Dell Australia Pty Limited, BSAFE Product Team

Dell BSAFE™ Crypto Module for C

Version 2.0

FIPS 140-3 Non-Proprietary Security Policy

Document Version: 1.4

DELL Technologies

Table of Contents

List of Tables

List of Figures

1 General

1.1 Overview

This document is a non-proprietary security policy for the BSAFE Crypto Module from Dell Australia Pty Limited, BSAFE Product Team. This document contains the security rules under which the module must operate and describes how this module meets the requirements of FIPS 140-3 overall Security Level 1.

This Non-Proprietary Security Policy may be freely reproduced and distributed, but only in its entirety and without modification.

Terminology

In this document, the Dell BSAFE™ Crypto Module for C is also referred to as:

- The Cryptographic Module
- The BSAFE Crypto Module
- The module

1.2 Security Levels

BSAFE Crypto Module is validated with an overall FIPS 140-3 Security Level 1. Security levels for individual areas are shown in the following table:

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	1
	Overall Level	1

Table 1: Security Levels

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

Dell BSAFE™ Crypto Module for C is a software module intended to be used as part of a software system, providing cryptographic services to that system.

The module is provided as a static library in Executable and Linkable Format (ELF), built for the Intel® x86_64 (64-bit) architecture. It follows the standard x86_64 calling conventions and provides a documented set of functions that can be called from user software. It is intended to be linked directly into the user software system.

Module Type: Software
Module Embodiment: Multi-Chip Standalone

Module Characteristics:

Cryptographic Boundary:

BSAFE Crypto Module is classified as a multi-chip standalone software cryptographic module for the purposes of FIPS 140-3. As such, it is tested on specific operating systems and computer platforms. The cryptographic boundary includes the module running on selected platforms running selected operating systems. The module is packaged as a library with an object file containing the module's entire executable code. The module relies on the physical security provided by the host computer in which it runs.

The cryptographic boundary of the module is the linked object file within the final application.

The underlying cryptographic boundary to the module is the API, documented in the Dell BSAFE Crypto Module Developers Guide. The module accepts Control Input through the API calls. Data Input and Output are provided in the variables passed with the API calls. Status Output is provided through the returns and error codes documented for each call. This is illustrated in Figure 1 BSAFE Crypto Module Logical Interfaces.

Tested Operational Environment's Physical Perimeter (TOEPP):

The TOEPP of the module is the general-purpose computer, which encloses the hardware running the module. The physical interfaces for the module are the physical interfaces of the computer running the module, such as the keyboard, monitor and network interface.

The following diagram illustrates the module's TOEPP and cryptographic boundary:

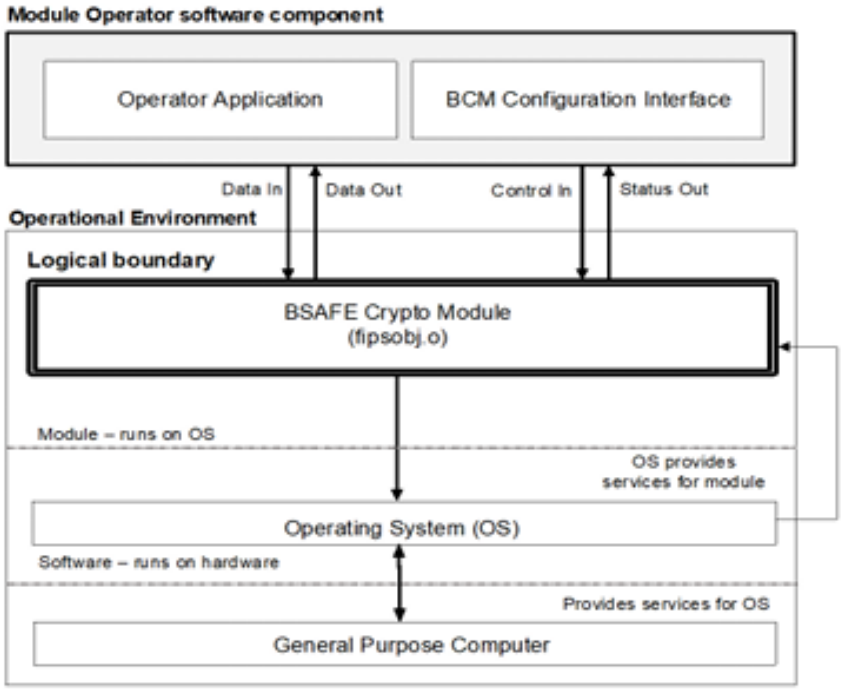

Figure 1: Block Diagram

2.2 Tested and Vendor Affirmed Module Version and Identification

Tested Module Identification – Hardware:

N/A for this module.

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

Package or File Name	Software/ Firmware Version	Features	Integrity Test
libdellbcm.a	2.0	linux-x64-gcc	HMAC-SHA2-256
libdellbcm.a	2.0	symmk-x64-gcc6	HMAC-SHA2-256

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

Tested Module Identification – Hybrid Disjoint Hardware:

N/A for this module.

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon Gold 6254	Yes		2.0 2.0
PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon Gold 6254	No		2.0 2.0
PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon Gold 8280L	No		2.0 2.0
PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon Gold 8280L	Yes		2.0 2.0
PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon® Gold 5218	No		2.0 2.0
PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon® Gold 5218	Yes		2.0 2.0

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

The PAA referred to in the table above is AES-NI.

Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform
PowerMaxOS 10	PowerMax storage array compute node (Intel Xeon Gold 6240L)
SUSE® Linux Enterprise Server 15 SP2	Intel x86_64 (64-bit)

Table 4: Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid

2.3 Excluded Components

There are no components within the cryptographic boundary that are excluded from the module.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
BCM_MODE_FIPS	The module allows the cryptographic algorithms listed in FIPS 140-3 Approved Algorithms.	Approved	BCM_ctx_is_fips(NULL) returns 1
BCM_MODE_NON_FIPS	The module allows all available cryptographic algorithms.	Non-Approved	BCM_ctx_is_fips(NULL) returns 0

Table 5: Modes List and Description

The module can operate in the Approved mode or the Non-Approved mode. The mode selected affects which algorithms are available for use. In:

- Approved mode (BCM_MODE_FIPS), the module allows the cryptographic algorithms listed in FIPS 140-3 Approved Algorithms. Non-Approved algorithms are not allowed in the approved mode of operation.

- Non-Approved mode (BCM_MODE_NON_FIPS), the module allows all available cryptographic algorithms.

Approved mode is also referred to as FIPS 140-3 mode in the product documentation.

In each mode of operation, the complete set of services listed in this Security Policy are available to the Crypto Officer.

Note: Critical Security Parameters must not be shared between modes. For example, a key generated in the Approved mode of operation must not be exported and then imported to an application running in the Non-Approved mode.

2.4.1 Module Mode Configuration

The module operator must provide a definition of the configuration function `BCM_get_config(BCM_CONFIG *config)` that is called by the module during startup.

The Module mode is set in the operator's function by assigning a value to config->mode.

To start the module in the Approved mode of operation, the operator's configuration function should assign the value BCM_MODE_FIPS to the mode member of the configuration structure:

```
BCM_STATUS BCM_get_config(BCM_CONFIG *config)
{
      // Start the module in the Approved mode of operation
      config->mode = BCM_MODE_FIPS;
      // ...
      return BCM_OK;
}
```

To start the module in the Non-Approved mode of operation, the operator's configuration function must assign the value BCM_MODE_NON_FIPS to the mode member of the configuration structure:

```
BCM_STATUS BCM_get_config(BCM_CONFIG *config)
{
      // Start the module in the Non-Approved mode of operation
      config->mode = BCM_MODE_NON_FIPS;
      // ...
      return BCM_OK;
}
```

Note: The default value of the mode member is set to BCM_MODE_FIPS. Therefore, if the operator's configuration function does not set the mode, the module starts in the Approved mode of operation.

Once the module is initialized and the pre-operational self-tests (POST) have completed successfully, the overall operating mode of the module can be changed by calling the `BCM_module_configure()` API.

2.4.2 Approved Mode Indicator

The module uses an approved mode indicator combined with a return status code from an approved security service to indicate the use of an approved service.

Approved security services that operate on a BCM_CTX use the API `BCM_ctx_is_fips()` with a return code of 1 as an indicator that the context is operating in the approved mode.

Approved security services that operate on a BCM_KEY use the API `BCM_key_is_fips()` with a return code of 1 as an indicator that the key is operating in the approved mode.

2.5 Algorithms

Approved Algorithms:

The module implements the following approved algorithms that have been tested under CAVP. Additional algorithms have also been CAVP-tested but are not claimed by this module.

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A1204	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A1204	Key Length - 128, 192, 256 Tag Length - 112, 128, 32, 48, 64, 80, 96 IV Length - IV Length: 56-104 Increment 8 Payload Length - Payload Length: 0-256 Increment 8 AAD Length - AAD Length: 0-524288 Increment 8	SP 800-38C
AES-CTR	A1204	Direction - Encrypt Key Length - 128, 192, 256 Payload Length - Payload Length: 8-128 Increment 8 Supports Counter larger than maximum value - No Incremental Counter - No Counter Tests Performed - No	SP 800-38A
AES-ECB	A1204	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A1204	Direction - Decrypt, Encrypt IV Generation - Internal IV Generation Mode - 8.2.2 Key Length - 128, 192, 256 Tag Length - 128 IV Length - IV Length: 96 Payload Length - Payload Length: 0-1024 Increment 8 AAD Length - AAD Length: 0-1024 Increment 8	SP 800-38D

Algorithm	CAVP Cert	Properties	Reference
AES-KW	A1204	Direction - Decrypt, Encrypt Cipher - Cipher Key Length - 128, 192, 256 Payload Length - Payload Length: 128-4096 Increment 64	SP 800-38F
AES-KWP	A1204	Direction - Decrypt, Encrypt Cipher - Cipher Key Length - 128, 192, 256 Payload Length - Payload Length: 808	SP 800-38F
AES-XTS Testing Revision 2.0	A1204	Direction - Decrypt, Encrypt Key Length - 128, 256 Payload Length - Payload Length: 128-2048 Increment 8 Tweak Mode - Number Data Unit Length Matches Payload Length - Yes	SP 800-38E
HMAC DRBG	A1204	Prediction Resistance - No Supports Reseed - Yes Mode - SHA2-512 Entropy Input - Entropy Input: 256 Nonce - Nonce: 128, 512 Personalization String Length - Personalization String Length: 0, 1024, Personalization String Length: 0, 1048576 Additional Input - Additional Input: 0, 1024, Additional Input: 0, 1048576 Returned Bits - 512	SP 800-90A Rev. 1
HMAC-SHA-1	A1204	MAC - MAC: 80, 96, 128, 160 Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A1204	MAC - MAC: 128, 192, 256 Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A1204	MAC - MAC: 192, 256, 320, 384 Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A1204	MAC - MAC: 256, 320, 384, 448, 512 Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
KAS-IFC-SSC	A1204	Modulo - 2048, 3072, 4096, 6144, 8192 Key Generation Methods - rsakpg2-basic Scheme - KAS1 - KAS Role - initiator, responder	SP 800-56A Rev. 3
PBKDF	A1204	Iteration Count - Iteration Count: 1-100000 Increment 1 HMAC Algorithm - SHA-1, SHA2-256, SHA2-384, SHA2-512 Password Length - Password Length: 8-128 Increment 1	SP 800-132

Algorithm	CAVP Cert	Properties	Reference
		Salt Length - Salt Length: 128-4096 Increment 8 Key Data Length - Key Data Length: 112-4096 Increment 8	
RSA Decryption Primitive (CVL)	A1204	Modulus Length - 2048	FIPS 186-4
RSA KeyGen (FIPS186-4)	A1204	Key Generation Mode - B.3.6 Modulo - 2048, 3072, 4096 Primality Tests - Table C.3 Info Generated By Server - No, Yes Public Exponent Mode - Random Private Key Format - Standard	FIPS 186-4
RSA SigGen (FIPS186-4)	A1204	Signature Type - ANSI X9.31, PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096 Hash Pair - Hash Algorithm - SHA2-224	FIPS 186-4
RSA SigVer (FIPS186-4)	A1204	Signature Type - ANSI X9.31, PKCS 1.5, PKCSPSS Modulo - 1024, 2048, 3072, 4096 Hash Pair - Hash Algorithm - SHA-1 Public Exponent Mode - Fixed, Random Fixed Public Exponent - 010001	FIPS 186-4
SHA-1	A1204	Message Length - Message Length: 0-51200 Increment 8 Function - SHA1	FIPS 180-4
SHA2-224	A1204	Message Length - Message Length: 0-51200 Increment 8 Function - SHA2	FIPS 180-4
SHA2-256	A1204	Message Length - Message Length: 0-51200 Increment 8 Function - SHA2	FIPS 180-4
SHA2-384	A1204	Message Length - Message Length: 0-65536 Increment 8 Function - SHA2	FIPS 180-4
SHA2-512	A1204	Message Length - Message Length: 0-65536 Increment 8 Function - SHA2	FIPS 180-4
SHA2-512/224	A1204	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512/256	A1204	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4

Table 6: Approved Algorithms

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
CKG-XTS	Key Type:Symmetric		NIST SP 800-133r2 and IG D.H: Section 6.3, approved method 1. Applicable to AES-XTS compliant to IG C.I because Key_1 and Key_2 are concatenated prior to usage
CKG-4	Key Type:Asymmetric and Symmetric		NIST SP800-133r2 Section 4 Example 1

Table 7: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms with No Security Claimed:

N/A for this module.

Non-Approved, Not Allowed Algorithms:

Name	Use and Function
MD5	Message Digesting
TDES in ECB and CBC modes	Symmetric encryption

Table 8: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
AES-XTS	BC-UnAuth	AES XTS Key generated to comply with the approved key generation guidelines of NIST SP 800-133rev2, Section 6.3		CKG-XTS: () AES-ECB: (A1204) AES-XTS Testing Revision 2.0: (A1204)
Authenticated Decryption	BC-AuthDecrypt	Decryption using authenticated modes		AES-CTR: (A1204) AES-CCM: (A1204) AES-GCM: (A1204)
Authenticated Encryption	BC-AuthEncrypt	Encryption using authenticated modes		AES-CTR: (A1204) AES-CCM: (A1204) AES-GCM: (A1204)

Name	Type	Description	Properties	Algorithms
Decryption	BC-UnAuthDecrypt	Decryption using unauthenticated modes		AES-CBC: (A1204) AES-CTR: (A1204) AES-ECB: (A1204)
Encryption	BC-UnAuthEncrypt	Encryption using unauthenticated modes		AES-CBC: (A1204) AES-CTR: (A1204) AES-ECB: (A1204)
Entropy	ENT-NP	Non-Physical Entropy source		
Entropy Conditioning	ENT-Cond	HMAC used for Entropy Conditioning		HMAC-SHA2-256: (A1204) SHA2-256: (A1204)
HMAC DRBG	DRBG	Random Number Generation		CKG-4: () Key Type: Asymmetric and Symmetric HMAC DRBG: (A1204) HMAC-SHA2-512: (A1204) SHA2-512: (A1204)
Key Derivation	PBKDF	Key Derivation		PBKDF: (A1204) HMAC-SHA-1: (A1204) HMAC-SHA2-256: (A1204) HMAC-SHA2-384: (A1204) HMAC-SHA2-512: (A1204)
Key Generation	AsymKeyPair-KeyGen	Key Generation		CKG-4: () Key Type: Asymmetric and Symmetric RSA KeyGen (FIPS186-4): (A1204)
Key Unwrap	BC-Auth	Key Unwrapping		AES-CBC: (A1204) AES-ECB: (A1204) AES-KW: (A1204) AES-KWP: (A1204)
Key Wrap	BC-Auth	Key Wrapping		AES-CBC: (A1204) AES-ECB: (A1204) AES-KW: (A1204) AES-KWP: (A1204)

Name	Type	Description	Properties	Algorithms
MAC Generation	MAC	MAC Generation		HMAC-SHA-1: (A1204) HMAC-SHA2-256: (A1204) HMAC-SHA2-384: (A1204) HMAC-SHA2-512: (A1204)
MAC Verification	MAC	MAC Verification		HMAC-SHA-1: (A1204) HMAC-SHA2-256: (A1204) HMAC-SHA2-384: (A1204) HMAC-SHA2-512: (A1204)
Message Digest	SHA	Message Digest		SHA-1: (A1204) SHA2-224: (A1204) SHA2-256: (A1204) SHA2-384: (A1204) SHA2-512: (A1204) SHA2-512/224: (A1204) SHA2-512/256: (A1204)
RSA SigGen	DigSig-SigGen	Digital Signature Generation using RSA		RSA SigGen (FIPS186-4): (A1204) SHA2-224: (A1204) SHA2-256: (A1204) SHA2-384: (A1204) SHA2-512: (A1204) SHA2-512/224: (A1204) SHA2-512/256: (A1204)

Name	Type	Description	Properties	Algorithms
RSA SigVer	DigSig-SigVer	Digital Signature Verification using RSA		RSA SigVer (FIPS186-4): (A1204) SHA2-224: (A1204) SHA2-256: (A1204) SHA2-384: (A1204) SHA2-512: (A1204) SHA2-512/224: (A1204) SHA2-512/256: (A1204)
Shared Secret Computation	KAS-SSC	Computation of shared secrets	Caveat:KAS-IFC-SSC provides between 112 and 200 bits of encryption strength	KAS-IFC-SSC: (A1204) RSA Decryption Primitive: (A1204)

Table 9: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 Symmetric Key Operations

GCM Mode Ciphers

When using GCM feedback mode for symmetric encryption, the authentication tag length and authenticated data length may be specified as input parameters, but the IV must not be specified. For compliance with IG C.H scenario 2, it must be generated internally. The generated IV is fully random, generated by the module's approved DRBG, with a default length of 96 bits. No special considerations are required provided the system has sufficient entropy.

XTS Mode Ciphers

AES-XTS is only approved for storage purposes.

The data encryption key and tweak key components of the double-length XTS key must be checked to ensure they are different. This check is performed automatically by the module.

Triple DES

TDES is not available as an approved algorithm in the Approved Mode of Operation. When TDES is used in Non-Approved mode, the amount of data that can be encrypted is restricted to 2^{16} 64-bit blocks.

Hashing/Message Digest Operations

SHA-1 is acceptable for non-digital signature applications. SHA-1 is not allowed in the Approved Mode of Operation for digital signature generation. For legacy use only, SHA-1 is allowed in the Approved Mode of Operation for the verification of existing digital signatures.

2.7.2 Asymmetric Key Operations

In the following, *Protect* refers to cryptographically protecting data for later use, e.g. signing, encrypting or wrapping. *Process* refers to processing previously protected data, e.g. verifying, decrypting or unwrapping.

Purpose	RSA modulus length	Note
Protect and Process	2048, 3072, 4096	Sizes approved in FIPS 186-4 and FIPS 140-3 IG. (CAVP validated)
Process only	1024	May be used for verification only.

Table 10: Asymmetric Key Operations

2.7.3 Digital Signature Operations

Keys used for digital signature generation and verification shall not be used for any other purpose. The module generates or loads keys with a particular purpose that is either signing or encryption. The same purpose must always be used for a given key when exported and loaded into the module again.

The length of an RSA key pair for digital signature generation must be greater than or equal to 2048 bits. For digital signature verification, the length must be greater than or equal to 2048 bits, however 1024 bits is allowed for legacy-use only.

RSA keys must pass validation before use. Keys generated by the module will pass validation. For RSA PKCS #1 PSS, the size relationship between the hash function output block length (hLen) and the length of the salt (sLen) shall be 0 <= slen <= hLen.

Verification of signatures with a SHA-1 digest is allowed for legacy use.

The security strength of both the key and the digest functions shall be chosen to meet or exceed the required security strength for the digital signature. The security strength of the digest function should be stronger or the same as that of the key.

2.7.4 Message Authentication Code Operations

HMACs

The key length for an HMAC generation or verification must be between 112 and 256 bits, inclusive. For HMAC verification, a key length greater than or equal to 80 and less than 112 is allowed for legacy-use.

2.7.5 Key Derivation Function Operations

Password-based Key Derivation

Keys generated using PBKDF2 shall only be used in data storage applications.

The minimum password length is 14 characters, which has a strength of approximately 112 bits, assuming a randomly selected password using the extended ASCII printable character set is used.

For random passwords, that is, a string of characters from a given set of characters in which each character is equally likely to be selected, the strength of the password is given by

$$S = L * (\log N / \log 2)$$

where:

- N is the number of possible characters. For example:
  - For the ASCII printable character set N = 95
  - For the extended ASCII printable character set N = 218.
- L is the number of characters.

A password of strength S can be guessed at random with the probability of 1 in 2^S.

The minimum length of the randomly-generated portion of the salt is 16 bytes.

The iteration count is as large as possible, with a minimum of 10,000 iterations recommended.

The derived key size can range from 1 byte to a maximum of $(2^{32} - 1) * b$, where b is the digest size of the message digest function in bytes.
Derived keys can be used as specified in SP 800-132, Section 5.4, option 1a

2.7.6 Key Transport Schemes Operations

Key Wrapping using AES-KW/KWP

The key establishment methodology provides between 128 and 256 bits (inclusive) of encryption strength.

The security strength of the key encryption key must be greater than or equal to the security strength of the key being wrapped.

The module does not establish Shared Secret Parameters (SSPs) using an approved Key Transport Scheme (KTS). However, it does offer approved authenticated encryption algorithms that can be used by an external operator/application as part of an approved KTS.

2.7.7 Key Validation Operations

Asymmetric keys are validated as they enter the module for use in processing existing data. Before the keys are used to protect data, they must be validated. The module provides services for the validation of RSA keys.

2.7.8 Key Agreement Operations (Underlying Primitives Only)

The module does not establish SSPs using an approved key agreement scheme (KAS). However, it does offer some or all the underlying KAS cryptographic functionality to be used by an external operator/application as part of an approved KAS.

2.8 RBG and Entropy

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
BSAFE Crypto Module Entropy Source Version 2.0	Non-Physical	PowerMaxOS 10 on Intel Xeon Gold 6254, PowerMaxOS 10 on Intel Xeon Gold 8280L, PowerMaxOS 10 on Intel Xeon Gold 5218	256 bits	255 bits	HMAC-SHA2-256 (A1204)

Table 11: Entropy Sources

2.8.1 Deterministic Random Bit Generator

BSAFE Crypto Module provides the following approved DRBGs for use in both Approved and Non-Approved modes:

DRBG	Entropy Obtained (bits)
HMAC DRBG with SHA2-512	255

Table 12: Deterministic Random Bit Generator - Entropy

Use	Details
RSA key-pair generation	Prime generation, and Miller-Rabin prime number testing[1] Blinding of random values
RSA key validation	Prime recovery testing[2]
Symmetric key generation	Direct generation of symmetric keys
Initialization vector generation	Direct generation of IVs for symmetric encryption
RSA PKCS #1 PSS signing	Generation of random value for message encoding

Table 13: Deterministic Random Bit Generator – Use and Details

[1]All seeds for asymmetric key generation are generated using the direct output of the approved DRBG.
[2]For details refer to NIST SP 900-56B Rev. 2, Appendix C.

2.8.2 Entropy sources

BSAFE Crypto Module provides an entropy source that is internal to the module. This entropy source generates entropy that is used to seed the Approved RBGs

The entropy source is provided as an ENT (NP) that is compliant with SP 800-90B and is estimated to provide a min entropy of 7.96875 bits per byte.

Entropy Sources	Sources Minimum Number of Bits of Entropy	Details
Execution time jitter	255	Instantiation of HMAC DRBG
Execution time jitter	DRBG instantiation bits	Application calls `BCM_random_seed()`

Table 14: Entropy sources

The BCM_CTX object manages an approved DRBG and an entropy source. The first call to a random number generation service using the BCM_CTX object instantiates the entropy source and the DRBG. At instantiation, the DRBG issues a GET call to the entropy source for the number of bits of entropy required to meet the security strength of the DRBG.

A call to the `BCM_random_seed()` API with the BCM_CTX object resets the DRBG seed. The DRBG issues a GET call to the entropy source for the number of bits of entropy data equivalent to the security strength of the DRBG.

A call to the `BCM_secure_random_bytes()` API with the BCM_CTX object adds a fixed number of additional input to the DRBG state before the DRBG generates output. The DRBG issues a GET call to the entropy source for 64 bits of additional input.

The entropy is collected from the jitter in the CPU execution time for performing an HMAC over the state and previous noise sample. By collecting multiple jitter samples, a bit stream that meets the statistical measurements which indicate a bit stream is random, is produced and whitened.

If the bits of entropy are not available on a GET call then the entropy source generates an error status code that is returned to the application by the random number generation service.

2.9 Key Generation

When using an approved DRBG to generate keys, the security strength of the DRBG must be at least as great as the security strength of the key being generated. For details about the comparable security strengths of symmetric block ciphers and asymmetric key algorithms refer to Table 2 of SP 800-57 Part 1 Rev. 5.

The module's default DRBG provides a security strength as great as that of any supported keys.

2.10 Key Establishment

The module does not implement an approved key transport scheme, however, it does offer approved authenticated algorithms that can be used by an external operator/application as part of an approved KTS.

The module provides RSA shared secret computation that is conformant to SP 800-56Br2 in alignment with IG D.F scenario 1 (path 1) via the Key Agreement service. The module supports the KAS1 scheme.

2.11 Industry Protocols

N/A. The module does not implement any industry protocols.

3 Cryptographic Module Interfaces

BSAFE Crypto Module is a software module that provides APIs only as logical interfaces. Physical ports and interfaces are not provided by the module.

The module conforms to the FIPS 140-3 Security Level 1 requirements for Cryptographic Module Interfaces and does not support a Trusted Channel Interface.

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	Service inputs
N/A	Data Output	Service outputs
N/A	Control Input	Configuration parameters for the API BCM_module_configure() which sets the mode of operation.
N/A	Status Output	Mode of operation indicator from the API BCM_ctx_is_fips(). The state of the module from the API BCM_module_state().

Table 15: Ports and Interfaces

4 Roles, Services, and Authentication

BSAFE Crypto Module meets all FIPS 140-3 Security Level 1 requirements for Roles, Services and Authentication, implementing only the Crypto Officer role. As allowed by FIPS 140-3, the module does not support identification or authentication of this role. There is no maintenance role, cryptographic bypass capability, or self-initiated cryptographic output. The module does not allow concurrent operators.

4.1 Authentication Methods

N/A for this module.

The module does not implement authentication. The Crypto Officer Role is implicitly assumed once the module is loaded and cleared once the module is unloaded.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	Crypto Officer	None

Table 16: Roles

The module supports a single role, denoted as Crypto Officer. This role is:

- Responsible for installing and loading the module and has access to all services provided by the module.
- Assumed automatically once the module has been loaded and the POST have run successfully. The POST are automatically run when the module is first loaded. They can be run manually at any time by calling `BCM_module_selftest()`.

4.3 Approved Services

The following convention is used to specify access rights to SSPs:

- G - Generate: The module generates or derives the SSP.
- R - Read: The module exports the SSP.
- W - Write: The SSP is imported or updated
- E - Execute: The module uses the SSP in performing a cryptographic operation.
- Z - Zeroize: The module zeroizes the SSP.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Authenticated decryption	Perform authenticated decryption	BCM_key_is_fips() return value	Ciphertext, MAC, key	Plaintext, Status	Authenticated Decryption	Crypto Officer - AES Key: E - AES-GCM IV: G,E
Authenticated encryption	Perform authenticated encryption	BCM_key_is_fips() return value	Plaintext, key	Ciphertext, Status	Authenticated Encryption	Crypto Officer - AES Key: E - AES-GCM IV: G,E
Decryption	Perform unauthenticated decryption	BCM_key_is_fips() return value	Ciphertext, key	Plaintext, Status	AES-XTS Decryption	Crypto Officer - AES Key: E - AES-XTS: E
Digital Signature Generation	Perform Digital Signature Generation	BCM_key_is_fips() return value	Message Digest	Signature, Status	RSA SigGen	Crypto Officer - RSA Private Key: W,E
Digital Signature Verification	Perform Digital Signature Verification	BCM_key_is_fips() return value	Message Digest, Signature	Verification Status, Status	RSA SigVer	Crypto Officer - RSA

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Public Key: W,E
DRBG Initialization	Prepare for random number or key generation	BCM_ctx_is_fips() return value	Entropy	n/a	Entropy HMAC DRBG	Crypto Officer - HMAC DRBG Key: G - HMAC DRBG Seed: G - HMAC DRBG V: G
Encryption	Perform unauthenticated encryption	BCM_key_is_fips() return value	Plaintext, key	Ciphertext, Status	AES-XTS Encryption	Crypto Officer - AES Key: E - AES-XTS: E
Key Agreement	Perform explicit validation of public and private keys	BCM_key_is_fips() return value	Keys	Validation Status, Status	Shared Secret Computation	Crypto Officer - RSA Private Key: R - RSA Public Key: R
Key Derivation	Perform Key Derivation	BCM_ctx_is_fips() return value	Secret	Key text, Status	Key Derivation	Crypto Officer - PBKDF derived key: G - PBKDF Password : R
Key Export	Perform Key Export	BCM_key_is_fips() return value	Key	Key text, Status	None	Crypto Officer - AES Key: R - RSA Private Key: R - RSA Public Key: R

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Key Generation	Perform Key Generation	BCM_ctx_is_fips() return value	Key Parameters	Key text, Status	Entropy Entropy Conditioning HMAC DRBG Key Generation	Crypto Officer - AES Key: G - Entropy input: W - HMAC DRBG Key: G,E - HMAC DRBG Seed: G,E - HMAC DRBG V: G,E - RSA Private Key: G - RSA Public Key: G
Key Import	Perform Key Import	BCM_ctx_is_fips() return value	Key text	Status	None	Crypto Officer - AES Key: W - RSA Private Key: W - RSA Public Key: W
Key Unwrap	Perform Key Unwrap	BCM_key_is_fips() return value	Wrapped key text	Key text, Status	Key Unwrap	Crypto Officer - AES Key Wrap Key: W,E
Key Wrap	Perform Key Wrap	BCM_key_is_fips() return value	Key	Wrapped key text, Status	Key Wrap	Crypto Officer - AES Key Wrap Key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Key Zeroization	Perform Key Zeroization	N/A	N/A	N/A	None	Crypto Officer - AES Key: Z - AES Key Wrap Key: Z - HMAC DRBG Key: Z - HMAC Keys: Z - PBKDF derived key: Z - RSA Private Key: Z - RSA Public Key: Z
MAC Generation	Perform MAC Generation	BCM_ctx_is_fips() return value	Secret, Message, key	MAC, Status	MAC Generation	Crypto Officer - HMAC Keys: W,E,Z
MAC Verification	Perform MAC Verification	BCM_ctx_is_fips() return value	Secret, Message, MAC	Verify Status, Status	MAC Verification	Crypto Officer - HMAC Keys: W,E,Z
Message Digest	Perform Message Digest Operation	BCM_ctx_is_fips() return value	Message	Message Digest, Status	Message Digest	Crypto Officer
Random Number Generation	Perform Random Number generation	BCM_ctx_is_fips() return value	Entropy	Random Bytes, Status	Entropy Entropy Conditioning HMAC DRBG	Crypto Officer - Entropy input: W - HMAC DRBG Key: G,E - HMAC

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						DRBG Seed: G,E - HMAC DRBG V: G,E
Self-test	Perform Self-test	BCM_ctx_is_fips () return value	Command	Status	None	Crypto Officer
Shared secret computation	Compute a shared secret	BCM_key_is_fips () return value	Peer public key text	Shared Secret, Status	Shared Secret Computation	Crypto Officer
Show Status	Show module status	N/A	API Call	Module Status	None	Crypto Officer
Show Version	Show module name and version	N/A	API Call	Module version, Status	None	Crypto Officer

Table 17: Approved Services

For each service, the FIPS indicator is obtained by checking the service Status and the Approved of the Context or Key:

- `BCM_ctx_is_fips()` returns the Approved Mode of the context.
- `BCM_key_is_fips()` returns the Approved Mode of the key.

For information about individual functions that implement each service, see the Dell BSAFE Crypto Module Developers Guide.

4.4 Non-Approved Services

The following is a list of Non-Approved services provided by the module:

Name	Description	Algorithms	Role
Message Digest	Perform Message Digest	MD5	Crypto Officer
Symmetric Decryption	Perform Symmetric Decryption	TDES in ECB and CBC modes	Crypto Officer
Symmetric Encryption	Perform Symmetric Encryption	TDES in ECB and CBC modes	Crypto Officer

Table 18: Non-Approved Services

4.5 External Software/Firmware Loaded

Not Applicable. The module does not support the loading of external software/firmware.

5 Software/Firmware Security

This section covers integrity measures to demonstrate protection of the software component of the BSAFE Crypto Module, which is the whole of the module. The integrity tests used are described in detail in Self-tests.

5.1 Integrity Techniques

The module is an object file (libdellbcm.a). When the module object file is created, a MAC is calculated over the executable code and static data sections, with the resulting integrity block embedded into the object file.

During the Integrity Test when the module is loaded, a MAC is again calculated over the executable code and static data. The resulting MAC is compared to the integrity block calculated when the module was created.

If the MAC differs, the Integrity Test fails, the POST fails, the module enters the BCM_MODULE_STATE_INTEGRITY_FAILED state and cannot be used for any cryptographic operation. Any attempted operation will return the BCM_ERROR_FIPS_INTEGRITY_FAILURE error indicator.

The only way to clear this error condition is to unload the module and load it again.

The pre-operational software integrity test uses HMAC-SHA2-256 (A1204).

5.2 Initiate on Demand

The module provides the `BCM_module_selftest()` API for on-demand integrity testing.

5.3 Additional Information

The module is built as a single ELF object file with an embedded FIPS 140-3 integrity signature. The ELF object file exports symbols for the operations it supports.

6 Operational Environment

6.1 Operational Environment Type and Requirements

BSAFE Crypto Module is FIPS 140-3 validated to operate in a modifiable environment.

The module is provided for operating systems running on a general-purpose computer platform based on an Intel CPU.

Type of Operational Environment: Modifiable

How Requirements are Satisfied:

Each instance of the module that is loaded within an operating system maintains its own instance of internal SSPs. Any additional SSPs loaded into a given instance of the module are not available to other instances.

The supported operating environments provide process isolation, with resource and memory protection. Each instance of the module is isolated from others such that SSPs can only be accessed or modified in the module to which they belong.

BSAFE Crypto Module does not spawn additional processes.

6.2 Configuration Settings and Restrictions

The module runs on a general-purpose computer running one of the operating environments listed in Tested Operational Environments and Vendor Affirmed Operational Environments. Each supported operating environment manages its own processes and memory in a logically separated manner. The process management setting is not configurable on the supported operating environments.

7 Physical Security

BSAFE Crypto Module is classified as a multi-chip standalone cryptographic module. The module is comprised of software only, validated at FIPS 140-3 Security Level 1, and does not claim any physical security.

8 Non-Invasive Security

The module does not implement any non-invasive mitigation techniques.

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
VM	Volatile Memory	Dynamic

Table 19: Storage Areas

Protection of the SSPs in volatile memory is provided by the operating environment which isolates the memory of separate processes.

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
From App	Calling Application	VM	Plaintext	Manual	Electronic	
To App	VM	Calling Application	Plaintext	Manual	Electronic	

Table 20: SSP Input-Output Methods

Sensitive security parameters are input and output using the module APIs. No security function or algorithm is used to transport the data.

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Explicit	SSPs are zeroized when the associated key or cryptographic object is deleted by the application. Zeroization is performed by overwriting the data with zero valued bytes	BCM provides BCM_KEY objects as opaque handles for cryptographic keys held in memory by the module. Any SSPs associated with a BCM_KEY persist in memory for as long as the key object exists. BCM_KEY objects are created explicitly by the operator, used by the operator, and then deleted explicitly by the operator. When a BCM_KEY is deleted, any SSPs associated with it are immediately zeroized.	API call to delete object
Immediate	Temporary SSPs are zeroized immediately after use. Zeroization is performed by overwriting the data with zero valued bytes	Intermediate values are zeroized at the end of each calculation function, before the function returns. This avoids leaving SSPs in unallocated stack or heap memory and minimizes the length of time such SSPs are stored. When this occurs, the SSPs are immediately zeroized.	N/A
Implicit	SSPs are zeroized when the cryptographic object is deleted by the module. Zeroization is performed by overwriting the data with zero valued bytes	Implicitly zeroized SSPs correspond to the internal state of any persistent DRBG maintained by a BCM_CTX. When needed, a BCM_CTX creates a DRBG and keeps it available for reuse across multiple operations. As a result, these SSPs persist for as long as the associated BCM_CTX exists. The default/global BCM_CTX is automatically deleted when the module is unloaded, and any DRBG it contains is deleted at that time. This deletion implicitly zeroizes the SSPs associated with the DRBG's internal state. All other BCM_CTX instances are deleted when the operator explicitly calls BCM_CTX_delete(). When this occurs, the SSPs	N/A

Zeroization Method	Description	Rationale	Operator Initiation
		for the DRBG's internal state are immediately zeroized.	

Table 21: SSP Zeroization Methods

The module encapsulates symmetric and asymmetric keys as BCM_KEY objects. For multi-part cryptographic operations, the module defines several object types to encapsulate the intermediate state of the operation.

Examples are BCM_CIPHER objects for symmetric ciphers and BCM_MAC objects for message authentication codes. These objects are created explicitly by the user with function calls to the module.

The module defines a BCM_CTX object which can contain SSPs in the form of internal random number generator (RNG) state and associated entropy state. A single BCM_CTX is created automatically when the module starts and is retained by the module as the default context. BCM_CTX objects can be created explicitly by the user with function calls to the module.

To zeroize all unprotected SSPs and key components, perform the following procedure:

1. For each object delete all:
   a) BCM_CIPHER cryptographic objects with a call to `BCM_cipher_delete()`.
   b) BCM_MAC cryptographic objects with a call to `BCM_mac_delete()`.
   c) BCM_DIGEST cryptographic objects with a call to `BCM_digest_delete()`.
   d) BCM_KEY objects with a call to `BCM_key_delete()`.
   e) BCM_CTX objects with a call to `BCM_ctx_delete()`.
2. Delete the default context created at startup. To do this, unload the module or call `BCM_module_unload()`.

9.4 SSPs

The following tables list the SSPs present in the module and details of how they are used and accessed.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES Key	Encryption and Decryption	128, 192, 256 bits - 128, 192, 256 bits	Symmetric Key - CSP	Key Generation		Authenticated Decryption Authenticated Encryption Decryption Encryption
AES Key Wrap Key	AES Key Wrap	128, 192, 256 bits - 128, 192, 256 bits	Symmetric Key - CSP	Key Generation		Key Wrap Key Unwrap

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES-GCM IV	AES-GCM Initialization Vector	96 bits - 96 bits	Initialization Vector - CSP	HMAC DRBG		HMAC DRBG
AES-XTS	Encryption and Decryption	128, 256 bits - 128, 256 bits	Symmetric Key - CSP	Key Generation		AES-XTS Decryption Encryption
Entropy input	Entropy collection and whitening	512 bits - 256 bits	Entropy - CSP	Entropy Entropy Conditioning		HMAC DRBG
HMAC DRBG Key	Random number generation	256 bits - 256 bits	DRBG State - CSP	HMAC DRBG		HMAC DRBG
HMAC DRBG Seed	Random number generation	256 bits - 256 bits	DRBG State - CSP	HMAC DRBG		HMAC DRBG
HMAC DRBG V	Random number generation	256 bits - 256 bits	DRBG State - CSP	HMAC DRBG		HMAC DRBG
HMAC Keys	MACs	160, 256, 384, 512 - 80, 128, 192, 256	HMAC keys - CSP			MAC Generation MAC Verification
PBKDF derived key	Key derivation	112- 256 bits - 112-256 bits	Symmetric Key - CSP			Key Derivation
PBKDF Password	Password for the PBKDF	14 characters (minimum) - 112 bits	Authentication data - CSP			Key Derivation
RSA Private Key	Signing	2048- 4096 bits - 112 to 150 bits	Private Key - CSP	Key Generation		RSA SigGen
RSA Public Key	Verification	2048- 4096 bits - 112 to 150 bits	Public Key - PSP	Key Generation		RSA SigVer

Table 22: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES Key	From App To App	VM:Plaintext	Duration of the service	Explicit	

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES Key Wrap Key	From App	VM:Plaintext	Duration of the service	Explicit	
AES-GCM IV		VM:Plaintext	Duration of the service	Explicit Implicit	
AES-XTS	From App To App	VM:Plaintext	Duration of the service	Explicit	
Entropy input		VM:Plaintext	Duration of the service	Implicit	
HMAC DRBG Key		VM:Plaintext	Duration of the service	Implicit	HMAC DRBG Seed:Derived From HMAC DRBG V:Used With
HMAC DRBG Seed		VM:Plaintext	Duration of the service	Implicit	HMAC DRBG V:Derives HMAC DRBG Key:Derives Entropy input:Incorporates
HMAC DRBG V		VM:Plaintext	Duration of the service	Implicit	HMAC DRBG Seed:Derived From HMAC DRBG Key:Used With
HMAC Keys	From App To App	VM:Plaintext	Duration of the service	Explicit	
PBKDF derived key	To App	VM:Plaintext	Duration of the service	Immediate	PBKDF Password :Derived From
PBKDF Password	From App	VM:Plaintext	Duration of the service	Explicit	PBKDF derived key:Used to establish
RSA Private Key	From App To App	VM:Plaintext	Duration of the service	Explicit	RSA Public Key:Paired With
RSA Public Key	From App To App	VM:Plaintext	Duration of the service	Explicit	RSA Private Key:Paired With

Table 23: SSP Table 2

9.5 Transitions

The module addresses the requirements of FIPS 140-3. *Transitioning the use of cryptographic algorithms and key lengths* (NIST SP 800-131A Rev. 2) provides more specific guidance in regard to transition periods or time frames where an algorithm or key length transitions from Approved to Non-Approved.

None of the Approved algorithms provided by the module are affected by transition periods or time frames.

Recommendation for *Key Management Part 1* (NIST SP 800-57 Part 1 Rev. 5) specifies security strengths that are acceptable for protecting data going forward. Application writers should consider these

acceptable use dates with respect to the expected deployment lifetime of the application and the life of the data being protected. This life depends on the type of key and use, but are from 1-3 years. Refer to SP 800-57 Part 1 Rev. 5 for more explanation.

Strength	Last Date Acceptable
<112	Already disallowed
112	31 Dec 2030
>=128	Acceptable to 2031 and beyond

Table 24: Transitions - Date

The correspondence between security strength, algorithms and key size is specified in the following:

- *Recommendation for Pair-wise Key Establishment Using Integer Factorization Cryptography* (NIST SP 800-56B Rev. 2)
- *Recommendation for Key Management Part 1* (NIST SP 800-57 Part 1 Rev. 5)
- *Recommendation for Applications Using Approved Hash Algorithms* (NIST SP 800-107 Rev. 1).

Strength	Symmetric	RSA	Hash	MAC and KDF
<80		1024	SHA-1	
112	Triple-DES	2048	SHA2-224	
128	AES-128	3072	SHA2-256	SHA-1
192	AES-192	7680	SHA2-384	SHA2-224, SHA2-512/224
256	AES-256	15360	SHA2-512	SHA2-256, SHA2-512/256, SHA2-384, SHA2-512

Table 25: Transitions - Details

10 Self-Tests

The module performs several pre-operational and conditional self-tests to ensure proper operation.

The cryptographic services of the module are disabled when the self-tests are running.

- When self-tests are running all cryptographic operations fail and return the BCM_ERROR_FIPS_MODULE_NOT_READY return status code.
- The `BCM_module_selftest()` status interface returns a state of BCM_MODULE_STATE_NOT_READY.

For all self-test failures, the library notifies the user through the return status codes for the API.

10.1 Pre-Operational Self-Tests

The following table lists the pre-operational self-tests:

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A1204)	128-bit key	Integrity Test	SW/FW Integrity	API return code	Pre-operational software integrity test executes automatically when the module is loaded into memory

Table 26: Pre-Operational Self-Tests

If all self-tests pass, the cryptographic services of the module are enabled, and the module can be used. The `BCM_module_state()` status interface returns a state of BCM_MODULE_STATE_READY.

If the pre-operational software integrity test fails, the module enters the self-test error state.

The pre-operational software integrity tests used are described in detail in Approved Integrity Techniques.

The Cryptographic Algorithm Self-Tests (CAST) are run prior to the pre-operational software integrity test to ensure the MAC implementation used in the integrity test has been self-tested before it is used in the pre-operational software integrity test.

10.2 Conditional Self-Tests

The following table lists the conditional self-tests:

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CCM Encrypt (A1204)	128, 192 and 256-bit	KAT	CAST	API return code	Encrypt	Module startup
AES-CBC Encrypt (A1204)	128, 192 and 256-bit	KAT	CAST	API return code	Decrypt	Module startup
AES-CBC Encrypt (A1204)	128, 192 and 256-bit	KAT	CAST	API return code	Encrypt	Module startup
AES-CCM Decrypt (A1204)	128, 192 and 256-bit	KAT	CAST	API return code	Decrypt	Module startup
AES-CTR Decrypt (A1204)	128, 192 and 256-bit	KAT	CAST	API return code	Decrypt	Module startup
AES-CTR Encrypt (A1204)	128, 192 and 256-bit	KAT	CAST	API return code	Encrypt	Module startup

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-ECB Decrypt (A1204)	128, 192 and 256-bit	KAT	CAST	API return code	Decrypt	Module startup
AES-ECB Encrypt (A1204)	128, 192 and 256-bit	KAT	CAST	API return code	Encrypt	Module startup
AES-GCM Decrypt (A1204)	128, 192 and 256-bit	KAT	CAST	API return code	Decrypt	Module startup
AES-GCM Encrypt (A1204)	128, 192 and 256-bit	KAT	CAST	API return code	Encrypt	Module startup
AES-KW Decrypt (A1204)	128, 192 and 256-bit	KAT	CAST	API return code	Unwrap	Module startup
AES-KW Encrypt (A1204)	128, 192 and 256-bit	KAT	CAST	API return code	Wrap	Module startup
AES-KWP Decrypt (A1204)	128, 192 and 256-bit	KAT	CAST	API return code	Unwrap	Module startup
AES-KWP Encrypt (A1204)	128, 192 and 256-bit	KAT	CAST	API return code	Wrap	Module startup
AES-XTS Decrypt (A1204)	128 and 256-bit	KAT	CAST	API return code	Decrypt	Module startup
AES-XTS Encrypt (A1204)	128 and 256-bit	KAT	CAST	API return code	Encrypt	Module startup
ENT (APT)	Adaptive Proportion Test	Fault Detection Test	CAST	API return code	Health Test runs when a BCM_CTX object creates an entropy source	Continuous
ENT (RCT)	Repetition Count Test	Fault Detection Test	CAST	API return code	Health Test runs when a BCM_CTX object creates an entropy source	Continuous
HMAC DRBG (A1204)	HMAC-SHA2-512	KAT	CAST	API return code	Instantiate, Reseed, Generate	Module startup
HMAC-SHA-1 (A1204)	HMAC with SHA1	KAT	CAST	API return code	MAC	Module Startup

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA2-256 (A1204)	HMAC with SHA2-256	KAT	CAST	API return code	MAC	Module Startup
HMAC-SHA2-384 (A1204)	HMAC with SHA2-384	KAT	CAST	API return code	MAC	Module Startup
HMAC-SHA2-512 (A1204)	HMAC with SHA2-512	KAT	CAST	API return code	MAC	Module Startup
KAS-IFC-SSC (A1204)	2048-bit	KAT	CAST	API return code	RSA Primitive Computation	Module Startup
PBKDF (A1204)	HMAC-SHA2-256	KAT	CAST	API return code	Key Derivation	Module startup
RSA KeyGen (FIPS186-4) (A1204)	2048 to 4096 bit	PCT	PCT	API return code	RSA Key Pair Generation	Key Pair Generation
RSA SigGen (FIPS186-4) (A1204)	2048-bit RSA X9.31 padding SHA2-256 hash	KAT	CAST	API return code	Sign	Module startup
RSA SigVer (FIPS186-4) (A1204)	2048-bit RSA X9.31 padding SHA2-256 hash	KAT	CAST	API return code	Verify	Module startup
SHA-1 (A1204)	SHA-1	KAT	CAST	API return code	Hash	Module startup
SHA2-224 (A1204)	SHA2-224	KAT	CAST	API return code	Hash	Module startup
SHA2-256 (A1204)	SHA2-256	KAT	CAST	API return code	Hash	Module startup
SHA2-384 (A1204)	SHA2-384	KAT	CAST	API return code	Hash	Module startup
SHA2-512 (A1204)	SHA2-512	KAT	CAST	API return code	Hash	Module startup
SHA2-512/224 (A1204)	SHA2-512/224	KAT	CAST	API return code	Hash	Module startup
SHA2-512/256 (A1204)	SHA2-512/256	KAT	CAST	API return code	Hash	Module startup

Table 27: Conditional Self-Tests

For the KATs, the module includes a set of fixed inputs for each algorithm along with corresponding pre-calculated expected outputs. The cryptographic algorithm is run with the fixed inputs, and the algorithm outputs are compared with the expected outputs. If there is any difference the algorithm self-test fails, the CAST fail and the module enters the self-test error state.
If a pair-wise consistency test fails, the key-generation operation fails and returns an error indicator through a return status code. The error is cleared by reattempting the key-generation operation.
If the critical functions test fails, then the entropy collection operation returns an error indicator through a return status code. The error cannot be cleared from the entropy source instance. A new entropy source object must be created to collect entropy.
The repetition count test (RCT) and adaptive proportion test (APT) are defined in SP 800-90B.

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A1204)	Integrity Test	SW/FW Integrity	On-Demand	Module load

Table 28: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CCM Encrypt (A1204)	KAT	CAST	On-Demand	On power on or reset
AES-CBC Encrypt (A1204)	KAT	CAST	On-Demand	On power on or reset
AES-CBC Encrypt (A1204)	KAT	CAST	On-Demand	On power on or reset
AES-CCM Decrypt (A1204)	KAT	CAST	On-Demand	On power on or reset
AES-CTR Decrypt (A1204)	KAT	CAST	On-Demand	On power on or reset
AES-CTR Encrypt (A1204)	KAT	CAST	On-Demand	On power on or reset
AES-ECB Decrypt (A1204)	KAT	CAST	On-Demand	On power on or reset
AES-ECB Encrypt (A1204)	KAT	CAST	On-Demand	On power on or reset
AES-GCM Decrypt (A1204)	KAT	CAST	On-Demand	On power on or reset
AES-GCM Encrypt (A1204)	KAT	CAST	On-Demand	On power on or reset
AES-KW Decrypt (A1204)	KAT	CAST	On-Demand	On power on or reset
AES-KW Encrypt (A1204)	KAT	CAST	On-Demand	On power on or reset

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-KWP Decrypt (A1204)	KAT	CAST	On-Demand	On power on or reset
AES-KWP Encrypt (A1204)	KAT	CAST	On-Demand	On power on or reset
AES-XTS Decrypt (A1204)	KAT	CAST	On-Demand	On power on or reset
AES-XTS Encrypt (A1204)	KAT	CAST	On-Demand	On power on or reset
ENT (APT)	Fault Detection Test	CAST	On-Demand	When Entropy is requested
ENT (RCT)	Fault Detection Test	CAST	On-Demand	When Entropy is requested
HMAC DRBG (A1204)	KAT	CAST	On-Demand	On power on or reset
HMAC-SHA-1 (A1204)	KAT	CAST	On-Demand	On power on or reset
HMAC-SHA2-256 (A1204)	KAT	CAST	On-Demand	On power on or reset
HMAC-SHA2-384 (A1204)	KAT	CAST	On-Demand	On power on or reset
HMAC-SHA2-512 (A1204)	KAT	CAST	On-Demand	On power on or reset
KAS-IFC-SSC (A1204)	KAT	CAST	On-Demand	On power on or reset
PBKDF (A1204)	KAT	CAST	On-Demand	On power on or reset
RSA KeyGen (FIPS186-4) (A1204)	PCT	PCT	On-Demand	Key Pair Generation
RSA SigGen (FIPS186-4) (A1204)	KAT	CAST	On-Demand	On power on or reset
RSA SigVer (FIPS186-4) (A1204)	KAT	CAST	On-Demand	On power on or reset
SHA-1 (A1204)	KAT	CAST	On-Demand	On power on or reset
SHA2-224 (A1204)	KAT	CAST	On-Demand	On power on or reset
SHA2-256 (A1204)	KAT	CAST	On-Demand	On power on or reset
SHA2-384 (A1204)	KAT	CAST	On-Demand	On power on or reset
SHA2-512 (A1204)	KAT	CAST	On-Demand	On power on or reset

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
SHA2-512/224 (A1204)	KAT	CAST	On-Demand	On power on or reset
SHA2-512/256 (A1204)	KAT	CAST	On-Demand	On power on or reset

Table 29: Conditional Periodic Information

10.4 Error States

The following table lists the error states:

Name	Description	Conditions	Recovery Method	Indicator
CAST	Failure while doing Cryptographic Algorithm Self-tests	CAST failure	Reload / Reboot Module	Cryptographic services return BCM_ERROR_FIPS_SELFTEST_FAILURE error code. The BCM_module_state() status interface returns a state of BCM_MODULE_STATE_SELFTEST_FAILED.
ODCAST	Failure when doing on demand Cryptographic Algorithm Self-tests	CAST failure	Reload / Reboot Module	Cryptographic services return BCM_ERROR_FIPS_SELFTEST_FAILURE error code. The BCM_module_state() status interface returns a state of BCM_MODULE_STATE_SELFTEST_FAILED.
ODPOST	Failure when doing on demand integrity test	POST Failure	Reload / Reboot Module	Cryptographic services return BCM_ERROR_FIPS_INTEGRITY_FAILURE error code.
POST	Failure while doing pre-operational integrity tests	POST Failure	Reload / Reboot Module	Cryptographic services return BCM_ERROR_FIPS_INTEGRITY_FAILURE error code. The BCM_module_state() status interface returns a state of BCM_MODULE_STATE_INTEGRITY_FAILED.

Table 30: Error States

Once the module enters the self-test error state then cryptographic services for the module can be re-enabled only by reloading the module.

In this state, the module remains loaded, but the cryptographic functionality of the module is inaccessible.

10.5 Operator Initiation of Self-Tests

The `BCM_module_selftest()` API runs self-tests on demand after the module has loaded. The on-demand self-tests are the software integrity test and the cryptographic algorithm self-tests. The module can also be reloaded to execute the self-tests on-demand.

If a self-test that is run by `BCM_module_selftest()` fails, the module enters the self-test error state.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

11.1.1 Installation

The module is linked to the application at compile time and installed as part of the target application. There is no physical installation, operation or maintenance of the module.

11.1.2 Initialization

In order for the module to be placed in the FIPS 140-3 compliant state, the operator must perform the following configuration steps:

1. Locate the user-supplied configuration function implementation used by the BCM module build. The typical shipped sample file is config_fips_usermode.c.
2. Modify the DRBG selection so that the module uses the approved HMAC DRBG. After this change, all DRBG operations performed by the module will use the HMAC DRBG exclusively.

The `BCM_get_config()` function shall be implemented as follows:

```
BCM_EXPORT BCM_STATUS BCM_CDECL BCM_get_config(BCM_CONFIG
*config)
{
    BCMI_USER_CTX *user_ctx = &g_user_ctx;

    if (config == NULL)
        return BCM_ERROR_NULL_ARG;
    if (config->version != BCM_CONFIG_V2)
        return BCM_ERROR_BAD_CONFIG_VERSION;

    /*
     * Put the module in fips mode.
     */
    config->mode = BCM_MODE_FIPS;

    /*
     * This DRBG algorithm will be used to generate random data.
     * Use the approved HMAC-DRBG with SHA-512 as the default
DRBG algorithm
    * is not approved.
     */
    config->drbg_alg = BCM_ALG_DRBG_HMAC_SHA2_256;
}
```

11.1.3 Startup

The module is started by starting the application that includes it. The module uses operating system services to perform the module startup when the application is started. This module startup includes running the pre-operational integrity test.

Before cryptographic services are made available by the module, the pre-operational integrity tests must complete successfully. These ensure that the application has made no modification to the module as part of its development or installation. For more information about the pre-operational integrity test, see Software/Firmware Security.

11.1.4 Maintenance

Maintenance applies only to the application maintainers. If modifications are made to the application, such as a new version or patch to the application, the module's pre-operational integrity test ensures that the module contained within is unaltered. Application writers should not attempt to modify the module

11.2 Administrator Guidance

For details of the administrative functions, security parameters, and logical interfaces available to the Crypto Officer, refer to Crypto Officer Role.
For access to the Dell BSAFE™ Crypto Module for C Developers Guide please reach out to Dell.

11.3 Non-Administrator Guidance

N/A. The module does not support a non-administrator role

12 Mitigation of Other Attacks

RSA key operations implement blinding, a reversible way of modifying the input data, so as to make the operation immune to timing attacks. Blinding has no effect on the algorithm other than to mitigate attacks on the algorithm.

This mitigation is enabled by default. For optimum security, it should not be disabled. If necessary, it can be disabled with BCM_FLAG_KEY_DISABLE_BLINDING. For more information, see Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems.

RSA signing operations implement a verification step after private key operations. This verification step is in place to prevent potential faults in optimized Chinese Remainder Theorem (CRT) implementations. It has no effect on the signature algorithm.
This mitigation is enabled by default. For optimum security, it should not be disabled. If necessary it can be disabled with BCM_FLAG_KEY_DISABLE_SIGNATURE_CHECK.

For more information, see Breaking public key cryptosystems on tamper resistant devices in the presence of transient faults: Bao, Deng, Han, Jeng and On the Importance of Eliminating Errors in Cryptographic Computations.

RSA PKCS #1 v1.5 encryption padding operations are implemented in constant time in order to make the operation immune to timing attacks.

For this mitigation, constant time padding is built-in and cannot be disabled.
For more information, see. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1.