a service (*Show Status*) the library user can call to verify that Normal Approved Mode is enabled.

The Module name may be validated by using the *Show Module Name / Identifier* service. An exact match to the below listed Module name validates the correctness. The approved Module name is:

Eclypses Cryptographic Library

The Module version may be validated by using the Versioning Information service. An exact match to the below listed Module version validates the correctness. The approved Module version is:

1.0.0

2.3 Security Functions

2.3.1 Approved Algorithms

The module supports the Approved algorithms listed in Table 3.

Table 3: Approved Algorithms

CAVP Cert(s)	Algorithm	Standards	Modes / Methods	Description / Key Sizes	Use / Function
A1406, A1422	AES	FIPS 197 [2], NIST SP 800-38A [3]	CBC	128, 192, and 256 bits	Data Encryption / Decryption
A1407, A1423	AES	FIPS 197 [2], NIST SP 800-38A [3]	CTR	128, 192, and 256 bits	Data Encryption / Decryption
A1408, A1424	AES	FIPS 197 [2], NIST SP 800-38A [3]	ECB	128, 192, and 256 bits	Data Encryption / Decryption
A1414, A1425	CTR_DRBG	NIST SP 800-90A [4]	AES-128-NODF AES-192-NODF AES-256-NODF	CTR_DRBG using AES without derivation function	Deterministic Random Bit Generation
A1413, A1426	CTR_DRBG	NIST SP 800-90A [4]	AES-128-DF AES-192-DF AES-256-DF	CTR_DRBG using AES with derivation function	Deterministic Random Bit Generation
A1412, A1419, A1427, A1432	Hash_DRBG	NIST SP 800-90A [4]	SHA-1 SHA- 256 SHA-512	HASH_DRBGs using SHA	Deterministic Random Bit Generation
A1409, A1428, A3926[1], A3929[2]	SHA-1[3]	FIPS 180-4 [1]	SHA-1	Message Digests Supports input message lengths from 1 byte to 8 GB depending on OE	Message Digest Creation
A1410, A1418, A3927[4], A3930[5]	SHA-256	FIPS 180-4 [1]	SHA-256	Message Digests Supports input message lengths from 1 byte to 8 GB depending on OE	Message Digest Creation
A1411, A3928[6], A3931[7]	SHA-512	FIPS 180-4 [1]	SHA-512	Message Digests Supports input message lengths from 1 byte to 8 GB depending on OE	Message Digest Creation

[1] SHA-1 Large Data Test (LDT) only applicable to OE #3, 5, 10, and 14 (w/ PAA enabled) from Table 2.
[2] SHA-1 LDT only applicable to OE #4, 6, 11, and 15 from Table 2.
[3] SHA-1 shall only be used for non-digital signature applications (ref: NIST SP 800-131r2, Section 9).
[4] SHA-256 LDT only applicable to OE #3, 4, 6, 10, 11, 14, and 15 from Table 2.
[5] SHA-256 LDT only applicable to OE #5 (w/ PAA enabled) from Table 2.
[6] SHA-512 LDT only applicable to OE #3, 4, 6, 10, 11, 14, and 15 from Table 2.
[7] SHA-512 LDT only applicable to OE #5 (w/ PAA enabled) from Table 2.

2.4 Security Design and Rules of Operation

This section documents the security rules enforced by the cryptographic module to implement the security requirements of a FIPS 140-3 Level 1 Module.

2.4.1 CSP Protection

For AES, the CSP (key) is used by the *Symmetric Encryption / Decryption* service. The library user shall protect or zeroise their copy of the key after keying. The *Zeroisation* service is provided to zeroise the library's key, and the library user shall use this service when the AES module is no longer needed for encryption or decryption.

For DRBGs, the CSP (entropy) is used by the Random Number Generation service. As soon as the entropy has been consumed as part of the Random Number Generation service, it is zeroised. The library user shall zeroise any additional copies of the entropy they may have. The *Zeroisation* service is provided to zeroise the library's CSP (the current state of the DRBG, including V, C, and/or Key, depending on the DRBG), and the library user shall use this service when the DRBG module is no longer needed to generate random bits.

2.4.2 Key Creation (N/A)

There is no assurance of minimum strength of generated random strings or SSPs. As such, the Module cannot create keys and can only import keys. There is also no assurance of minimum security of SSPs that are externally loaded (or of SSPs established with externally loaded SSPs). The DRBGs produce random strings, but the library cannot use them as keys. The library user may use the random strings as they choose. The AES services use keys provided by the library user. The library user shall protect the user provided cryptographic keys.

2.4.3 Indicators

All services return a status code to indicate to the library user the success or failure of the operation. The only exceptions to this rule are the *Control PAA* service and *Show Module Name/Identifier and Versioning Information* service, which indicate success by their completion. The library user shall check every status code and take appropriate action based on the result.

2.4.4 SHA-1

When utilizing the Message Digest service, the output for SHA-1 shall only be used for non-digital signature applications (*ref: NIST SP 800-131r2, Section 9*).

2.4.5 Inhibiting Output (Error State)

When the consuming application starts, the Module automatically commands the Perform Pre- Operational Self-Tests service, which includes the Perform Conditional Self-Tests service in addition to the library integrity test. The Perform Pre-Operational Self-Tests service will set the error state if the library integrity check or the Conditional Self-Tests fail. All output is inhibited while in an error state; therefore, all output is inhibited until the library integrity check and Conditional Self-Tests pass.

For AES, if the Module is in an error state, encryption and decryption each return an error status. No encryption or decryption is performed while in the error state.

For DRBGs, if the Module or instance is in an error state, instantiation and generation each return an error status. No random bits are created while in the error state.

For SHA, if the Module is in an error state, feeding data to be digested returns an error status. No message digest update is performed while in the error state.

2.4.6 Inhibiting Output (Self-Test)

The Module has a self-test flag to indicate when self-tests are running. While this flag is set, all output is inhibited. All self-test results are zeroised and are not output.

2.4.7 Inhibiting Output (Zeroisation)

The Module is in an error state during zeroisation. Since output is inhibited while in an error state, output is inhibited during zeroisation.

2.5 Initialization Requirements

To use PAA on ARM64 platforms, the PAA availability must be known or probed by the library user in some way, and then communicated to the library using the Control PAA service. The Perform Conditional Self-Tests service must then be commanded for the PAA option to take effect (the PAA and non-PAA were self-tested during the Pre-Operational Self-Tests, which includes the Conditional Self-Tests in addition to the library integrity test; the self-test requirement here is just part of the PAA enable/disable procedure). This applies to the following operational environments (refer to Table 2 for OE numbers in brackets):

- [1] Apple iOS 13.5 (64-bit) Apple A9 (APL1022) with Cryptography Extensions (ARM v8)
- [5] Apple macOS Big Sur 11.3 (64-bit) Apple M1 (APL1102) with Cryptography Extensions (ARM v8)
- [8] Google Android 9 (64-bit) Qualcomm Snapdragon 835 with Cryptography Extensions (ARM v8)

To use PAA on x86_64 platforms, the choice must be communicated to the library using the Control PAA service. If the user chooses to allow PAA, the library probes to determine if PAA is available and uses it if so. The Perform Conditional Self-Tests service must then be commanded for the PAA option to take effect (the PAA and non-PAA were self-tested during the Pre-Operational Self-Tests, which includes the Conditional Self-Tests in addition to the library integrity test; the self-test requirement here is just part of the PAA enable/disable procedure). This applies to the following operational environments (refer to Table 2 for OE numbers in brackets):

- [3] Apple macOS Big Sur 11.3 (64-bit) Intel Core i9 (I9-9880H) with AES-NI, AVX, SSE2, and SSSE3
- [10] Microsoft Windows 10 Pro 20H2 (64-bit) Intel Core i7 (i7-7500U) with AES-NI, AVX, SSE2, and SSSE3
- [14] Ubuntu Linux 20.04 (64-bit) Intel Core i7 (i7-7500U) with AES-NI, AVX, SSE2, and SSSE3

3 Cryptographic Module Interfaces

Table 4: Physical Ports and Logical Interfaces

Physical Port	Logical Interface	Description
N/A	Data Input, Data Output, Control Input, Status Output	The logical interface is a C API. All data input, data output, control input, and status output are done via the C API. All SSPs are given to the API or requested by it via C callbacks. No SSPs are generated by the Module. All SSPs consumed by the module are passed in as parameters except DRBG entropy and nonce, which are loaded via callback functions that conform with the requirements in NIST SP 800- 90A [4] section 9.1. DRBG entropy and nonce cannot be provided as input parameters to the instantiate function. The Module accepts a C function pointer that implements the *Get_entropy_input* function defined in NIST SP 800-90A [4] Section 9 in order to request the entropy as required. The Module accepts a second C function pointer to acquire the nonce in a similar manner. Both callback function interfaces are described below. This is the Software Module Interface.
N/A	Data Input	When a DRBG instantiates, it needs entropy. The C API invokes a callback function registered by the library user at the moment the entropy is needed. The callback function requires the library user to acquire the entropy and load it in to either a provided buffer or their own buffer for use by the instantiation procedure. Immediately after use by the instantiation procedure, the entropy buffer (either provided by the Module or by the user) is zeroised. The callback function is provided with the minimum number of bits of entropy required and the minimum/maximum length of the entropy bitstring that must contain the required number of bits of entropy for the desired security strength. The minimum number of bits of entropy provided is equal to the security strength for each Hash_DRBG and each CTR_DRBG with derivation function (i.e., 128, 192, or 256 bits), and equal to the sum of the block size and key length for each CTR_DRBG without derivation function (i.e., 256, 320, or 384 bits). The callback function must return the status and entropy bitstring. The callback function must return an error if the minimum number of entropy bits are not available. The instantiation procedure verifies the status and entropy bitstring size constraints and returns an error if the status is not success or the entropy bitstring is not a valid size. This is the DRBG Entropy Callback Function.
N/A	Data Input	When a DRBG instantiates, it may need a nonce, depending on the DRBG algorithm. The API invokes a callback function registered by the library user at the moment the nonce is needed. The callback function must load the nonce in a provided buffer. The instantiation procedure verifies the nonce bitstring size constraints. This is the DRBG Nonce Callback Function.

N/A	Data Input	Upon module initialization the module queries the CPU to determine if PAA is available. If PAA is available both non-PAA and PAA cryptographic algorithms are self-tested (refer to Section 10).

4 Roles, Services, and Authentication

4.1 Roles

There is one role for this Module: Cryptographic Officer (aka. Crypto Officer or CO).

4.1.1 Crypto Officer

The Crypto Officer role is supported for installation of the dynamic library. The Crypto Officer controls access to the dynamic library to ensure it is not tampered with before and after installation. The Crypto Officer can use all the cryptographic services of the Module.

4.2 Service Inputs and Outputs

4.2.1 Approved Service Inputs and Outputs

The role names are abbreviated as: CO = Crypto Officer

Table 5: Roles, Approved Service Commands, Input, And Output

Roles with Service Access	Service	Input	Output
CO	Control PAA	N/A	N/A
CO	Message Digest	Data to be digested	Message digest, status
CO	Perform Conditional Self-Tests	N/A	Self-test status
CO	Perform Pre-Operational Self-Tests	N/A	Self-test status
CO	Random Number Generation	Entropy, nonce, and personalization string	Random number, status
CO	Show Module Name / Identifier and Versioning Information	N/A	Module name, version
CO	Show Status	N/A	Approved service status
CO	Symmetric Encryption / Decryption	Key, IC, IV, data	Encrypted / decrypted data, status
CO	Zeroisation	N/A	Status

4.3 Identification and Authentication

The module does not support authentication; all services are assigned to the CO role. There is only one role (CO), which does not require authentication to assume. The module is a level 1 software module that does not support authentication mechanisms.

4.4 Services

4.4.1 Approved Services

The module supports the following Approved Services described in Table 6.

Table 6: Approved Services

Role	Service	Description	Security Functions Used	Access Rights to Keys / SSPs	Approved Service Indicator
CO	Control PAA	Control the use of PAA	N/A	N/A	Service completion
CO	Message Digest	Generate message digest	SHA-1 or SHA-256 or SHA-512	N/A	Status indicating success or failure
CO	Perform Conditional Self-Tests	Perform KAT on algorithms	AES CBC & AES CTR & AES ECB & CTR_DRBG & Hash_DRBG & SHA	N/A	Status indicating success or failure
CO	Perform Pre-Operational Self-Tests	Perform library integrity check and KAT on algorithms	AES CBC & AES CTR & AES ECB & CTR_DRBG & Hash_DRBG & SHA	N/A	Status indicating success or failure when commanded or application abort on startup
CO	Random Number Generation	Generate random numbers	CTR_DRBG or Hash_DRBG	CTR_DRBG V: (G, E) CTR_DRBG Key: (G, E) Hash_DRBG V, C: (G, E) DRBG Entropy (W, E, Z)	Status indicating success or failure
CO	Show Module Name / Identifier and Versioning Information	Retrieve information about the Module	N/A	N/A	Service completion
CO	Show Status	Retrieve the approved service status	N/A	N/A	Non-zero if approved or zero if not approved
CO	Symmetric Encryption / Decryption	Perform symmetric encryption and decryption	AES CBC or AES CTR or AES ECB	AES Key: W, E	Status indicating success or failure

CO	Zeroisation	Zeroise SSP	AES CBC & AES CTR & AES ECB & CTR_DRBG & Hash_DRBG	AES Key: Z DRBG Entropy: Z CTR_DRBG V: Z CTR_DRBG Key: Z Hash_DRBG V, C: Z	Status indicating success or failure for DRBGs; service completion for ciphers

SSP access rights are defined as follows:

G = Generate: The module generates or derives the SSP.

R = Read: The SSP is read from the module (e.g., the SSP is output). Read access is N/A for the Module.

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 = Zeroise: The module zeroises the SSP.

5 Software / Firmware Security

The executable code is provided as a dynamic library (ecl) along with an associated set of C header files used by consuming applications to interface to the library.

The Eclypses Cryptographic Library Module consists of the following files, depending on operational environment (refer to Table 2 for OE numbers in brackets):

- libecl.so, libecl.so.sha256
  - Google Android 9 (32-bit) [7]
  - Google Android 9 (64-bit) [8, 9]
  - OpenWRT Linux 19.07 (32-bit) [12]
  - Raspbian Linux 5.10 (32-bit) [13]
  - Ubuntu Linux 20.04 (64-bit) [14, 15]
- libecl.dylib, libecl.dylib.sha256
  - Apple iOS 13.5 (64-bit) [1, 2]
  - Apple macOS Big Sur 11.3 (64-bit) [3, 4, 5, 6]
- ecl.dll, ecl.dll.sha256
  - Microsoft Windows 10 Pro 20H2 (64-bit) [10, 11]

The Perform Pre-Operational Self-Test service runs a self-test on the SHA-256 algorithm to verify it is operating correctly. It then performs a SHA-256 hash calculation on the copy of the dynamic library (ecl) currently being executed (found via dynamic linker functions on most OEs; found in the APK on Android). This hash digest is compared against the hash digest stored in the hash file, which was calculated when the dynamic library (ecl) was built. Finally, the Perform Conditional Self-Tests service is performed. The Pre-Operational Self-Test, which includes the Conditional Self-Tests in addition to the library integrity test, succeeds only if the calculated hash digest of the library in use matches the stored hash from when the library was built and the Conditional Self-Tests succeed.

Pre-Operational Self-Test, which includes the Conditional Self-Tests in addition to the library integrity test, runs automatically when the consuming application starts (when the library is loaded on Android) due to the use of default entry points. The Pre-Operational Self-Test checks the dynamic library to ensure integrity. If the Pre-Operational Self-Test fails for any reason, a message is printed to standard error (the assert logcat on Android) and the C abort() function is called to force termination of the application. If the abort() is ignored (on systems that can ignore it), there is still a flag set which prevents any of the algorithms from passing their Conditional Self-Tests, and since those cannot pass, all algorithms inhibit output.

The consuming application user can initiate the Pre-Operational Self-Test, which includes the Conditional Self-Tests in addition to the library integrity test, at any time using the Perform Pre- Operational Self-Tests service.

6 Operational Environment

The module is a software library and the physical embodiment is provided by the multiple-chip standalone general purpose computing platform on which it is installed. All operational environments for the module are classified as modifiable. The tested operating systems and tested platforms are detailed in Table 2.

All operational environments the Module is certified for support default entry points, which are required for the Pre-Operational Self-Test. Modifying the operational environment to disable the default entry point renders the Module in an error state so it cannot be used. The Module has no other interaction with the operational environment, so all requirements for Level 1 are satisfied by all operational environments the Module is certified for.

There are no security rules, settings, or restrictions on the configuration of the operational environment imposed by the Module. The Module has no interaction with the operational environment beyond probing the hardware for PAA in certain cases and reading the library integrity hash file.

Consuming applications may define security rules, settings, or restrictions on the configuration of the operational environment.

All operational environments the Module is certified for segregate processes. Each process is logically separated from all other processes by the operating system and hardware. The Module functions entirely within a single process. The operational environment enforces process separation by the use of virtual memory. The virtual memory system uses the memory management unit (MMU) hardware to prevent processes from accessing the memory image of another process.

7 Physical Security

Not applicable because the Module is purely software.

8 Non-Invasive Security

The Module does not implement non-invasive mitigation techniques.

9 Sensitive Security Parameter Management

The Module does not permanently store SSPs. The only SSPs are the current state of any instance of any parts of the Module. The SSPs the Module creates are only kept in memory state structures. The Module provides zeroisation services to ensure all SSPs are zeroised when no longer needed.

The SSPs created by the Module are for Module use only to perform cryptographic services.

9.1 SSPs

Table 7: SSPs

Key / SSP / Name / Type	Strength	Security Function & Cert. Number	Generation	Import / Export	Establishment	Storage	Zeroisation	Use & Related Keys
AES Key	128 bits, 192 bits, 256 bits	AES CBC (A1406, A1422) AES CTR (A1407, A1423) AES ECB (A1408, A1424)	N/A	Import: Plaintext; Export: N/A Method: Electronic	Provided by consuming application	Plaintext in RAM (temporarily)	Zeroisation service	Key used to encrypt or decrypt data
DRBG Entropy	≥112 bits	CTR_DRBG (A1414, A1425) Hash_DRBG (A1412, A1419, A1427, A1432)	N/A	Import: Plaintext; Export: N/A Method: Electronic	Provided by consuming application	Plaintext in RAM (temporarily)	Zeroised implicitly in instantiation immediately upon use	Used to instantiate the DRBG
CTR_DRBG V	128 bits	CTR_DRBG (A1414, A1425)	N/A	Import: N/A; Export: N/A	Derived from entropy, nonce, and personalization string	Plaintext in RAM (temporarily)	Zeroisation service	State of the DRBG
CTR_DRBG Key	128 bits, 192 bits, 256 bits	CTR_DRBG (A1414, A1425)	N/A	Import: N/A; Export: N/A	Derived from entropy, nonce, and personalization string	Plaintext in RAM (temporarily)	Zeroisation service	State of the DRBG
Hash_DRBG V, C	440 bits, 888 bits	Hash_DRBG (A1412, A1419, A1427, A1432)	N/A	Import: N/A; Export: N/A	Derived from entropy, nonce, and personalization string	Plaintext in RAM (temporarily)	Zeroisation service	State of the DRBG

RBG state information and intermediate generated values are CSPs.

As noted in the table above, any entropy input is defined as a CSP. It is zeroised immediately after use.

CSPs may be provided in plaintext form because they will be maintained within the operational environment. The Module does not store or transmit any CSP entered.

The DRBG entropy is loaded when needed during instantiation. The minimum number of bits of entropy required and the minimum/maximum entropy bitstring length are specified in the load request and the minimum number of bits of entropy is believed to have been provided if the load returns a successful status (refer to Table 4 for additional detail). Additionally, if the length of the bitstring loaded is less than the minimum bits of entropy required or the status is not success, the instantiation fails under the assumption that insufficient entropy was available; otherwise, the number of bits loaded is believed to contain the minimum required bits of entropy for the security strength desired. The minimum number of bits of entropy provided is equal to the security strength for each Hash DRBG and each CTR_DRBG with derivation function (i.e., 128, 192, or 256 bits), and equal to the sum of the block size and key length for each CTR_DRBG without derivation function (i.e., 256, 320, or 384 bits). FIPS caveat: no assurance of minimum strength of generated random strings.

The DRBG SSPs (V, C, Key) are derived from SSPs entered into the module in an approved manner defined in the DRBG algorithm.

9.2 PSPs

The Module does not have any PSPs.

9.3 RBG

The Module does not incorporate any non-deterministic random number generators.

Entropy provided to the (NIST SP 800-90A) DRBGs must be provided by an approved random source (NIST SP 800-90B).

9.4 Zeroisation

The Module provides zeroisation services that will zeroise all SSPs on demand. The zeroisation is optimized to run in the least number of processor instructions to complete as quickly as possible. The zeroisation is initiated by the library user and the indication of completion is the function call returning.

10 Self-Tests

All cryptographic operations are inhibited during error states and while self-tests are being performed. Any attempted cryptographic operation during an error state or while self-tests are being performed will result in an error status (i.e., ecl_status_output_inhibited) and/or zeroised output.

10.1 Pre-Operational Self-Tests

All Pre-Operational Self-Tests test the non-PAA implementation. If PAA is available, the PAA implementation is also tested.

Table 8: Pre-Operational Self-Tests

Tested Function	Self-Test	Conditions For Performing Test	Error Indicator
Library integrity	SHA-256 digest of the dynamic library file compared to the known digest computed at build time	Library load	Failure to load ***ecl_g_fips_error*** set to true

10.2 Conditional Self-Tests

All Conditional Self-Tests test the non-PAA implementation. If PAA is available, the PAA implementation is also tested.

Table 9: Conditional Self-Tests

Tested Function	Self-Test	Conditions For Performing Test	Error Indicator
AES CBC (Certs A1406, A1422)	AES-128-CBC Encrypt KAT AES-128-CBC Decrypt KAT AES-192-CBC Encrypt KAT AES-192-CBC Decrypt KAT AES-256-CBC Encrypt KAT AES-256-CBC Decrypt KAT	On demand or when changing the PAA option	Status output: ***ecl_status_cipher_test_failed***
AES CTR (Certs A1407, A1423)	AES-128-CTR Encrypt KAT AES-128-CTR Decrypt KAT AES-192-CTR Encrypt KAT AES-192-CTR Decrypt KAT AES-256-CTR Encrypt KAT AES-256-CTR Decrypt KAT	On demand or when changing the PAA option	Status output: ***ecl_status_cipher_test_failed***

AES ECB (Certs A1408, A1424)	AES-128-ECB Encrypt KAT AES-128-ECB Decrypt KAT AES-192-ECB Encrypt KAT AES-192-ECB Decrypt KAT AES-256-ECB Encrypt KAT AES-256-ECB Decrypt KAT	On demand or when changing the PAA option	Status output: ***ecl_status_cipher_test_failed***
CTR_DRBG NODF[8] (Certs A1414, A1425)	CTR-AES-128-NODF Instantiate KAT CTR-AES-128-NODF Generate KAT CTR-AES-192-NODF Instantiate KAT CTR-AES-192-NODF Generate KAT CTR-AES-256-NODF Instantiate KAT CTR-AES-256-NODF Generate KAT	On demand or when changing the PAA option	Status output: ***ecl_status_drbg_catastrophic***
CTR_DRBG DF[9] (Certs A1413, A1426)	CTR-AES-128-DF Instantiate KAT CTR-AES-128-DF Generate KAT CTR-AES-192-DF Instantiate KAT CTR-AES-192-DF Generate KAT CTR-AES-256-DF Instantiate KAT CTR-AES-256-DF Generate KAT	On demand or when changing the PAA option	Status output: ***ecl_status_drbg_catastrophic***
Hash_DRBG[10] (Certs A1412, A1419, A1427, A1432)	HASH-SHA-1 Instantiate KAT HASH-SHA-1 Generate KAT HASH-SHA-256 Instantiate KAT HASH-SHA-256 Generate KAT HASH-SHA-512 Instantiate KAT HASH-SHA-512 Generate KAT	On demand or when changing the PAA option	Status output: ***ecl_status_drbg_catastrophic***
SHA-1 (Certs A1409, A1428)	SHA-1 Digest KAT	On demand or when changing the PAA option	Status output: ***ecl_status_hash_test_failed***
SHA-256 (Certs A1410, A1418)	SHA-256 Digest KAT	On demand, when changing the PAA option, or prior to the library integrity check	Status output: ***ecl_status_hash_test_failed***
SHA-512 (Certs A1411, A1433)	SHA-512 Digest KAT	On demand or when changing the PAA option	Status output: ***ecl_status_hash_test_failed***

[8] DRBG reseeding is not supported by the module.
[9] DRBG reseeding is not supported by the module.
[10] DRBG reseeding is not supported by the module.

To resolve errors, restart the consuming application or command the Perform Pre-Operational Self- Tests service. The Module can be used if the Perform Pre-Operational Self-Tests service returns success.

10.3 Periodic Self-Tests

The Module's consuming application can use any of the Perform Pre-Operational Self-Tests or Perform Conditional Self-Test services at any time to perform a periodic self-test. If the on demand self-tests fail, the module will return ECL_FALSE.

11 Life-Cycle Assurance

Module code is stored and managed within private and secure configuration management systems, GIT and Azure DevOps. The Module documentation is stored in the same repository to maintain the history synchronized with the source code.

Module code utilizes versioning along with release notes providing guidance in the use of each iteration of Module's code. The Module has a semantic version number. The Git configuration management system assigned a message digest to each commit which uniquely identifies the version.

Configuration management systems retaining Module code are periodically backed up using a solution that allows both full and incremental restoration.

A comprehensive developer guide documenting the entire API is provided. The developer guide provides all Administrator guidance. The developer guide is versioned and managed within the same secure configuration management systems.

The configuration management system is protected with user authentication measures to prevent unauthorized access. All changes can be made only by authorized individuals. An automated system is in place to notify other users when any change has been made to prevent unknown changes from being made. All changes are made on development branches in Git and are reviewed by separate people before releasing to production.

The only maintenance required is to install new versions of the Module when they are made available to fix discrepancies or add features. The update procedure is the same as the installation procedure, with the new Module file(s) replacing the existing ones.

Any new version of the Module is outside the scope of this validation and requires a separate FIPS 140-3 validation.

11.1 Installation

11.1.1 Linux, macOS, Windows

The Crypto Officer shall install the dynamic library and library integrity hash file in a standard location that is protected from modification by Users. The dynamic library path shall be restricted to protected paths to avoid library interposition. The library install location must be in a path that is MAX_PATH or PATH_MAX (depending on operating system) characters or less. The dynamic library and library integrity hash file must be collocated and shall be protected against modification by Users.

11.1.2 iOS

The Crypto Officer shall configure their iOS app project according to instructions detailed in the SDK. This will ensure the proper embedded locations of the library and integrity hash file in the IPA.

11.1.3 Android

The Crypto Officer shall configure their Android app project according to instructions detailed in the SDK. This will ensure the proper embedded locations of the library and integrity hash file in the APK.

11.1.4 General

There is no authentication component to the Module, so no further setup is required. No zeroisation is required during installation.

11.2 Security Rules

All Pre-Operational Self-Tests and Conditional Self-Tests are performed at power-on. Where PAA is available, the PAA and non-PAA versions are tested as part of the Perform Pre-Operational Self-Tests service, which includes the Perform Conditional Self-Tests service in addition to the library integrity test. To set the PAA option for use after the Pre-Operational Self-Tests, the Module requires the conditional self-test services to be commanded again after setting the desired PAA options.

All algorithms have self-tests to verify the algorithm is working correctly. The Perform Pre-Operational Self-Test service commands all algorithm self-tests after first verifying library integrity. A library user can command the Perform Conditional Self-Test service to run at any time. A library user can command the Perform Pre-Operational Self-Test service, which includes the Perform Conditional Self-Tests service in addition to the library integrity check, to run again at any time.

12 Mitigation Of Other Attacks

This module is not designed to mitigate other attacks beyond the scope of FIPS 140-3 requirements.

13 References

Table 10: References

Reference Number	Reference Title	Publishing Entity	Publication Date
[1]	FIPS 180-4	NIST	08/2015
[2]	FIPS 197	NIST	11/26/2001
[3]	NIST SP 800-38A	NIST	12/2001
[4]	NIST SP 800-90A	NIST	06/2015

14 Abbreviations And Definitions

Table 11: Abbreviations and Definitions

Term	Definition
AES	Advanced Encryption Standard
CAST	Cryptographic Algorithm Self-Test
CBC	Cipher Block Chaining mode of operation
CSP	Critical Security Parameters
CTR	Counter mode of operation
DF	Derivation Function
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book mode of operation
IC	Initial Count
IV	Initialization Vector
KAT	Known Answer Test
NODF	No Derivation Function
PAA	Processor Algorithm Acceleration
POST	Pre-Operational Self-Test
PSP	Public Security Parameters
SDK	Software Development Kit
SHA	Secure Hash Algorithm
SSP	Sensitive Security Parameters (CSP + PSP)