Allegro Software Development Corporation

Allegro Cryptographic Engine

FIPS 140-3 Non-Proprietary Security Policy

Document Revision 1.1
September 2024

Table of Contents

List of Tables

List of Figures

1 General

1.1 Overview

This document is a non-proprietary cryptographic module security policy for the Allegro Cryptographic Engine (Software Version 6.50) from Allegro Software Development Corporation. This security policy 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.

1.2 Security Levels

Section	Title	Security Level
1	General	1
2	Cryptographic module specification	1
3	Cryptographic module interfaces	1
4	Roles, services, and authentication	1
5	Software/Firmware security	1
6	Operational environment	1
7	Physical security	N/A
8	Non-invasive security	N/A
9	Sensitive security parameter management	1
10	Self-tests	1
11	Life-cycle assurance	1
12	Mitigation of other attacks	N/A
	Overall Level	1

Table 1: Security Levels

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

The Allegro Cryptographic Engine (also informally referred to as “ACE,” and in this security policy as “the module”) is a software cryptographic module that runs on a general-purpose computer (GPC). It provides FIPS 140-3 approved cryptography that can be used by calling applications via a C language Application Programming Interface (API). The module meets the overall requirements applicable to a multi-chip stand-alone embodiment at FIPS 140-3, Level 1. The module is a shared cryptographic library providing symmetric and asymmetric encryption and decryption, message digest, message authentication, random number generation, key generation, digital signature generation and verification, and other cryptographic functionality.

As a software cryptographic module that executes on a general-purpose computer, the module depends upon the physical characteristics of the host platform. The module’s physical perimeter is defined by the enclosure around the host system on which it executes.

The logical interface of the module is its Application Programming Interface, which a calling application must utilize to invoke the cryptographic services of the module, pass input data to the module and receive output data and status from the module.

The module is packaged as a shared object for Linux 5.15 (Mint 21) and Windows 11 Pro. The module also includes a data file that is used for verifying the integrity of the module. The module has been validated on Linux 5.15 (Mint 21) and Windows 11 Pro.

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

Module Type: Software

Module Embodiment: MultiChipStand

Cryptographic Boundary:

The module's cryptographic boundary is comprised of a single binary:

- On Linux 5.15 (Mint 21), the binary is **Acelib.so** with the associated digest in **AceLib.dat**.
- On Windows 11 Pro, the binary is **AceDll.dll** with the associated digest in **AceDll.dll.dat**.

Tested Operational Environment's Physical Perimeter (TOEPP)

Figure 1 shows a block diagram of the module executing in memory, and its interactions with surrounding software components, as well as the module's cryptographic boundary. The module supports an Application Programming Interface (API) which provides logical interfaces between the calling application and the module's services.

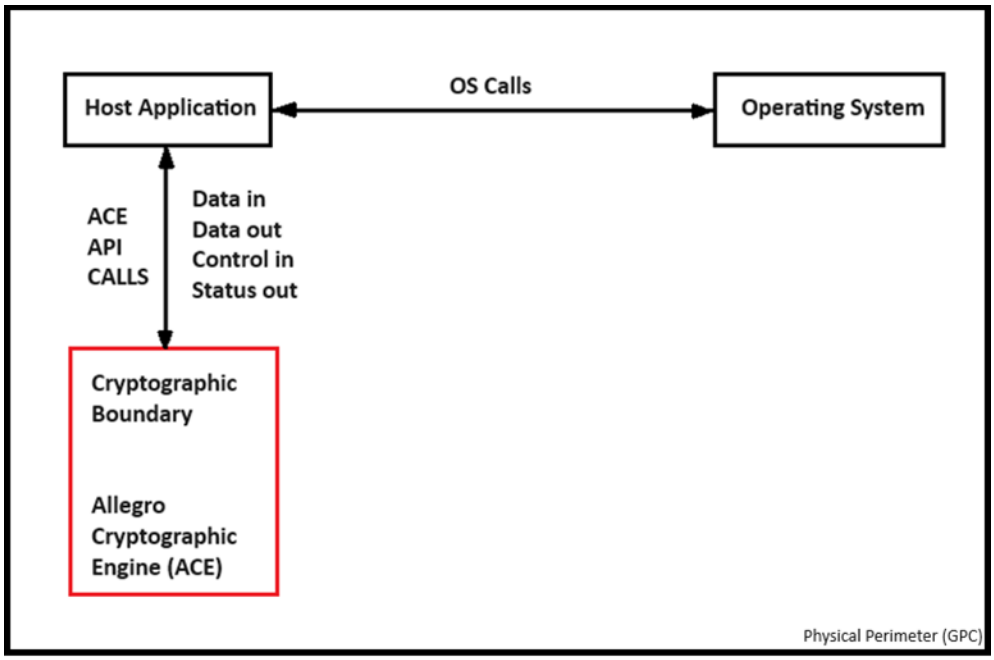

Figure 1: Block Diagram

2.2 Tested and Vendor Affirmed Module Version and Identification

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

Package or File Name	Software/ Firmware Version	Features	Integrity Test
Acelib.so (Linux 5.15) (Mint 21) (PAA Enabled)	6.50	PAA Enabled Binary	HMAC-SHA2-256 (AceLib.dat)
AceDll.dll (Windows 11 Pro) (PAA Enabled)	6.50	PAA Enabled Binary	HMAC-SHA2-256 (AceDll.dll.dat)

Package or File Name	Software/ Firmware Version	Features	Integrity Test
Acelib.so (Linux 5.15) (Mint 21) (PAA Disabled)	6.50	PAA Disabled Binary	HMAC-SHA2-256 (AceLib.dat)
AceDll.dll (Windows 11 Pro) (PAA Disabled)	6.50	PAA Disabled Binary	HMAC-SHA2-256 (AceDll.dll.dat)

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

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Linux 5.15 (Mint 21)	Intel NUC	Intel® Core™ i7-1260P	No	N/A	6.50
Linux 5.15 (Mint 21)	Intel NUC	Intel® Core™ i7-1260P	Yes	N/A	6.50
Windows 11 Pro	Intel NUC	Intel® Core™ i7-1260P	No	N/A	6.50
Windows 11 Pro	Intel NUC	Intel® Core™ i7-1260P	Yes	N/A	6.50

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

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

2.3 Excluded Components

There are no components excluded from the module.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved Mode	Mode of operation where only approved security functions and services can be utilized	Approved	Pass

Table 4: Modes List and Description

The module supports an approved mode of operation only.

2.5 Algorithms

Approved Algorithms:

Cipher

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A3332	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A3332	Key Length - 128, 192, 256	SP 800-38C
AES-CFB1	A3332	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB128	A3332	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A3332	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A

Algorithm	CAVP Cert	Properties	Reference
AES-CTR	A3332	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A3332	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-FF1	A3332	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38G
AES-GCM	A3332	Direction - Decrypt, Encrypt IV Generation - Internal IV Generation Mode - 8.2.1 Key Length - 128, 192, 256	SP 800-38D
AES-OFB	A3332	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-XTS Testing Revision 2.0	A3332	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38E

Table 5: Approved Algorithms - Cipher

Message Authentication

Algorithm	CAVP Cert	Properties	Reference
AES-CMAC	A3332	Direction - Generation, Verification Key Length - 128, 192, 256	SP 800-38B
AES-GMAC	A3332	Direction - Decrypt, Encrypt IV Generation - Internal IV Generation Mode - 8.2.1 Key Length - 128, 192, 256	SP 800-38D
HMAC-SHA-1	A3332	Key Length - Key Length: 256-448 Increment 8	FIPS 198-1
HMAC-SHA2-224	A3332	Key Length - Key Length: 256-448 Increment 8	FIPS 198-1
HMAC-SHA2-256	A3332	Key Length - Key Length: 256-448 Increment 8	FIPS 198-1
HMAC-SHA2-384	A3332	Key Length - Key Length: 256-448 Increment 8	FIPS 198-1
HMAC-SHA2-512	A3332	Key Length - Key Length: 256-448 Increment 8	FIPS 198-1
HMAC-SHA3-224	A3332	Key Length - Key Length: 256-448 Increment 8	FIPS 198-1
HMAC-SHA3-256	A3332	Key Length - Key Length: 256-448 Increment 8	FIPS 198-1
HMAC-SHA3-384	A3332	Key Length - Key Length: 256-448 Increment 8	FIPS 198-1
HMAC-SHA3-512	A3332	Key Length - Key Length: 256-448 Increment 8	FIPS 198-1

Table 6: Approved Algorithms - Message Authentication

Symmetric Key Wrap

Algorithm	CAVP Cert	Properties	Reference
AES-KW	A3332	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
AES-KWP	A3332	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F

Table 7: Approved Algorithms - Symmetric Key Wrap

Asymmetric Key Generation

Algorithm	CAVP Cert	Properties	Reference
ECDSA KeyGen (FIPS186-4)	A3332	Curve - P-224, P-256, P-384, P-521 Secret Generation Mode - Testing Candidates	FIPS 186-4
RSA KeyGen (FIPS186-4)	A3332	Key Generation Mode - B.3.6 Modulo - 2048, 3072, 4096 Primality Tests - Table C.3 Private Key Format - Standard	FIPS 186-4

Table 8: Approved Algorithms - Asymmetric Key Generation

Asymmetric Key Verification

Algorithm	CAVP Cert	Properties	Reference
ECDSA KeyVer (FIPS186-4)	A3332	Curve - P-192, P-224, P-256, P-384, P-521	FIPS 186-4

Table 9: Approved Algorithms - Asymmetric Key Verification

Asymmetric Signature Generation

Algorithm	CAVP Cert	Properties	Reference
ECDSA SigGen (FIPS186-4)	A3332	Curve - P-224, P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
RSA SigGen (FIPS186-4)	A3332	Signature Type - ANSI X9.31, PKCSPSS Modulo - 2048, 3072, 4096	FIPS 186-4

Table 10: Approved Algorithms - Asymmetric Signature Generation

Asymmetric Signature Verification

Algorithm	CAVP Cert	Properties	Reference
ECDSA SigVer (FIPS186-4)	A3332	Curve - P-224, P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
RSA SigVer (FIPS186-4)	A3332	Signature Type - ANSI X9.31, PKCSPSS Modulo - 2048, 3072, 4096	FIPS 186-4

Table 11: Approved Algorithms - Asymmetric Signature Verification

Random Number Generation

Algorithm	CAVP Cert	Properties	Reference
Hash DRBG	A3332	Mode - SHA2-256, SHA2-512	SP 800-90A Rev. 1

Table 12: Approved Algorithms - Random Number Generation

Shared Secret Computation

Algorithm	CAVP Cert	Properties	Reference
KAS-ECC-SSC Sp800-56Ar3	A3332	Domain Parameter Generation Methods - P-224, P-256, P-384, P-521 Scheme - ephemeralUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3
KAS-FFC-SSC Sp800-56Ar3	A3332	Domain Parameter Generation Methods - MODP-2048, MODP-3072	SP 800-56A Rev. 3

Table 13: Approved Algorithms - Shared Secret Computation

Key Derivation

Algorithm	CAVP Cert	Properties	Reference
KDA HKDF Sp800-56Cr1	A3332	Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-2048 Increment 8 HMAC Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512	SP 800-56C Rev. 2
KDF SSH (CVL)	A3332	Cipher - AES-128, AES-192, AES-256 Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1
PBKDF	A3332	Iteration Count - Iteration Count: 1000-100000 Increment 1 Password Length - Password Length: 8-128 Increment 1	SP 800-132

Algorithm	CAVP Cert	Properties	Reference
TLS v1.2 KDF RFC7627 (CVL)	A3332	Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1
TLS v1.3 KDF (CVL)	A3332	HMAC Algorithm - SHA2-256, SHA2-384 KDF Running Modes - PSK-DHE	SP 800-135 Rev. 1

Table 14: Approved Algorithms - Key Derivation

Hash Function

Algorithm	CAVP Cert	Properties	Reference
SHA-1	A3332	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-224	A3332	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-256	A3332	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-384	A3332	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-512	A3332	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA3-224	A3332	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-256	A3332	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-384	A3332	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-512	A3332	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHAKE-128	A3332	Output Length - Output Length: 16-65536 Increment 8	FIPS 202
SHAKE-256	A3332	Output Length - Output Length: 16-65536 Increment 8	FIPS 202

Table 15: Approved Algorithms - Hash Function

Safe Primes Generation

Algorithm	CAVP Cert	Properties	Reference
Safe Primes Key Generation	A3332	Safe Prime Groups - modp-2048, modp-3072	SP 800-56A Rev. 3

Table 16: Approved Algorithms - Safe Primes Generation

Safe Primes Verification

Algorithm	CAVP Cert	Properties	Reference
Safe Primes Key Verification	A3332	Safe Prime Groups - modp-2048, modp-3072	SP 800-56A Rev. 3

Table 17: Approved Algorithms - Safe Primes Verification

The module's approved algorithms are specified above. There are some algorithm modes that were tested but not implemented by the module. Only the algorithms, modes, and key sizes that are implemented by the module are shown in this table.

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
CKG Section 4	Key Type:Symmetric	N/A	NIST SP 800-133 Rev. 2 (Section 4)
CKG Section 4	Key Type:Seed for Asymmetric Key	N/A	NIST SP 800-133 Rev. 2 (Section 4)
CKG Section 6.1	Key Type:Symmetric	N/A	NIST SP 800-133 Rev. 2 (Section 6.1)

Table 18: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

Name	Properties	Implementation	Reference
Key Unwrap	Key Type:Symmetric	N/A	IG D.G

Table 19: Non-Approved, Allowed Algorithms

Non-Approved, Allowed Algorithms with No Security Claimed:

Name	Caveat	Use and Function
MD5	Allowed in the approved mode with no security claimed	Used for TLS 1.2 interoperability

Table 20: Non-Approved, Allowed Algorithms with No Security Claimed

Caveats:

- The module includes vendor-affirmed Component Key Generation (CKG) per IG D.H and NIST SP 800-133rev2. The CKG uses the method described in Sections 4 and 6.1 of SP 800-133rev2. The module generates symmetric keys and seeds for asymmetric keys, from the direct output of the DRBG.
- The module implements MD5 for use with TLS communications, which is allowed in the Approved mode of operation.
- The module provides key derivation functions for use in TLS and SSHv2. No parts of the TLS or SSHv2 protocols, other than the KDF, have been tested by the CAVP and CMVP.

Non-Approved, Not Allowed Algorithms:

N/A for this module.

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
DRBG	DRBG	Random Bit Generation		Hash DRBG
Message Digest	SHA	Create Message Digest		SHA-1 SHA2-224 SHA2-256 SHA2-384 SHA2-512 SHA3-224 SHA3-256 SHA3-384 SHA3-512 SHAKE-128 SHAKE-256
Generate Digital Signature	DigSig-SigGen	Create Digital Signatures		ECDSA SigGen (FIPS186-4) RSA SigGen (FIPS186-4)
Verify Digital Signature	DigSig-SigVer	Verify Digital Signature		ECDSA SigVer (FIPS186-4) RSA SigVer (FIPS186-4)
Generate Symmetric Keys	CKG DRBG	Generate Symmetric Keys		Hash DRBG CKG Section 4 Key Type: Symmetric CKG Section 6.1 Key Type: Symmetric

Name	Type	Description	Properties	Algorithms
Generate Asymmetric Keys	AsymKeyPair-KeyGen CKG	Generation of Asymmetric Keys		ECDSA KeyGen (FIPS186-4) RSA KeyGen (FIPS186-4) Hash DRBG CKG Section 4 Key Type: Seed for Asymmetric Key
Shared Secret Computation (KAS-FFC-SSC)	AsymKeyPair-SafePri KAS-SSC	Shared Secret Computation (KAS-FFC-SSC)	Shared Secret Computation:Provides between 112 and 128 bits of encryption strength	KAS-FFC-SSC Sp800-56Ar3 Safe Primes Key Generation Safe Primes Key Verification
Shared Secret Computation (KAS-ECC-SSC)	AsymKeyPair-DomPar AsymKeyPair-KeyGen AsymKeyPair-PubKeyVal DRBG KAS-SSC	NIST SP 800-56Ar3 shared secret computation (KAS-ECC-SSC)	Shared Secret Computation:Provides between 128 and 256 bits of encryption strength	KAS-ECC-SSC Sp800-56Ar3 ECDSA KeyGen (FIPS186-4) ECDSA SigGen (FIPS186-4) ECDSA SigVer (FIPS186-4) Hash DRBG
Derive Key (HKDF)	KAS-56CKDF SHA	Key Derivation		KDA HKDF Sp800-56Cr1 HMAC-SHA2-224 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512 HMAC-SHA3-224 HMAC-SHA3-256 HMAC-SHA3-384 HMAC-SHA3-512
Derive Key (TLS 1.2)	KAS-135KDF SHA	Key Derivation for TLS 1.2 RFC 7627		TLS v1.2 KDF RFC7627 SHA2-256 SHA2-384 SHA2-512
Derive Key (TLS 1.3)	KAS-135KDF SHA	Key Derivation for TLS 1.3		TLS v1.3 KDF SHA2-256 SHA2-384
Derive Key (SSH)	KAS-135KDF SHA	Key Derivation for SSH		KDF SSH SHA-1 SHA2-224 SHA2-256 SHA2-384 SHA2-512
Derive Key (PBKDF)	MAC PBKDF	Password-Based Key Derivation		PBKDF HMAC-SHA-1
Message Authentication	MAC	Message Authentication Algorithms		HMAC-SHA-1 HMAC-SHA2-224 HMAC-SHA2-256

Name	Type	Description	Properties	Algorithms
				HMAC-SHA2-384 HMAC-SHA2-512 HMAC-SHA3-224 HMAC-SHA3-256 HMAC-SHA3-384 HMAC-SHA3-512 AES-CMAC AES-GMAC Hash DRBG CKG Section 4 Key Type: Symmetric CKG Section 6.1 Key Type: Symmetric
Symmetric Cipher	BC-Auth BC-UnAuth	Encryption & Decryption		AES-CBC AES-CCM AES-CFB1 AES-CFB128 AES-CFB8 AES-ECB AES-FF1 AES-GCM AES-KW AES-KWP AES-OFB AES-XTS Testing Revision 2.0 AES-CTR
Key Wrapping	KTS-Wrap	Key Wrapping Method (SP 800-38F) (IG D.G)	KTS:Key establishment methodology provides between 128 and 256 bits of encryption strength	AES-KW AES-KWP
Verify Asymmetric Keys	AsymKeyPair-KeyVer	Verify Asymmetric Keys		ECDSA KeyVer (FIPS186-4)
CKG Section 4	CKG	NIST SP 800-133 Rev. 2 (Section 4)		CKG Section 4
CKG Section 6.1	CKG	NIST SP 800-133 Rev. 2 (Section 6.1)		CKG Section 6.1

Table 21: Security Function Implementations

2.7 Algorithm Specific Information

SHA-3 & SHAKE (IG C.C)

SHA-3 and SHAKE were tested and validated on all of the module's operating environments.

RSA (IG C.F)

- Modulus lengths supported by the module for RSA signature generation are 2048, 3072, and 4096 bits.
- The module supports the number of Miller-Rabin tests specified in Table C.3 of FIPS 186-4.
- The RSA signature algorithm implementations are tested.
- For signature verification, the modulus size is at least 2048.

AES-GCM (IG C.H)

An AES-GCM key may either be generated internally or provided by application code to the cryptographic module. The IV for AES-GCM encryption shall not be generated outside the module. The probability that the authenticated encryption function will be invoked with the same initialization vector and the same key on two or more distinct sets of input data shall be no greater than 2^{-32}. If the module's power is lost and then restored, a new key for use with AES-GCM encryption/decryption must be established.

Per IG C.H Option 2, the module generates 96-bit GCM IVs randomly as specified in SP800-38D section 8.2.2 using an approved DRBG (Cert. #A3332), that is internal to the module's boundary. The Module does not implement the TLS protocol itself, however, it provides the cryptographic functions required for implementing the protocol. AES-GCM encryption is used in the context of the TLS protocol versions 1.2 and 1.3 (per Scenario 1 and Scenario 5 in FIPS 140-3 C.H respectively). For TLS v1.2, the mechanism for IV generation is compliant with RFC 5288. The counter portion of the IV is strictly increasing. When the IV exhausts the maximum number of possible values for a given session key, this results in a failure in encryption and a handshake to establish a new encryption key will be required. It is the responsibility of the user of the module i.e., the first party, client or server, to encounter this condition, to trigger this handshake in accordance with RFC 5246. For TLS v1.3, the mechanism for IV generation is compliant with RFC 8446.

The TLS 1.2 AES-GCM cipher suites from Section 3.3.1 of SP 800-52 Rev2 supported by the module are:

- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xC0, 0x2B);
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA384 (0xC0, 0x2C);
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xC0, 0x2F);
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xC0, 0x30);
- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x00, 0x9E);
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x00, 0x9F);
- TLS_DHE_DSS_WITH_AES_128_GCM_SHA256 (0x00, 0xA2); and
- TLS_DHE_DSS_WITH_AES_256_GCM_SHA384 (0x00, 0xA3).

AES-XTS (IG C.I)

AES-XTS shall only be used in storage applications. The module implements the Key_1 ≠ Key_2 check prior to the first use of the algorithm.

AES-FF1 (IG C.J)

The lengths of the following parameters from SP 800-38G:

- radix = 2 … 96;
- minlen = 6;
- maxlen = 2048; and
- maxTlen = 21.

SP 800-107 Requirements (IG C.L)

The Allegro Cryptographic Engine is a software module toolkit. The operator utilizing the algorithms and schemes specified in IG C.L shall operate in accordance with the guidance.

PBKDF2 (IG D.N)

The Allegro Cryptographic Engine requires the password to be at least ten characters in length, the iteration count at least 1000, the salt at least 128 bits in length, and that the master key output from the PBKDF2 is at least 112 bits in length. Master keys may be used as Device Protection Keys (Option 1(a) from Section 5.4 of NIST SP 800-132). Alternately, they may be used with a key derivation function to produce a Device Protection Key (Option 1(b) from Section 5.4 of NIST SP 800-132).

Passwords passed to the PBKDF2 implemented shall have a length of at least 10 characters and shall consist of upper- and lower-case letters and numbers (52 letters) and digits (0-9) as well as characters from the set ~!@#$%^&*. There are 71 different characters that can be used, in any order. The probability of guessing this password at random is 71^{10} = 1: 3.3 * 10^{18}. This provides a password search space of more than 60 bits. The length of the random salt used in PBKDF2 must be at least 128 bits. The iteration count used in PBKDF2 must be at least 1000 and should be as large as is tolerable by the calling application. The length of the master key generated by PBKDF2 must be at least 112 bits. The calling application may use the master key, the Data Protection Key, or it may derive the Data Protection Key from the master key using a key derivation function. The Data Protection Key shall be used for storage purposes only and shall use only approved encryption algorithms.

2.8 RBG and Entropy

The entropy for seeding the SP 800-90Ar1 DRBG is determined by the user of the module, which is outside of the module's cryptographic boundary. To be compliant, the target application shall supply at least 256 bits of entropy in order to meet the security strength required for the random number generation mechanism. Since entropy is loaded passively into the module, there is no assurance of the minimum strength of generated SSPs (e.g., keys).

2.9 Key Generation

SSPs that are generated internally by the module, are generated using the module's approved DRBG.

2.10 Key Establishment

The module is capable of performing key establishment when utilizing the implemented NIST SP 800-56Ar3 shared secret computation methods, with one of the approved key derivation functions. The module also supports key transport methods compliant with NIST SP 800-38F.

2.11 Industry Protocols

While the module does not implement the TLS or SSH protocols, it does implement the key derivation functions for both, per NIST SP 800-135r1.

2.12 Additional Information

Please see Section 11 for details regarding the preparation of the operational environments, and installation of the module.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	Input data passed via API calls as function arguments or in memory buffers referenced by function arguments
N/A	Data Output	Data returned by API calls using function arguments and related memory buffers
N/A	Control Input	API function calls that initialize and control the operation of the module
N/A	Status Output	Values returned from API calls

Table 22: Ports and Interfaces

As a software cryptographic module, the module's physical and electrical characteristics, manual controls and physical indicators are those of the host system. The host system provides physical ports that the operating system or applications may use. The cryptographic module does not access or control the physical interface ports or physical indicators of the GPC. The module does not support a control output interface.

4 Roles, Services, and Authentication

4.1 Authentication Methods

N/A for this module.

The module does not identify or authenticate the operator. The Crypto Officer role is assumed by the operator. Only one operator can operate the module at any time.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
CO	Role	Crypto Officer	None

Table 23: Roles

The Allegro Cryptographic Engine supports only one role, which is the Crypto Officer role.

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
AcInit()	Initialize the module for use in Approved mode	Successful Invocation (Pass)	N/A	Invocation Success or Invocation Failure	None	CO
AcDeInit()	Zeroize all keys and CSPs and disable crypto services	Successful Invocation (Pass)	N/A	Invocation Success or Invocation Failure	None	CO
AcRunSelfTest()	Run cryptographic self-tests on demand	Successful Invocation (Pass)	N/A	Invocation Success or Invocation Failure	DRBG Message Digest Generate Digital	CO

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					Signature Verify Digital Signature Generate Asymmetric Keys Shared Secret Computation (KAS-FFC-SSC) Shared Secret Computation (KAS-ECC-SSC) Derive Key (HKDF) Derive Key (TLS 1.2) Derive Key (TLS 1.3) Derive Key (SSH) Derive Key (PBKDF) Message Authentication Symmetric Cipher Verify Asymmetric Keys	
AcAceLibraryInfo()	Return the module name and version	Successful Invocation (Pass)	N/A	Module Name, Major Version, Minor Version, Build Number, Invocation Success or Invocation Failure	None	CO
AcGenerateRandomNumbers()	Generate random data	Successful Invocation (Pass)	API call parameters	Random Number, Invocation Success or Invocation Failure	DRBG	CO

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
AcDigest() AcDigestInit() AcDigestUpdate() AcDigestFinal()	Create message digest from input data	Successful Invocation (Pass)	API call parameters	Hash, Invocation Success or Invocation Failure	Message Digest	CO
AcDigestClone()	Duplicate a message digest	Successful Invocation (Pass)	API call parameters	Hash, Invocation Success or Invocation Failure	None	CO
AcKeyedDigestInit()	Create a keyed message digest of input data	Successful Invocation (Pass)	API call parameters	Digest, Invocation Success or Invocation Failure	Message Digest Message Authentication	CO - HMAC Key: R,E - AES GMAC Key: R,E - AES CMAC Key: R,E
AcSign() AcSignInit() AcSignUpdate() AcSignFinal()	Create a Digital Signature	Successful Invocation (Pass)	API call parameters	Signature, Invocation Success or Invocation Failure	Generate Digital Signature	CO - RSA Private Key: R,E - ECDSA Private Key: R,E
AcSignDigestBuffer()	Create a digital signature for a previously computed message digest	Successful Invocation (Pass)	API call parameters	Signature, Invocation Success or Invocation Failure	Generate Digital Signature	CO - RSA Private Key: R,E - ECDSA Private Key: R,E
AcVerify() AcVerifyInit() AcVerifyUpdate() AcVerifyFinal()	Verify a digital signature	Successful Invocation (Pass)	API call parameters	Signature, Invocation Success or Invocation Failure	Verify Digital Signature Verify Asymmetric Keys	CO - RSA Public Key: R,E - ECDSA Public Key: R,E
AcVerifyDigestBuffer()	Verify a digital signature for a previously computed digest	Successful Invocation (Pass)	API call parameters	Signature, Invocation Success or Invocation Failure	Verify Digital Signature Verify Asymmetric Keys	CO - RSA Public Key: R,E - ECDSA Public Key: R,E
AcEncryptInit()	Encrypt or decrypt a block of data	Successful Invocation (Pass)	API call parameters	Plaintext, Ciphertext, Invocation Success or Invocation Failure	Symmetric Cipher	CO - AES Key : R,E - AES GCM Key: R,E - AES GCM IV: R,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- AES CCM Key: R,E - AES-XTS Testing Revision 2.0 Key: R,E - AES CMAC Key: R,E - AES GMAC Key: R,E
AcEncryptUpdate()	Encrypt or decrypt a block of data	Successful Invocation (Pass)	API call parameters	Plaintext, Ciphertext, Invocation Success or Invocation Failure	Symmetric Cipher	CO - AES Key : R,E - AES GCM Key: R,E - AES GCM IV: R,E - AES CCM Key: R,E - AES-XTS Testing Revision 2.0 Key: R,E - AES CMAC Key: R,E - AES GMAC Key: R,E
AcEncryptFinal()	Encrypt or decrypt a block of data	Successful Invocation (Pass)	API call parameters	Plaintext, Ciphertext, Invocation Success or Invocation Failure	Symmetric Cipher	CO - AES Key : R,E - AES GCM Key: R,E - AES GCM IV: R,E - AES CCM Key: R,E - AES-XTS Testing Revision 2.0 Key: R,E - AES CMAC Key: R,E - AES GMAC Key: R,E
AcGenerateKey()	Generate symmetric keys	Successful Invocation (Pass)	API call parameters	Key, Invocation Success or Invocation Failure	Generate Symmetric Keys CKG Section 4	CO - AES Key : G,W - AES GCM Key: G,W

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					CKG Section 6.1	- AES CMAC Key: G,W - AES-XTS Testing Revision 2.0 Key: G,W - Key Encryption Key (KEK): G,W
AcGenerateKeyPair()	Generate asymmetric key pairs	Successful Invocation (Pass)	API call parameters	KeyPair, Invocation Success or Invocation Failure	Generate Asymmetric Keys CKG Section 4	CO - RSA Private Key: G,W - RSA Public Key: G,W - ECDSA Private Key: G,W - ECDSA Public Key: G,W - ECDH Private Components: G,W - ECDH Public Components: G,W - DH Private Components: G,W - DH Public Components: G,W
AcBuildKeyPairFromParams()	Generate asymmetric key pairs using specific key parameters	Successful Invocation (Pass)	API call parameters	KeyPair, Invocation Success or Invocation Failure	Generate Asymmetric Keys Shared Secret Computation (KAS-FFC-SSC) Shared Secret Computation (KAS-ECC-SSC)	CO - RSA Private Key: G,W - RSA Public Key: G,W - ECDSA Private Key: G,W - ECDSA Public Key: G,W - ECDH Private Components: G,W

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- ECDH Public Components: G,W - DH Private Components: G,W - DH Public Components: G,W
AcExportKey()	Wrap Key	Successful Invocation (Pass)	API call parameters	Key/KeyPair, Invocation Success or Invocation Failure	Key Wrapping	CO - Key Encryption Key (KEK): R,E
AcImportKey()	Unwrap Key	Successful Invocation (Pass)	API call parameters	Key/KeyPair, Invocation Success or Invocation Failure	Key Wrapping	CO - Key Encryption Key (KEK): W,E
AcKeySize()	Return the key size for a selected Key	Successful Invocation (Pass)	API call parameters	Key Size, Invocation Success or Invocation Failure	None	CO
AcKeyExchange()	Establish a shared secret	Successful Invocation (Pass)	API call parameters	Shared, Secret, Key, Invocation Success or Invocation Failure	Shared Secret Computation (KAS-FFC-SSC) Shared Secret Computation (KAS-ECC-SSC)	CO - ECDH Private Components: R,E - ECDH Public Components: R,E - DH Private Components: R,E - DH Public Components: R,E
AcDeriveKey()	Derive a key	Successful Invocation (Pass)	API call parameters	Key, Invocation Success or Invocation Failure	Derive Key (TLS 1.2) Derive Key (TLS 1.3) Derive Key (PBKDF)	CO - TLS Session Key: G,W - PBKDF2 DPK: G,W - AES Key : G,W - TLS RSA Premaster Secret: G,W - TLS Master

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Secret: G,W - TLS Integrity Key: G,W - PBKDF2 Password: G,W - TLS Extended Master Secret: G,W
AcReleaseHandle()	Zeroize Keys	Successful Invocation (Pass)	API call parameters	Invocation Success or Invocation Failure	None	CO - AES Key : Z - AES GCM Key: Z - AES GCM IV: Z - AES CCM Key: Z - AES-XTS Testing Revision 2.0 Key: Z - AES CMAC Key: Z - AES GMAC Key: Z - HMAC Key: Z - Key Encryption Key (KEK): Z - PBKDF2 DPK: Z - PBKDF2 Password: Z - RSA Private Key: Z - ECDSA Private Key: Z - ECDH Private Components: Z - TLS RSA Premaster Secret: Z - TLS Master

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Secret: Z - TLS Session Key: Z - TLS Integrity Key: Z - DRBG Entropy: Z - DRBG Seed: Z - DRBG 'C' Value: Z - DRBG 'V' Value : Z - RSA Public Key: Z - ECDSA Public Key: Z - ECDH Public Components : Z - TLS Extended Master Secret: Z
AcAceLibraryStatus()	Query whether library is in the soft error state	Successful Invocation (Pass)	API call parameters	Invocation Success or Invocation Failure	None	CO
AcKeyedDigest()	Message authentication	Successful Invocation (Pass)	API call parameters	Invocation Success or Invocation Failure	Message Digest Message Authentication	CO - HMAC Key: R,E - AES CMAC Key: R,E - AES GMAC Key: R,E
AcDigestSize()	Message authentication digest size	Successful Invocation (Pass)	API call parameters	Invocation Success or Invocation Failure	None	CO
AcEncrypt()	Encrypt plaintext	Successful Invocation (Pass)	API call parameters	Ciphertext, Invocation Success or Invocation Failure	Message Authentication Symmetric Cipher	CO - AES Key : R,E - AES GCM Key: R,E - AES GCM IV: R,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- AES CCM Key: R,E - AES-XTS Testing Revision 2.0 Key: R,E - AES CMAC Key: R,E - AES GMAC Key: R,E
AcEncryptBlockSize()	Return size of ciphertext	Successful Invocation (Pass)	API call parameters	Invocation Success or Invocation Failure	None	CO
AcSetOperationParameter()	Sets/Configures non-security relevant operational parameters	Successful Invocation (Pass)	API call parameters	Invocation Success or Invocation Failure	None	CO
AcGetVendorImplementation()	Returns the Module's Algorithm capabilities	Successful Invocation (Pass)	API call parameters	Invocation Success or Invocation Failure	None	CO
AcAESFpeEncrypt( )	Encrypt plaintext	Successful Invocation (Pass)	API call parameters	Ciphertext, Invocation Success or Invocation Failure	Symmetric Cipher	CO - AES Key : R,E
AsGetCryptoDataPtr()	Get Data Pointer	Successful Invocation (Pass)	API call parameters	Data Pointer, Invocation Success or Invocation Failure	None	CO
AcGetAceError()	Get Error Message	Successful Invocation (Pass)	API call parameters	Error Message, Invocation Success or Invocation Failure	None	CO
AcGatherSystemNoise()	Gather Entropy Input	Successful Invocation (Pass)	API call parameters	Entropy Input, Invocation Success or Invocation Failure	None	CO - DRBG Entropy: G,W,E - DRBG 'V' Value : G,W,E - DRBG 'C' Value: G,W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
AcGetPersonalizationData()	Get DRBG Personalization Data	Successful Invocation (Pass)	API call parameters	Personalization Data, Invocation Success or Invocation Failure	None	CO - DRBG Personalization String: R - DRBG Seed: G,W,E
AcEncryptClone()	Duplicate an Encrypt Operation	Successful Invocation (Pass)	API call parameters	Encrypt Operation, Invocation Success or Invocation Failure	None	CO

Table 24: Approved Services

4.4 Non-Approved Services

N/A for this module.

4.5 External Software/Firmware Loaded

The module does not support the external loading of software or firmware.

5 Software/Firmware Security

5.1 Integrity Techniques

The module, which is made up of a single component, is provided in the form of binary executable code (Acelib.so for Linux and AceDll.dll for Windows). A software integrity test is performed on the runtime image of the module. The HMAC-SHA2-256 (Cert. #A3332) implemented in the module is used as an approved algorithm for the integrity test. If the test fails, the module enters an error state where no cryptographic services are provided, and data output is prohibited. (the module is not operational)

5.2 Initiate on Demand

The software integrity test is performed as part of the Pre-Operational self-tests. It is automatically executed at power-on. It can also be invoked by powering-off and reloading the module, or using the AcRunSelfTest() service.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

How Requirements are Satisfied:

The module operates in a modifiable operational environment as described by the FIPS 140-3 definition. The operating systems on which the module was tested run user processes in logically separate process spaces. When

the module is present in memory, the operating system protects the module's memory space from unauthorized access. The module functions entirely within the process space of the calling application.

7 Physical Security

The physical security requirements of FIPS 140-3 do not apply to software modules.

8 Non-Invasive Security

The module does not implement non-invasive attack mitigations.

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
RAM	Random Access Memory	Dynamic

Table 25: Storage Areas

The module does not persistently store SSPs.

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
Input	Calling Process	Call stack (API) input parameters	Plaintext	Manual	Electronic	
Output	Call stack (API) output parameters	Calling Process	Plaintext	Manual	Electronic	

Table 26: SSP Input-Output Methods

Note: To prevent the inadvertent output of sensitive information, two independent internal actions shall be required in order to output any plaintext CSP.

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Unload Module	Unload module from memory	SSPs no longer present in memory after unload	Operator unloads module
API Call	API zeroize instruction	SSPs no longer present in memory after API call	AcDeInit() AcReleaseHandle()
Remove Power	Power removed from host GPC	SSPs no longer present in memory after GPC power loss	Operator powers off GPC

Table 27: SSP Zeroization Methods

SSPs are implicitly zeroized when unloading the module or removing power, and explicitly zeroized when using AcDeInit() and AcReleaseHandle() where SSPs are overwritten with zeros.

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES Key	AES-ECB, AES-CBC, AES-CFB1, AES-CFB8, AES-CFB128, AES-CTR, AES-OFB, AES-FF1	128, 192, 256 bits - 128, 192, 256 bits	Symmetric - CSP	Generate Symmetric Keys		Symmetric Cipher
AES GCM Key	AES-GCM	128, 192, 256 bits - 128, 192, 256 bits	Symmetric - CSP	Generate Symmetric Keys		Symmetric Cipher
AES GCM IV	AES-GCM	96 bits - 96 bits	Initialization Vector - CSP	DRBG		Symmetric Cipher
AES CCM Key	AES-CCM	128, 192, 256 bits - 128, 192, 256 bits	Symmetric - CSP	Generate Symmetric Keys		Symmetric Cipher
AES-XTS Testing Revision 2.0 Key	AES-XTS Testing Revision 2.0	128, 256 bits - 128, 256 bits	Symmetric - CSP	Generate Symmetric Keys		Symmetric Cipher
AES CMAC Key	AES-CMAC	128, 192, 256 bits - 128, 192, 256 bits	Message Authentication - CSP	Generate Symmetric Keys		Message Authentication
AES GMAC Key	AES-GMAC	128, 192, 256 bits - 128, 192, 256 bits	Message Authentication - CSP	Generate Symmetric Keys		Message Authentication
HMAC Key	HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512, HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512	160, 224, 256, 384, 512 bits - 160, 224, 256, 384, 512 bits	Message Authentication - CSP	Message Authentication		Message Authentication
Key Encryption Key (KEK)	AES-KW, AES-KWP	128, 192, 256 bits - 128, 192, 256 bits	Symmetric Key Wrapping (KTS) - CSP	Generate Symmetric Keys		Key Wrapping
PBKDF2 DPK	PBKDF	112 bits - 112 bits	Data Protection Key - CSP	Derive Key (PBKDF)		Derive Key (PBKDF)

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
PBKDF2 Password	PBKDF Password	Greater than or equal to 80 bits of data - Greater than or equal to 80 bits of data	Password - CSP			Derive Key (PBKDF)
RSA Private Key	RSA (186-4)	2048, 3072, 4096 bits - 112, 128, 150 bits	Asymmetric Private Key - CSP	Generate Asymmetric Keys		Generate Digital Signature
ECDSA Private Key	ECDSA (186-4)	P-224, P-256, P-384, P-521 - 112, 128, 192, 256 bits	Asymmetric Private Key - CSP	Generate Asymmetric Keys		Generate Digital Signature
ECDH Private Components	KAS-ECC-SSC (NIST SP 800-56Ar3)	P-224, P-256, P-384, P-521 - 112, 128, 192, 256 bits	Asymmetric Private Key - CSP	Generate Asymmetric Keys		Shared Secret Computation (KAS-ECC-SSC)
TLS RSA Premaster Secret	Used to derive the master secret	384 bits - 384 bits	Premaster Secret - CSP			Derive Key (TLS 1.2)
TLS Master Secret	Used to generate the session keys	384 bits - 384 bits	Master Secret - CSP		Derive Key (TLS 1.2) Derive Key (TLS 1.3)	Derive Key (TLS 1.2) Derive Key (TLS 1.3)
TLS Session Key	Used for data encryption	128 or 256 bits - 128 or 256 bits	Session Key - CSP		Derive Key (TLS 1.2) Derive Key (TLS 1.3)	Derive Key (TLS 1.2) Derive Key (TLS 1.3)
TLS Integrity Key	Used for data integrity and authenticity	160 bits - 160 bits	Integrity Key - CSP		Derive Key (TLS 1.2) Derive Key (TLS 1.3)	Derive Key (TLS 1.2) Derive Key (TLS 1.3)
DRBG Entropy	NIST SP 800-90A DRBG Entropy Input	256 bits - 256 bits	Entropy Input - CSP			DRBG
DRBG Seed	NIST SP 800-90A DRBG Seed	440-888 bits - 440-888 bits	Seed - CSP			DRBG

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
DRBG ‘C’ Value	NIST SP 800-90A DRBG ‘C’ Value (IG D.L)	440-888 bits - 440-888 bits	Internal State Value - CSP	DRBG		DRBG
DRBG ‘V’ Value	NIST SP 800-90A DRBG ‘V’ Value (IG D.L)	440-888 bits - 440-888 bits	Internal State Value - CSP	DRBG		DRBG
RSA Public Key	RSA Public Key	2048, 3072, 4096 bits - 112, 128, 150 bits	Asymmetric Public Key - PSP	Generate Asymmetric Keys		Verify Digital Signature
ECDSA Public Key	ECDSA Public Key	P-224, P-256, P-384, P-521 - 112, 128, 192, 256 bits	Asymmetric Public Key - PSP	Generate Asymmetric Keys		Verify Digital Signature
ECDH Public Components	ECDH Public Components	P-224, P-256, P-384, P-521 - 112, 128, 192, 256 bits	Key Agreement Components - PSP	Generate Asymmetric Keys		Shared Secret Computation (KAS-ECC-SSC)
TLS Extended Master Secret	Binds the master secret to a log of the full handshake	384-bits - 384-bits	Extended Master Secret - CSP		Derive Key (TLS 1.2)	Derive Key (TLS 1.2)
DH Private Components	Private components for KAS-FFC-SSC	MODP 2048, MODP 3072 - 112 bits, 128 bits	Asymmetric Private Key - CSP	Generate Asymmetric Keys		Shared Secret Computation (KAS-FFC-SSC)
DH Public Components	Public components for KAS-FFC-SSC	MODP 2048, MODP 3072 - 112 bits, 128 bits	Asymmetric Public Key - PSP	Generate Asymmetric Keys		Shared Secret Computation (KAS-FFC-SSC)
DRBG Personalization String	Optional input to the DRBG instantiate function	128 to 256 bits - 128 to 256 bits	Personalization String - Neither			DRBG

Table 28: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES Key	Input	RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call	

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
				Remove Power	
AES GCM Key	Input	RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	AES GCM IV:Used With
AES GCM IV	Output	RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	AES GCM Key:Used With
AES CCM Key	Input	RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	
AES-XTS Testing Revision 2.0 Key	Input	RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	
AES CMAC Key	Input	RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	
AES GMAC Key	Input	RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	
HMAC Key	Input	RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	
Key Encryption Key (KEK)	Input Output	RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	
PBKDF2 DPK	Output	RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	PBKDF2 Password:Derived From
PBKDF2 Password	Input	RAM:Plaintext	In volatile memory until zeroized	Unload Module Remove Power	PBKDF2 DPK:Used With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
RSA Private Key	Input Output	RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	RSA Public Key:Paired With
ECDSA Private Key	Input Output	RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	ECDSA Public Key:Paired With
ECDH Private Components	Input Output	RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	ECDH Public Components:Paired With
TLS RSA Premaster Secret		RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	
TLS Master Secret		RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	TLS RSA Premaster Secret:Derived From
TLS Session Key		RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	TLS Master Secret:Derived From
TLS Integrity Key		RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	TLS Master Secret:Derived From
DRBG Entropy	Input	RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	DRBG Seed:Used With
DRBG Seed		RAM:Plaintext	In volatile memory until zeroized	Unload Module Remove Power	DRBG Entropy:Derived From
DRBG ‘C’ Value		RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	DRBG Seed:Derived From

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
DRBG ‘V’ Value		RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	DRBG Seed:Derived From
RSA Public Key	Input Output	RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	RSA Private Key:Paired With
ECDSA Public Key	Input Output	RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	ECDSA Private Key:Paired With
ECDH Public Components	Input Output	RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	ECDH Private Components:Paired With
TLS Extended Master Secret		RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	TLS RSA Premaster Secret:Derived From
DH Private Components	Input Output	RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	
DH Public Components	Input Output	RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	
DRBG Personalization String	Input	RAM:Plaintext	In volatile memory until zeroized	Unload Module API Call Remove Power	

Table 29: SSP Table 2

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A3332)	HMAC-SHA2-256	Software Integrity Test	SW/FW Integrity	Pass	Keyed hash performed on Acelib.so or AceDll.dll

Table 30: Pre-Operational Self-Tests

The module performs the pre-operational software integrity test automatically upon every instantiation. (A CAST for HMAC-SHA2-256 executes prior to the software integrity test.)

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC (A3332)	128, 192, 256 bit Key	KAT	CAST	Pass	Encrypt	Power-On
AES-CBC (A3332)	128, 192, 256 bit Key	KAT	CAST	Pass	Decrypt	Power-On
AES-CCM (A3332)	128, 192, 256 bit Key	KAT	CAST	Pass	Encrypt	Power-On
AES-CCM (A3332)	128, 192, 256 bit Key	KAT	CAST	Pass	Decrypt	Power-On
AES-CFB1 (A3332)	128, 192, 256 bit Key	KAT	CAST	Pass	Encrypt	Power-On
AES-CFB1 (A3332)	128, 192, 256 bit Key	KAT	CAST	Pass	Decrypt	Power-On
AES-CFB128 (A3332)	128, 192, 256 bit Key	KAT	CAST	Pass	Encrypt	Power-On
AES-CFB128 (A3332)	128, 192, 256 bit Key	KAT	CAST	Pass	Decrypt	Power-On
AES-CFB8 (A3332)	128, 192, 256 bit Key	KAT	CAST	Pass	Encrypt	Power-On
AES-CFB8 (A3332)	128, 192, 256 bit Key	KAT	CAST	Pass	Decrypt	Power-On
AES-CMAC (A3332)	128, 192, 256 bit Key	KAT	CAST	Pass	Encrypt	Power-On
AES-CMAC (A3332)	128, 192, 256 bit Key	KAT	CAST	Pass	Decrypt	Power-On
AES-CTR (A3332)	128, 192, 256 bit Key	KAT	CAST	Pass	Encrypt	Power-On
AES-CTR (A3332)	128, 192, 256 bit Key	KAT	CAST	Pass	Decrypt	Power-On
AES-ECB (A3332)	128, 192, 256 bit Key	KAT	CAST	Pass	Encrypt	Power-On
AES-ECB (A3332)	128, 192, 256 bit Key	KAT	CAST	Pass	Decrypt	Power-On
AES-FF1 (A3332)	128 bit Key	KAT	CAST	Pass	Encrypt	Power-On
AES-GCM (A3332)	128, 192, 256 bit Key	KAT	CAST	Pass	Encrypt	Power-On
AES-GCM (A3332)	128, 192, 256 bit Key	KAT	CAST	Pass	Decrypt	Power-On
AES-OFB (A3332)	128, 192, 256 bit Key	KAT	CAST	Pass	Encrypt	Power-On
AES-OFB (A3332)	128, 192, 256 bit Key	KAT	CAST	Pass	Decrypt	Power-On

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-XTS Testing Revision 2.0 (A3332)	128, 256 bit Key	KAT	CAST	Pass	Encrypt	Power-On
AES-XTS Testing Revision 2.0 (A3332)	128, 256 bit Key	KAT	CAST	Pass	Decrypt	Power-On
ECDSA SigGen (FIPS186-4) (A3332)	P-384 Curve	KAT	CAST	Pass	Sign	Power-On
ECDSA SigVer (FIPS186-4) (A3332)	P-384 Curve	KAT	CAST	Pass	Verify	Power-On
Hash DRBG (A3332)	SHA2-256, SHA2-512 (NIST SP 800-90A, Section 11.3 Health Tests)	KAT	CAST	Pass	Hash_DRBG	1. Power On 2. Instantiate: Any time that a new DRBG instance is created 3. Generate: When new random data is generated 4. Reseed: When the reseed counter has reached its pre-determined maximum value and the DRBG needs to be reseeded
HMAC-SHA-1 (A3332)	160 bit Hash	KAT	CAST	Pass	Keyed Hash	Power-On
HMAC-SHA2-224 (A3332)	224 bit Hash	KAT	CAST	Pass	Keyed Hash	Power-On
HMAC-SHA2-256 (A3332)	256 bit Hash	KAT	CAST	Pass	Keyed Hash	Power-On
HMAC-SHA2-384 (A3332)	384 bit Hash	KAT	CAST	Pass	Keyed Hash	Power-On
HMAC-SHA2-512 (A3332)	512 bit Hash	KAT	CAST	Pass	Keyed Hash	Power-On
HMAC-SHA3-224 (A3332)	224 bit Hash	KAT	CAST	Pass	Keyed Hash	Power-On
HMAC-SHA3-256 (A3332)	256 bit Hash	KAT	CAST	Pass	Keyed Hash	Power-On
HMAC-SHA3-384 (A3332)	384 bit Hash	KAT	CAST	Pass	Keyed Hash	Power-On

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA3-512 (A3332)	512 bit Hash	KAT	CAST	Pass	Keyed Hash	Power-On
KAS-ECC-SSC Sp800-56Ar3 (A3332)	P-384 Curve	KAT	CAST	Pass	Primitive "Z"	Power-On
KAS-FFC-SSC Sp800-56Ar3 (A3332)	MODP-2048, MODP-3072	KAT	CAST	Pass	Primitive "Z"	Power-On
KDA HKDF Sp800-56Cr1 (A3332)	SHA2-256	KAT	CAST	Pass	Hash	Power-On
KDF SSH (A3332)	AES-128, AES-192, AES-256, SHA2-256	KAT	CAST	Pass	Hash	Power-On
PBKDF (A3332)	160 bit Keyed Hash	KAT	CAST	Pass	Keyed Hash	Power-On
RSA SigGen (FIPS186-4) (A3332)	2048 bit Key	KAT	CAST	Pass	Sign	Power-On
RSA SigVer (FIPS186-4) (A3332)	2048 bit Key	KAT	CAST	Pass	Verify	Power-On
SHA-1 (A3332)	160 bit Hash	KAT	CAST	Pass	Hash	Power-On
SHA2-224 (A3332)	224 bit Hash	KAT	CAST	Pass	Hash	Power-On
SHA2-256 (A3332)	256 bit Hash	KAT	CAST	Pass	Hash	Power-On
SHA2-384 (A3332)	384 bit Hash	KAT	CAST	Pass	Hash	Power-On
SHA2-512 (A3332)	512 bit Hash	KAT	CAST	Pass	Hash	Power-On
SHA3-224 (A3332)	224 bit Hash	KAT	CAST	Pass	Hash	Power-On
SHA3-256 (A3332)	256 bit Hash	KAT	CAST	Pass	Hash	Power-On
SHA3-384 (A3332)	384 bit Hash	KAT	CAST	Pass	Hash	Power-On
SHA3-512 (A3332)	512 bit Hash	KAT	CAST	Pass	Hash	Power-On
TLS v1.2 KDF RFC7627 (A3332)	SHA2-256	KAT	CAST	Pass	Hash	Power-On

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
TLS v1.3 KDF (A3332)	SHA2-256	KAT	CAST	Pass	Hash	Power-On
AES-XTS Testing Revision 2.0 (A3332)	128, 256 bit key	Key_1 ≠ Key_2 (IG C.I)	CAST	Pass	Check	Conditional upon first use of AES-XTS
KAS-ECC-SSC Sp800-56Ar3 (A3332)	P-224, P-256, P-384, P-521	Public Key Assurance Test	CAST	Pass	Check	Conditional upon ECDSA KeyPair Generation
ECDSA KeyGen (FIPS186-4) (A3332)	P-384	Pairwise Consistency Test	PCT	Pass	Sign & Verify	Conditional upon ECDSA KeyPair Generation
RSA KeyGen (FIPS186-4) (A3332)	2048 bit key	Pairwise Consistency Test	PCT	Pass	Sign & Verify	Conditional upon RSA KeyPair Generation

Table 31: Conditional Self-Tests

Conditional CASTs are performed automatically upon every instantiation. The pairwise consistency tests are performed on the condition that an asymmetric keypair is requested, and the AES-XTS key validation test is performed prior to using the keys, per IG C.I. The DRBG health tests required by NIST SP 800-90A, Section 11.3 (instantiate, generate, reseed) execute upon instantiation of the module and also upon the following conditions:

1. **Instantiate**: Any time that a new DRBG instance is created.
2. **Generate**: When new random data is generated.
3. **Reseed**: When the reseed counter has reached its pre-determined maximum value and the DRBG needs to be reseeded.

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A3332)	Software Integrity Test	SW/FW Integrity	On Demand	Reload Module or AcRunSelfTest()

Table 32: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CBC (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
AES-CBC (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
AES-CCM (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
AES-CCM (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
AES-CFB1 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CFB1 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
AES-CFB128 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
AES-CFB128 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
AES-CFB8 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
AES-CFB8 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
AES-CMAC (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
AES-CMAC (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
AES-CTR (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
AES-CTR (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
AES-ECB (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
AES-ECB (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
AES-FF1 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
AES-GCM (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
AES-GCM (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
AES-OFB (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
AES-OFB (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
AES-XTS Testing Revision 2.0 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
AES-XTS Testing Revision 2.0 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
ECDSA SigGen (FIPS186-4) (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
ECDSA SigVer (FIPS186-4) (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
Hash DRBG (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
HMAC-SHA-1 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
HMAC-SHA2-224 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
HMAC-SHA2-256 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-384 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
HMAC-SHA2-512 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
HMAC-SHA3-224 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
HMAC-SHA3-256 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
HMAC-SHA3-384 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
HMAC-SHA3-512 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
KAS-ECC-SSC Sp800-56Ar3 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
KAS-FFC-SSC Sp800-56Ar3 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
KDA HKDF Sp800-56Cr1 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
KDF SSH (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
PBKDF (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
RSA SigGen (FIPS186-4) (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
RSA SigVer (FIPS186-4) (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
SHA-1 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
SHA2-224 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
SHA2-256 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
SHA2-384 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
SHA2-512 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
SHA3-224 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
SHA3-256 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
SHA3-384 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
SHA3-512 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
TLS v1.2 KDF RFC7627 (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()
TLS v1.3 KDF (A3332)	KAT	CAST	On Demand	Reload Module or AcRunSelfTest()

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-XTS Testing Revision 2.0 (A3332)	Key_1 ≠ Key_2 (IG C.I)	CAST	N/A	N/A
KAS-ECC-SSC Sp800-56Ar3 (A3332)	Public Key Assurance Test	CAST	N/A	N/A
ECDSA KeyGen (FIPS186-4) (A3332)	Pairwise Consistency Test	PCT	N/A	N/A
RSA KeyGen (FIPS186-4) (A3332)	Pairwise Consistency Test	PCT	N/A	N/A

Table 33: Conditional Periodic Information

The pre-operational software integrity test and all conditional CASTs can be executed on-demand by calling the AcRunSelfTest() service.

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Hard Error State	Non-recoverable error state	Result of pre-operational self-test failure Result of CAST failure	Reload module / Reinstall module	Fail
Soft Error State	Recoverable error state	Result of RSA pairwise consistency test failure Result of ECDSA pairwise consistency test failure Result of Key_1 ≠ Key_2 for AES-XTS	Automatic	Fail

Table 34: Error States

The module implements two error states. A hard error state, whereby recovery may be attempted by restarting or reinstalling the module, and a software error state whereby conditional self-test failures such as the pairwise tests and the AES-XTS key validation test may be recovered. Self-tests in the module return an indication of whether the invocation passed or failed.

If any self-test fails, the module's data output interfaces will be inhibited, and only control input and status output commands will be allowed to execute. To correct an on-demand or conditional self-test error, the module must be restarted by calling the AllegroTaskInit() service after the module has been de-initialized. To correct a pre-operational self-test error, the module must be reloaded into memory by terminating and restarting the host application. If the pre-operational self-test fails after restarting the host application, it will be necessary to re-install the module.

10.5 Operator Initiation of Self-Tests

The pre-operational software integrity test and all conditional CASTs can be executed by the operator using the API call AcRunSelfTest().

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

When built and executed, the module automatically operates in the approved mode. (Additional guidance is provided in Section 11.2.)

11.2 Administrator Guidance

Initial setup for the module consists of:

1. Installing the host operating system: Linux 5.15 (Mint 21) or Windows 11 Pro.
2. Creating a new user account on the Operating System (and providing that user account with a password).

The host operating system will provide the operational environment required for the module to meet FIPS 140-3, Level 1 security specifications. The Crypto Officer will create a new admin account per the guidelines of the OS user manual. (The Crypto Officer shall refer to all administrative and guidance documents in order to create a new user account on the Operating System.)

The Crypto Officer is in charge of the secure management and handling of the module. The Allegro Cryptographic Engine is shipped on a DVD and delivered via FedEx. A tracking number is provided to the Crypto Officer in order to track the progress of the shipment and ensure secure delivery of the module. The Crypto Officer shall sign for the DVD upon arrival and shall maintain control of the DVD throughout its lifetime. Following the secure delivery of the module, the Crypto Officer shall first follow the steps outlined above.

After the operational environment has been prepared, the module can be built in any configuration specified in Table 2 of this security policy, by consulting the build instructions provided in Chapter 5 of the **ACE™ Allegro Cryptography Engine Programming Reference, Version 6.50**, included on the DVD.

During normal operation, the operator may check the status of the module by attempting to run a service. If the service executes and does not return an error, the module is operating in the approved mode.

11.3 Non-Administrator Guidance

The module supports the role of Crypto Officer only, which is an administrative role.

11.4 End of Life

The module may be sanitized by uninstalling the binary and power-cycling the host GPC.

11.5 Additional Information

The operator shall adhere to the guidelines of this Security Policy. Operators in the Crypto Officer role are able to use the approved services listed in this security policy. The operator is responsible for monitoring the module for any irregular activity.

12 Mitigation of Other Attacks

The module does not attempt to mitigate specific attacks.