FIPS 140-2 Non-Proprietary Security Policy

Allegro Cryptographic Engine
Software Version: 6.2

FIPS Security Level: 1
Document Version: 0.5

Allegro Software Development Corporation
1740 Massachusetts Avenue
Boxborough, MA 01719
Telephone: (978) 264-6600
Fax: (978) 266-2839
www.allegrosoft.com

Allegro Software Development Corporation
1740 Massachusetts Avenue
Boxborough, MA 01719
Telephone: (978) 264-6600
Fax: (978) 266-2839
www.allegrosoft.com

FIPS 140-2 Non-Proprietary Security Policy, Version 6.2
Allegro Cryptographic Engine
May 31, 2017

Contents

List of Figures

List of Tables

1 INTRODUCTION

This document is the non-proprietary Cryptographic Module Security Policy for the Allegro Cryptographic Engine from Allegro Software Development Corporation. It describes how the Allegro Cryptographic Engine meets the security requirements of Federal Information Processing Standards (FIPS) Publication 140-2, as well as how to run the module in the secure FIPS-Approved mode of operation. This security policy was prepared as part of the Level 1 FIPS 140-2 validation of the module. It may be reproduced and distributed only in its original entirety including its copyright notice and without any revision.

The Allegro Cryptographic Engine is referred to in this document as ACE, the crypto module, or the module.

2 ALLEGRO CRYPTOGRAPHIC ENGINE

2.1 Cryptographic Module Specification

The Allegro Cryptographic Engine (ACE) is a general-purpose, software cryptographic module providing FIPS 140-2 Approved cryptography that can be used by calling applications via a C language Application Programming Interface. ACE meets the overall requirements applicable to a multi-chip stand-alone embodiment at Level 1 security of FIPS 140-2. ACE 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.

The module is packaged as a dynamic link library for Windows 10 Professional and as a shared object for Linux Mint Version 18. The module also includes a data file that is used for verifying the integrity of the module. ACE has been validated as a cryptographic module on Windows 10 Professional and Linux Mint Version 18.

The module was evaluated in the following configurations.

- Microsoft Surface Pro 4 with an Intel Core i7 processor running the Windows 10 Professional Operating System with PAA.
- Microsoft Surface Pro 4 with an Intel Core i7 processor running the Windows 10 Professional Operating System without PAA.
- Intel NUC6i7KYK with an Intel Core i7 processor running Linux Mint Version 18 with PAA..
- Intel NUC6i7KYK with an Intel Core i7 processor running Linux Mint Version 18 without PAA

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

The module's logical cryptographic boundary includes the two files that are needed for operation.
On Windows 10 Professional: AceDll.dll and AceDll.dll.dat.
On Linux Mint Version 18: libfipsace.so and libfipsace.dat.

The logical interface of the module is its Application Programming Interface (API) 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.

ACE meets the overall requirements applicable at Level 1 security of FIPS 140-2.

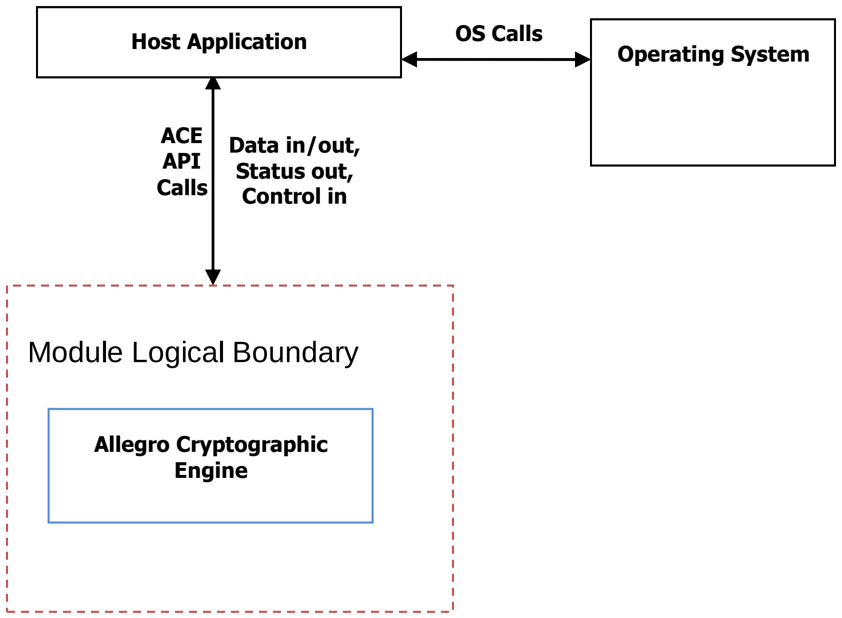
Table 2-1 shows the security level claimed for each of the eleven sections of FIPS 140-2:

Table 2-1. Security Level Per FIPS 140-2 Section

Section	Section Title	Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1
3	Roles, Services, and Authentication	1
4	Finite State Model	1
5	Physical Security	N/A
6	Operational Environment	1
7	Cryptographic Key Management	1
8	EMI/EMC	1
9	Self-tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A

Figure 2-1 shows a logical block diagram of the module executing in memory and its interactions with surrounding software components, as well as the module's logical cryptographic boundary. ACE supports an Application Programming Interface (API) which provides logical interfaces between the calling application and the module's services.

Figure 2-1. Allegro Cryptographic Engine Logical Cryptographic Boundary



The Allegro Cryptographic Engine implements the FIPS-Approved algorithms listed in Table 2-2.

Table 2-2. FIPS-Approved Algorithm Implementations

Algorithm	Certificate Number
AES encryption/decryption Modes: ECB, CBC, CTR, CFB1, CFB8, CFB128, OFB, CCM Key sizes: 128, 192, and 256 bits	4121
AES wrap/unwrap Key sizes: 128, 192, and 256 bits	4121
AES-GCM encryption/decryption and message authentication Key sizes: 128, 192, and 256 bits	4121
XTS-AES encryption/decryption with XTS_128- and XTS_256-bit keys	4121
AES CMAC generation and verification Key sizes: 128, 192, and 256 bits	4121
Triple-DES encryption/decryption; 3-key Modes: ECB, CBC, CFB1, CFB8, CFB64, OFB	2251
RSA (FIPS186-4) key pair generation Key sizes: 2048 and 3072 bits	2227
RSA (FIPS186-4) (ANSI X9.31) Signature generation Key sizes: 2048 and 3072 bits	2227
RSA (FIPS186-4) (ANSI X9.31) Signature verification Key sizes: 2048 and 3072 bits	2227
RSA (FIPS186-4) (PKCS #1 v1.5) Signature generation Key sizes: 2048 and 3072 bits	2227
RSA (FIPS186-4) (PKCS #1 v1.5) Signature verification Key sizes: 2048 and 3072 bits	2227
RSA (FIPS186-4) (PSS) Signature generation Key sizes: 2048 and 3072 bits	2227
RSA (FIPS186-4) (PSS) Signature verification Key sizes: 2048 and 3072 bits	2227
DSA (FIPS186-4) key pair generation Key sizes: 2048 and 3072 bits	1116
DSA (FIPS186-4) signature generation Key sizes: 2048 and 3072 bits	1116
DSA (FIPS186-4) signature verification Key sizes: 2048 and 3072 bits	1116
DSA (FIPS186-4) PQG Generation and Verification Key sizes: 2048 and 3072 bits	1116

Algorithm	Certificate Number
ECDSA (FIPS186-4) key pair generation NIST curves: P-224, P-256, P-384, and P- 521	936
ECDSA (FIPS186-4) signature generation NIST curves: P-224, P-256, P-384, and P- 521	936
ECDSA (FIPS186-4) signature verification NIST curves: P-224, P-256, P-384, and P- 521	936
SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512	3390
SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128 and SHAKE256	8
HMAC with SHA-1, SHA-224, SHA-256. SHA-384, and SHA-512	2692
Diffie-Hellman FFC Key Agreement Scheme (SP800-56A)	927
EC Diffie-Hellman ECC Key Agreement Scheme (SP800-56A) NIST curves: P-224, P-256, P-384, and P-521	927
TLS Key Derivation Functions (SP 800-135)	1074
SP 800-90A Hash_DRBG	1241

Caveats:

- The module generates keys per Scenario 1 of IG 7.8.
- The module implements MD5 for use with TLS communications, which is allowed in the FIPS-approved mode of operation.
- The module provides key derivation functions for use in TLS. The implementation of the TLS protocol has not been reviewed or tested by the CAVP and CMVP.

The module employs the following key establishment methodologies, which are allowed for use in a FIPS-approved mode of operation:

- RSA (key wrapping; key establishment methodology provides 112 or 128 bits of encryption strength)
- Diffie-Hellman (CVL Cert. #927, key agreement; key establishment methodology provides 112 or 128 bits of encryption strength)
- EC Diffie-Hellman (CVL Cert. #927, key agreement; key establishment methodology provides between 112 and 256 bits of encryption strength)

Allegro Software Development Corporation affirms compliance with SP 800-132 for the full implementation of PBKDF2[1]. The Allegro Cryptographic Engine requires the password to be at least ten characters in length, the iteration count is at least 1000, the salt is at least 128 bits in length and that the master key output from the PBDKF2 is at least 112 bits in length. Master keys may be used as Device Protection Keys (option 1(a) from section 5.4 of SP 800-132) or they may be used with a key derivation function to produce a Device Protection Key (option 1(b) from section 5.4 of SP 800-132).

2.2 Cryptographic Module 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 physical host system.

The module's Application Programming Interface provides the following logical interfaces defined by FIPS 140-2:

- Data input
- Data output
- Control input
- Status output

The host's physical ports are enumerated in Table 2-3 and characterized using the interface definitions of FIPS 140-2. Additionally, the logical interface column of Table 2-3 describes the logical interfaces supported by the ACE API.

[1] PBKDF2 – Password-Based Key Derivation Function 2 - PBKDF2 is published in Internet Engineering Task Force Request for Comments (RFC) 2898 and maps to PBKDF defined in NIST SP 800-132

Table 2-3. FIPS Logical Interface Mapping

FIPS-140-2 Interface	Physical Interface		Logical Interface
	Microsoft Surface Pro 4	Intel NUC6i7KYK	
Data Input	Keyboard (1) USB (1) 802.11ac (1) Bluetooth (1) MicroSD card slot (1) Touchscreen (1) Surface Pen (1) Infra-red (1) Microphone (2) Camera (2) Ambient light sensor (1) Accelerometer (1) Gyroscope (1)	Keyboard (1) USB (4) 802.11ac (1) Bluetooth (1) Ethernet (1) SD card slot (1) Infra-red (1) Microphone (1)	Input data passed via ACE API calls as function arguments or in memory buffers referenced by function arguments
Data Output	Display (1) USB (1) 802.11ac (1) Bluetooth (1) MicroSD card slot (1) SurfaceConnect (1) Speakers (2) Headset jack (1)	HDMI (1) Mini DisplayPort (1) USB (4) 802.11ac (1) Bluetooth (1) Ethernet (1) SD card (1) Headset jack (1) Speaker/TOSLINK port (1) Thunderbolt port (1)	Data returned by ACE API calls using function arguments and related memory buffers
Control Input	Keyboard (1) USB (1) 802.11ac (1) Bluetooth (1) MicroSD card slot (1) Touchscreen (1) Surface Pen (1) Infra-red (1) Microphone (2) Camera (2) Ambient light sensor (1) Accelerometer (1) Gyroscope (1) Power input (1) Power switch (1) Volume control switch (2)	Keyboard(1) USB (4) 802.11ac(1) Bluetooth (1) Ethernet (1) SD card (1) Infra-red (1) Microphone (1) Power input (1) Power switch (1)	ACE API function calls that initialize and control the operation of the module.
Status Output	Display (1) USB port (1) 802.11ac (1) Bluetooth (1) MicroSD card slot (1) SurfaceConnect (1) Speakers (2) Headset jack (1)	HDMI (1) Mini DisplayPort (1) USB (4) 802.11ac (1) Bluetooth (1) Ethernet (1) SD card (1) Headset jack (1) Speaker/TOSLINK port (1) Thunderbolt port (1)	Values returned from ACE API calls.

2.3 Roles, Services and Authentication

The Allegro Cryptographic Engine supports the Crypto Officer role and the User role. The

Maintenance role is not supported. An operator must be successfully authenticated by the operating system before accessing the module. The Module does not identify or authenticate the operator that is accessing the Module. Roles are assumed implicitly based on the service that is accessed by the operator. Only one operator assuming a specific role may operate the module at any time. Services associated with each role are listed in Sections 2.3.1 and 2.3.2.

The keys and CSPs listed in Table 2-4, and Table 2-5 indicate the type of access required using the following notation:

- R – Read: The CSP is read.
- W – Write: The CSP is established, generated, modified, or zeroized.
- X – Execute: The CSP is used within an Approved or Allowed security function mechanism.

2.3.1 Crypto Officer Role and Services

The Crypto Officer (CO) role is assumed to perform the initial installation of the module onto the Operating System. The CO role initializes the module for operation (AllegroTaskInit) as well as performs zeroization of all keys and CSPs during de-initialization (AllegroTaskDeInit). The CO role can also perform on-demand self-tests (AcRunSelfTest). Descriptions of the services available to the CO are provided in Table 2-4.

Table 2-4. Crypto Officer Services

Service	Description	CSP and Type of Access
AllegroTaskInit()	Initialize the module for use in FIPS mode	None
AllegroTaskDeInit()	Zeroize all keys and CSPs Disable Crypto Services	All CSPs – W
AcRunSelfTest()	Run cryptographic self-tests on-demand	None

2.3.2 User Role and Services

The User role is assumed when services such as random number generation, digest calculation, encryption and decryption, key wrapping and unwrapping, message authentication and signature generation and verification are requested. The User role services are described in Table 2-5.

Table 2-5. User Services

Service	Description	CSP and Type of Access
AcGenerateRandomNumbers()	Generate random data	DRBG Entropy – R/X DRBG 'V' Value –W/R DRBG 'C' Value –W/R

Service	Description	CSP and Type of Access

AcDigest()
AcDigestInit()
AcDigestUpdate()
AcDigestFinal()
	Create message digest from input data	None

AcDigestClone()	Duplicate a message digest	None

AcKeyedDigestInit()
AcDigestUpdate()
AcDigestFinal()
	Create a keyed message digest of input data	HMAC Key – R/X
		AES GMAC Key – R/X
		AES CMAC Key – R/X

AcSign()
AcSignInit()
AcSignUpdate()
AcSignFinal()
	Create a digital signature	RSA Private Key – R/X
		DSA Private Key – R/X
		ECDSA Private Key – R/X

AcSignDigestBuffer()	Create a digital signature for a previously computed message digest	RSA Private Key – R/X DSA Private Key – R/X ECDSA Private Key – R/X

AcVerify()
AcVerifyInit()
AcVerifyUpdate()
AcVerifyFinal()
	Verify a digital signature	None

AcVerifyDigestBuffer()	Verify a digital signature for a previously computed digest	None

AcEncryptInit()
AcEncryptUpdate()
AcEncryptFinal()
	Encrypt or decrypt a block of data	AES Key – R/X
		AES CCM Key – R/X
		AES GCM Key – R/X
		XTS-AES Key – R/X
		Triple-DES Key – R/X

AcGenerateKey()	Generate symmetric keys	AES Key – W AES GCM Key – W AES GCM IV – W AES CMAC Key – W XTS-AES Key – W Triple-DES Key – W KEK – W

Service	Description	CSP and Type of Access
AcGenerateKeyPair()	Generate asymmetric key pairs	RSA Private Key – W RSA Public Key – W DSA Private Key – W DSA Public Key – W ECDH Private Key – W ECDH Public Key – W ECDSA Private Key – W ECDSA Public Key – W DH Private Key – W DH Public Key – W
AcBuildKeyPairFromParams()	Generate asymmetric key pairs using specific key parameters	RSA Private Key – W RSA Public Key – W DSA Private Key – W DSA Public Key – W ECDH Private Key – W ECDH Public Key – W ECDSA Private Key – W ECDSA Public Key – W DH Private Key – W DH Public Key – W
AcExportKey()	Wrap/encrypt a key	KEK – R/X
AcImportKey()	Unwrap/decrypt an encrypted key	KEK – W/X
AcKeySize()	Return the key size for a selected Key	All Keys – R
AcKeyExchange()	Establish a shared secret using DH or ECDH	DH Private Key – R/X ECDH Private Key – R/X
AcDeriveKey()	Derive a key from an existing key’s data	TLS Session Key – W PBKDF2 DPK – W
AcReleaseHandle()	Zeroize Keys	All Keys – W
AcAceLibraryState()	Query whether library is in the soft error state	None

2.4 Finite State Model

The Module implements the finite state model detailed in submission item 4A.

2.5 Physical Security

The physical security requirements of FIPS 140-2 do not apply because the module is a software module.

2.6 Operational Environment

This module operates in a modifiable operational environment as described by the FIPS 140-2 definition. The operating systems that were 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, satisfying the FIPS 140-2 requirement for a single user mode of operation.

2.7 Cryptographic Key Management

The module supports the CSPs listed in Table 2-6.[2] [3]

Table 2-6. List of Cryptographic Keys, Cryptographic Key Components, and CSPs

CSP	CSP Type	Generation / Input	Output	Storage	Zeroization	Use
AES Key	AES 128-, 192-, or 256-bit key	Internally Generated via approved DRBG; or Input via API through GPC INT Path	Output using module API via GPC INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Encrypt and decrypt blocks of data
AES GCM Key	AES 128-, 192-, or 256-bit key	Internally Generated via approved DRBG; or Input via API through GPC INT Path	Output using module API via GPC INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Encrypt and decrypt blocks of data; Keyed Message Authentication Code
AES GCM IV	>= 96 bits of random data	Internally Generated via approved DRBG	Output using module API via GPC INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	IV input to AES GCM function
XTS-AES Key	AES XTS_128- or AES XTS_256-bit key	Internally Generated via approved DRBG; or Input via API through GPC INT Path	Output using module API via GPC INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Storage encryption or decryption

[2] The minimum number of entropy bits generated by the module for use in key generation is 256 bits.

[3] When the module generates symmetric keys or seeds used for generating asymmetric keys, unmodified DRBG output is used as the symmetric key or as the seed for generating the asymmetric keys.

CSP	CSP Type	Generation / Input	Output	Storage	Zeroization	Use
AES CMAC Key	AES 128-, 192-, or 256-bit key	Internally Generated via approved DRBG; or Input via API through GPC INT Path	Output using module API via GPC INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Keyed Message Authentication Code
Triple-DES Key	Triple-DES 168-bit key	Internally Generated via approved DRBG; or Input via API through GPC INT Path	Output using module API via GPC INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Encrypt and decrypt blocks of data
HMAC Key	112- to 512-bit HMAC Key	Internally Generated via approved DRBG; or Input via API through GPC INT Path	Output using module API via GPC INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Keyed Message Authentication Code
Key Encryption Key (KEK)	AES 128-, 192-, 256- bit key or RSA 2048-, 3072-bit key	Internally Generated via approved DRBG; or Internally Generated via PBKDF2; or Input via API through GPC INT Path	Output using module API via GPC INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Key Wrapping / Key Unwrapping
PBKDF2 DPK	112-bits of data	Internally Generated	Output using module API via GPC INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Protection of stored data
PBKDF2 Password	>= 80-bits of data	Input via API through GPC INT Path	Never	Never	Unload module; Remove Power	Application data passed as input to PBKDF2 calculation
RSA Private Key	2048- or 3072-bit RSA Private Key	Internally Generated via approved DRBG; or Input via API through GPC INT Path	Output using module API via GPC INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Signature Generation; Decryption

CSP	CSP Type	Generation / Input	Output	Storage	Zeroization	Use
DSA Private Key	224- or 256-bit DSA Private Key	Internally Generated via approved DRBG; or Input via API through GPC INT Path	Output using module API via GPC INT Path; or Output using module API via GPC INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Signature Generation
ECDSA Private Key	NIST Recommended Curves: P- 224, P-256, P-384 or P-521	Internally Generated via approved DRBG; or Input via API through GPC INT Path	Output encrypted via KEK; Output using module API via GPC INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Signature Generation
DH Private Components	224- or 256-bit Diffie Hellman Private Key	Internally Generated via approved DRBG; or Input via API through GPC INT Path	Output using module API via GPC INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Establish shared secret
ECDH Private Components	NIST Recommended Curves: P-224, P-256, P-384 or P-521	Internally generated via approved DRBG; or Input via API through GPC INT Path	Output using module API via GPC INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Establish Shared Secret
TLS RSA Premaster Secret	Random material	Input via API through GPC INT Path	Never	Not persistently stored by the module	Unload module; API call; Remove Power	Input to TLS Master Secret generation
TLS Master Secret	Random material	Internally Generated using TLS KDF	Never	Not persistently stored by the module	Unload module; API call; Remove Power	Master secret used for deriving TLS encryption keys, session key and integrity key
TLS Session Key	Shared TLS symmetric key	Internally Generated using TLS KDF	Never	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Encrypt/Decrypt communications over TLS
TLS Integrity Key	HMAC SHA-1 key	Internally Generated using TLS KDF	Never	Keys are not persistently stored by the module	N/A	Protects the integrity of data sent over TLS

CSP	CSP Type	Generation / Input	Output	Storage	Zeroization	Use
DRBG Entropy	random material	Generated by Host GPC Processor (Intel Core i7)	Never	Not persistently stored by the module	Unload module; API call; Remove Power	Entropy material for Hash_DRBG
DRBG Seed	random material	Internally generated	Never	Not stored by the module	Unload module; Remove Power	Seed material for Hash_DRBG
DRBG 'C' Value	Internal Hash_DRBG state value	Internally Generated	Never	Not persistently stored by the module	Unload module; API call; Remove Power	Used for Hash_DRBG
DRBG 'V' Value	Internal Hash_DRBG state value	Internally Generated	Never	Not persistently stored by the module	Unload module; API call; Remove Power	Used for Hash_DRBG
HMAC key for Code Integrity test	HMAC SHA256 key	Internally stored in data memory	Never	Internally stored in data memory	N/A	Integrity test HMAC key
RSA Public Key	2048- or 3072-bit RSA Public Key	Internally generated; or Input via API	Output using module API via GPC INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Digital Signature Verification; Encryption
DSA Public Key	2048- or 3072-bit DSA Public Key	Internally generated; or Input via API	Output using module API via GPC INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Digital Signature Verification
ECDSA Public Key	NIST Recommended Curves: P-224, P-256, P-384 or P-521	Internally Generated; or Input via API through GPC INT Path	Output using module API via GPC INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Digital Signature Verification
DH Public Components	2048-, or 3072-bit Diffie-Hellman Public Key	Internally Generated via approved DRBG; or Input via API through GPC INT Path	Output using module API via GPC INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Establish Shared Secret

CSP	CSP Type	Generation / Input	Output	Storage	Zeroization	Use
ECDH Public Components	NIST Recommended Curves: P-224, P-256, P-384 or P-521	Internally generated via approved DRBG; or Input via API through GPC INT Path	Output using module API via GPC INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Establish Shared Secret

2.8 EMC/EMI

The Allegro Cryptographic Engine is a software module. The only electromagnetic interference produced when it is executing is produced by the target on which the module resides and executes. FIPS 140-2 requires that the host systems on which FIPS 140-2 testing is performed meet the Federal Communications Commission (FCC) EMI and EMC requirements for business use as defined in Subpart B, Class A of FCC 47 Code of Federal Regulations Part 15. However, all systems sold in the United States must meet these applicable FCC requirements.

2.9 Self-Tests

The Allegro Cryptographic Engine performs power-up self-tests automatically each time the module is loaded into memory. Conditional self-tests are performed each time the module needs to generate a new random number or a new asymmetric key pair, or when establishing a new Diffie-Hellman Key Agreement. The module's random bit generator will perform critical function tests as needed to assure its security. While the module is performing these self-tests, all data output interfaces are inhibited.

If any self-test fail, the module's data output interfaces will be inhibited. Only control input and status output commands will be allowed to execute. To correct an on-demand self-test 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 power-up self-test error, the module must be reloaded into memory by terminating the host application and then restarting the host application. If the power-up self-test fails after restarting the host application, it will be necessary to re-install the module.

2.9.1 Power-Up Self-Tests

The Allegro Cryptographic Engine performs the following self-tests at power-up:

- Software integrity check using HMAC SHA-256 Message Authentication Code
- Algorithm Self- Tests
  - AES KAT
  - AES CMAC KAT
  - Triple-DES KAT
  - RSA Signature Generation KAT
  - RSA Signature Verification KAT
  - DSA Sign/Verify Pairwise Consistency Test
  - ECDSA Sign/Verify Pairwise Consistency Test
  - SHA-1 KAT

  - HMAC with SHA-1 KAT, SHA-224, SHA-256, SHA-384, SHA-512 KAT
  - SHA-224, SHA-256, SHA-384, SHA-512 KAT
  - SHA3-224, SHA3-256, SHA3-384, SHA3-512 KAT
  - Diffie-Hellman Primitive 'Z' Computation KAT
  - EC Diffie-Hellman Primitive 'Z' Computation KAT
  - SP 800-90A Hash_DRBG KAT

All Known Answer Tests may be called on-demand by calling the **`AcRunSelfTest()`** service.

2.9.2 Conditional Self-Tests

The Allegro Cryptographic Engine performs the following conditional self-tests when needed:

- Conditional Self Tests (CSTs)
  - RSA Pairwise Consistency Test
  - DSA Pairwise Consistency Test
  - ECDSA Pairwise Consistency Test
  - Diffie-Hellman Public Key Assurance Test
  - EC Diffie-Hellman Public Key Assurance Test
  - AES-XTS Key Validation Test
  - Repetition Count Test for NDRNG (Entropy Source)
  - Continuous Random Number Generator test for DRBG

2.9.3 Critical Functions Self-Tests

Critical function tests are performed conditionally by the module any time a random number is generated using the SP 800-90A Hash_DRBG. The SP 800-90A Hash_DRBG contains three critical functions; DRBG Instantiate, DRBG Generate and DRBG Reseed. The instantiation test is tested during power-up and any time that a new DRBG instance is created. The generation test is performed during power-up and conditionally when new random data must be generated. The reseed test is performed during power-up and conditionally when the reseed counter has reached its pre-determined maximum value and the DRBG needs to be reseeded. If any of these critical function tests fail, the module will transition to a soft error state. Follow the guidance in Section 2.9 to correct the error state.

The Allegro Cryptographic Engine performs the following critical function tests:

- SP 800-90A DRBG Instantiation Critical Function Test
- SP 800-90A DRBG Generate Critical Function Test

- SP 800-90A DRBG Reseed Critical Function Test

2.10 Design Assurance

Allegro uses Perforce Server as their configuration management tool to track the progress and design of their source code and product manuals. To ensure secure delivery of the Allegro Cryptographic Engine, Allegro places the module onto a DVD and ships the DVD via FedEx. Tracking numbers are used to track the progress of the shipment to the customer. FedEx requires the recipient of the product to sign for the package to ensure the product arrives securely to the intended recipient.

2.11 Mitigation of Other Attacks

This section is not applicable. The module does not attempt to mitigate specific attacks.

3 SECURE OPERATION

The Allegro Cryptographic Engine meets Level 1 requirements for FIPS 140-2. The sections below describe how to operate the module in FIPS-Approved mode of operation.

3.1 Initial Setup

Initial setup for the Allegro Cryptographic Engine consists of installing the host operating system, Windows 10 Professional or Linux Mint 18, creating a new user account on the Operating System and providing that user account with a password. After creating a new user account and password, the CO shall follow the CO Guidance in Section 3.2.1 to use ACE in its FIPS-Approved mode of operation.

3.1.1 Operating System Configuration

The host operating system will provide the operational environment required for the module to meet Level 1 FIPS-140 security specifications.

To run ACE in its FIPS-Approved mode of operation, a new user account shall be created on the OS. After logging into the Admin account, the CO will create a new user account following the guidelines of the OS user manual. When creating a new user, the CO shall require a password to log into the account. The CO shall refer to all administrative and guidance documents in order to create a new user account on the Operating System.

3.2 Secure Management

The Cryptographic Officer is in charge of the secure management and handling of the ACE cryptographic module. The Allegro Cryptographic Engine is shipped on a DVD and delivered via FedEx. A tracking number is provided to the CO in order to track the progress of the shipment and ensure secure delivery of the module. The CO 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 CO shall follow the steps outlined in Section 3.1.1 for proper configuration of the Operating System prior to installing the module onto the OS.

3.2.1 CO Guidance

As explained in the sections above, the CO is in charge of setting up the host Operating System and receiving the DVD containing the cryptographic module, installation guides, user guides, and other supporting documentation. The CO shall follow the installation procedures detailed in the included installation guides to properly install the Allegro Cryptographic Engine onto the operating system. The module is shipped in its FIPS-Approved mode of operation. No further configuration is needed by the CO in order to operate the module in its FIPS-Approved mode of operation. During normal operation, the User 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 FIPS mode.

3.2.1.1 Guidance for Password-Based Key Derivation Function

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.

3.2.2 User Guidance

The user shall adhere to the guidelines of this Security Policy.

The User does not have any ability to install or configure the module. Operators in the User role are able to use the services available to the User role listed in Table 2-4.

The user is responsible for reporting to the Cryptographic Officer if any irregular activity is noticed

During operation, the User may check the status of the module by attempting to run a service. If the service executes, the module is operating in FIPS mode.

An AES-GCM key may either be generated internally or provided by application code to the cryptographic module. Initialization vectors provided by application code for use with AES-GCM must be at least 96 bits long. If the initialization vector is provided by application code, 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.

4 ACRONYMS

Acronym	Definition
ACE	Allegro Cryptographic Engine
AES	Advanced Encryption System
ANSI	American National Standards Institute
API	Application Programming Interface
CBC	Cipher Block Chaining
CCM	Counter with CBC-MAC
CFB	Cipher Feedback
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CO	Cryptographic Officer
CPU	Central Processing Unit
CSEC	Communications Security Establishment Canada
CSP	Critical Security Parameter
CST	Conditional Self-Test
CTR	Counter
DES	Data Encryption Standard
DH	Diffie-Hellman
DPK	Data Protection Key
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Algorithm
DVD	Digital Video Disc
EC	Elliptic Curve
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FCC	Federal Communications Commission
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standard

Acronym	Definition
GCM	Galois/Counter Mode
GPC	General Purpose Computer
HMAC	(keyed-) Hash Message Authentication Code
INT	Internal
KAT	Known Answer Test
KEK	Key Encrypting Key
MAC	Message Authentication Code
NIST	National Institute of Standards and Technology
OEM	Original Equipment Manufacturer
OFB	Output Feedback
OS	Operating System
PKDF2	Password-Based Key Derivation Function 2
PKCS	Public Key Cryptography Standard
PSS	Probabilistic Signature Scheme
RAM	Random Access Memory
RSA	Rivest Shamir and Adelman
SHA	Secure Hash Algorithm
SP	Special Publication
SSH	Secure Shell
TLS	Transport Layer Security
Triple-DES	Triple Data Encryption Standard
USB	Universal Serial Bus
XEX	XOR-Encrypt-XOR
XOR	Exclusive Or
XTS	XEX-Based Tweaked-Codebook Mode with Ciphertext Stealing

Allegro Software Development Corporation
1740 Massachusetts Avenue
Boxborough, MA 01719

www.allegrosoft.com