Dell Australia Pty Limited, BSAFE Product Team

Dell BSAFE™ Crypto Module for C

Version 3.0.1

FIPS 140-3 Non-Proprietary Security Policy

Document Version: 1.2

DELL Technologies

1 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 in the following formats:

- Dell PowerMaxOS™ platform: Static library in Executable and Linkable Format (ELF) format, built for the Intel® x86_64 (64-bit architecture).

- Linux® platform: Shared library in ELF format, built for the Intel x86 (32-bit) and x86_64 (64-bit architecture).

- Windows® platform: Dynamic Link Library in Portable Executable (PE) format, built for the Intel x86_64 (64-bit architecture).

For Linux and Windows platforms, the module is dynamically loaded into the address space of a user process. For the PowerMaxOS platform, it is linked directly into the user software system.

The module follows the standard x86 and x86_64 calling conventions and provides a documented set of functions that can be called from user software.

The name and version of the module can be accessed from the APIs *BCM_module_info()* and *BCM_module_version()*.

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 module boundary is the library. This is a shared library for most platforms, dynamically loaded by the application. For some, it is a static library linked directly into the final application

The underlying logical interface to the module is the API, documented in the *Dell BSAFE™ Crypto Module for C Developers Guide*. The module interfaces are provided as follows:

- Control Input: Provided through the API calls.
- Data Input and Output: Provided through the variables passed with the API calls.
- Status Output: Provided through the return status codes documented for each API call.

These interfaces are 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:

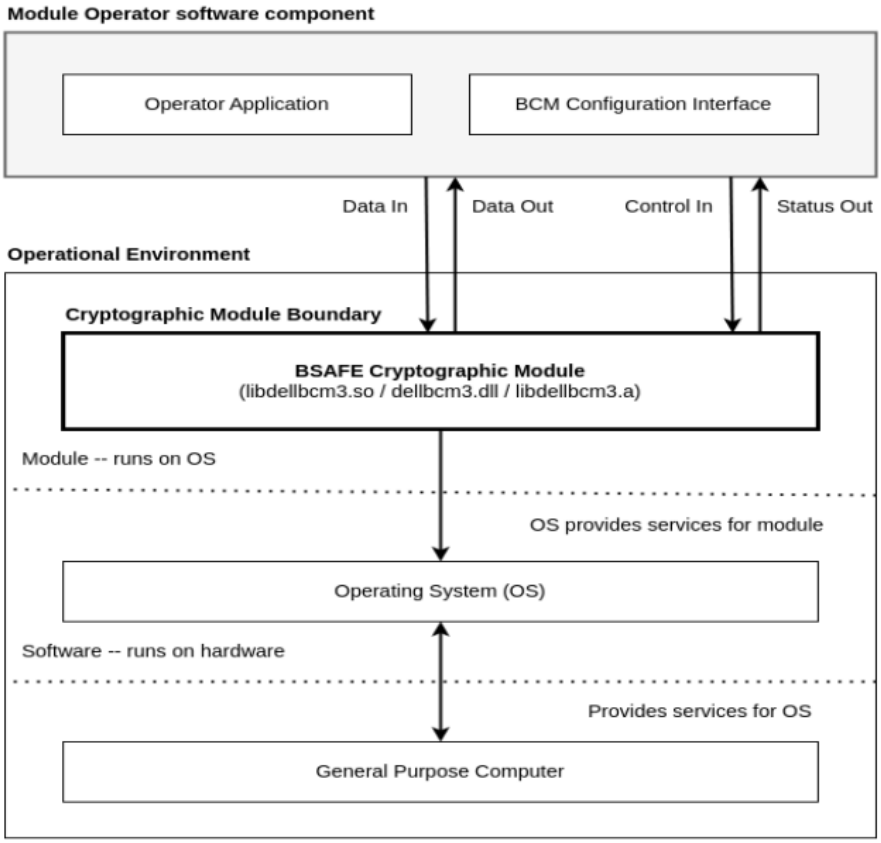

Figure 1: Block Diagram

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
dellbcm3.dll	3.0.1	Windows platform	HMAC-SHA2-256
libdellbcm3.a	3.0.1	PowerMaxOS platform	HMAC-SHA2-256
libdellbcm3.so	3.0.1	Linux platform	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)
Dell PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon Gold 5218	No		3.0.1
Dell PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon Gold 5218	Yes		3.0.1
Dell PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon Gold 6240L	No		3.0.1
Dell PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon Gold 6240L	Yes		3.0.1
Dell PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon Gold 6254	No		3.0.1
Dell PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon Gold 6254	Yes		3.0.1
Dell PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon Platinum 8280L	No		3.0.1
Dell PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon Platinum 8280L	Yes		3.0.1
Microsoft Windows 10 Enterprise x86_64 (64-bit) (Visual Studio 2017)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	No		3.0.1
Microsoft Windows 10 Enterprise x86_64 (64-bit) (Visual Studio 2017)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	Yes		3.0.1
Microsoft Windows 10 Enterprise x86_64 (64-bit) (Visual Studio 2019)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	No		3.0.1
Microsoft Windows 10 Enterprise x86_64 (64-bit) (Visual Studio 2019)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	Yes		3.0.1
Microsoft Windows 11	VMware ESXi 7.0.2 on Dell PowerEdge R630	Intel Xeon E5-2620 v4	No		3.0.1
Microsoft Windows 11	VMware ESXi 7.0.2 on Dell	Intel Xeon E5-2620 v4	Yes		3.0.1

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
	PowerEdge R630				
Microsoft Windows Server 2019 x86_64 (64-bit) (Visual Studio 2017)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6246	No		3.0.1
Microsoft Windows Server 2019 x86_64 (64-bit) (Visual Studio 2017)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6246	Yes		3.0.1
Microsoft Windows Server 2019 x86_64 (64-bit) (Visual Studio 2017)	VMware ESXi 6.7.0 on Dell PowerEdge R7425	AMD EPYC 7451	No		3.0.1
Microsoft Windows Server 2019 x86_64 (64-bit) (Visual Studio 2017)	VMware ESXi 6.7.0 on Dell PowerEdge R7425	AMD EPYC 7451	Yes		3.0.1
Microsoft Windows Server 2019 x86_64 (64-bit) (Visual Studio 2019)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6246	No		3.0.1
Microsoft Windows Server 2019 x86_64 (64-bit) (Visual Studio 2019)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6246	Yes		3.0.1
Microsoft Windows Server 2019 x86_64 (64-bit) (Visual Studio 2019)	VMware ESXi 6.7.0 on Dell PowerEdge R7425	AMD EPYC 7451	No		3.0.1
Microsoft Windows Server 2019 x86_64 (64-bit) (Visual Studio 2019)	VMware ESXi 6.7.0 on Dell PowerEdge R7425	AMD EPYC 7451	Yes		3.0.1
Red Hat Enterprise Linux 7.9 x86 (32-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	No		3.0.1
Red Hat Enterprise Linux 7.9 x86 (32-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	Yes		3.0.1
Red Hat Enterprise Linux 7.9 x86_64 (64-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	No		3.0.1

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Red Hat Enterprise Linux 7.9 x86_64 (64-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	Yes		3.0.1
Red Hat Enterprise Linux 8.5 x86 (32-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	No		3.0.1
Red Hat Enterprise Linux 8.5 x86 (32-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	Yes		3.0.1
Red Hat Enterprise Linux 8.5 x86_64 (64-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	No		3.0.1
Red Hat Enterprise Linux 8.5 x86_64 (64-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	Yes		3.0.1
SUSE Linux Enterprise Server 12 SP5 x86 (32-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	No		3.0.1
SUSE Linux Enterprise Server 12 SP5 x86 (32-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	Yes		3.0.1
SUSE Linux Enterprise Server 12 SP5 x86_64 (64-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	No		3.0.1
SUSE Linux Enterprise Server 12 SP5 x86_64 (64-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	Yes		3.0.1
SUSE Linux Enterprise Server 15 SP3 x86 (32-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6246	No		3.0.1
SUSE Linux Enterprise Server 15 SP3 x86 (32-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6246	Yes		3.0.1
SUSE Linux Enterprise Server 15 SP3 x86_64 (64-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6246	No		3.0.1

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
SUSE Linux Enterprise Server 15 SP3 x86_64 (64-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6246	Yes		3.0.1
SUSE Linux Enterprise Server 15 SP3 x86_64 (64-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R7425	AMD EPYC 7451	No		3.0.1
SUSE Linux Enterprise Server 15 SP3 x86_64 (64-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R7425	AMD EPYC 7451	Yes		3.0.1

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
Dell OneFS 9.7	x86_64 (64-bit)
Dell OneFS 9.8	x86_64 (64-bit)
Dell PowerMaxOS 10	PowerMax storage array compute node (x86_64)
Dell PowerProtect Data Domain OS 7	x86_64 (64-bit)
Dell PowerProtect™ Data Domain™ OS 8	x86_64 (64-bit)
Dell PowerStoreOS 4 (kernel)	x86_64 (64-bit)
Dell PowerStoreOS 4 (user)	x86_64 (64-bit)
Microsoft Windows 10 Enterprise	x86_64 (64-bit)
Microsoft Windows 10 IoT Enterprise LTSC	x86_64 (64-bit)
Microsoft Windows 11	x86_64 (64-bit)
Microsoft Windows Server 2019 x86_64 (64-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640 (Intel Xeon Gold 6246)
Red Hat Enterprise Linux 8.5	x86 (32-bit)
Red Hat Enterprise Linux 8.5	x86_64 (64-bit)
Red Hat Enterprise Linux 9.4	x86_64 (64-bit)
SUSE Linux Enterprise Server 15 SP2	x86 (32-bit)
SUSE Linux Enterprise Server 15 SP2	x86_64 (64-bit)
SUSE Linux Enterprise Server 15 SP3	x86 (32-bit)
SUSE Linux Enterprise Server 15 SP3	x86_64 (64-bit)
SUSE Linux Enterprise Server 15 SP4	x86_64 (64-bit)
SUSE Linux Enterprise Server 15 SP5	x86_64 (64-bit)
SUSE Linux Enterprise Server 15 SP6	x86 (32-bit)
SUSE Linux Enterprise Server 15 SP6	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
Approved	In the Approved mode (BCM_MODE_FIPS), the module only allows for Approved cryptographic algorithms. Non-Approved algorithms not allowed in the approved mode of operation are not available.	Approved	BCM_ctx_is_fips(NULL) returns 1
Non-Approved	In Non-Approved mode (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. The following section details the algorithms available in each mode:

- 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.

- In 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. This is enforced by the module for key objects, but care must be taken when a key is exported from the module. 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_PARAM* use the API *BCM_param_is_fips()* with a return code of 1 as an indicator that the parameters are 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	A2308	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC-CS3	A2308	Direction - decrypt, encrypt Key Length - 128, 192, 256 Payload Length - Payload Length: 128-2048 Increment 8	SP 800-38A
AES-CCM	A2308	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: 0-1024 Increment 8	SP 800-38C
AES-CFB128	A2308	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A2308	Direction - Generation, Verification Key Length - 128, 192, 256 MAC Length - MAC Length: 8-128 Increment 8 Message Length - Message Length: 0-8192 Increment 8	SP 800-38B
AES-CTR	A2308	Direction - Decrypt, 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	A2308	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A2308	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: 0-2048 Increment 8 AAD Length - AAD Length: 0-1024 Increment 8	SP 800-38D

Algorithm	CAVP Cert	Properties	Reference
AES-GMAC	A2308	Direction - Decrypt, Encrypt IV Generation - External Key Length - 128, 192, 256 Tag Length - 128, 32 IV Length - IV Length: 96-120 Increment 8 AAD Length - AAD Length: 0-8192 Increment 8	SP 800-38D
AES-KW	A2308	Direction - Decrypt, Encrypt Cipher - Cipher Key Length - 128, 192, 256 Payload Length - Payload Length: 128-1024 Increment 64	SP 800-38F
AES-KWP	A2308	Direction - Decrypt, Encrypt Cipher - Cipher Key Length - 128, 192, 256 Payload Length - Payload Length: 8-1024 Increment 8	SP 800-38F
AES-OFB	A2308	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-XTS Testing Revision 2.0	A2308	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
DSA KeyGen (FIPS186-4)	A2308	L - 2048, 3072 N - 224, 256	FIPS 186-4
DSA PQGGen (FIPS186-4)	A2308	P/Q Generation Methods - Probable G Generation Methods - Canonical L - 2048, 3072 N - 224, 256 Hash Algorithm - SHA2-224, SHA2-256	FIPS 186-4
DSA PQGVer (FIPS186-4)	A2308	P/Q Generation Methods - Probable G Generation Methods - Canonical L - 1024, 2048, 3072 N - 160, 224, 256 Hash Algorithm - SHA-1, SHA2-224, SHA2-256	FIPS 186-4
DSA SigGen (FIPS186-4)	A2308	L - 2048, 3072 N - 224, 256 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	FIPS 186-4
DSA SigVer (FIPS186-4)	A2308	L - 2048, 3072 N - 224, 256 Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	FIPS 186-4
ECDSA KeyGen (FIPS186-4)	A2308	Curve - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Secret Generation Mode - Testing Candidates	FIPS 186-4

Algorithm	CAVP Cert	Properties	Reference
ECDSA KeyVer (FIPS186-4)	A2308	Curve - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521	FIPS 186-4
ECDSA SigGen (FIPS186-4)	A2308	Component - Yes Curve - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512	FIPS 186-4
ECDSA SigVer (FIPS186-4)	A2308	Component - Yes Curve - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512	FIPS 186-4
HMAC DRBG	A2308	Prediction Resistance - No, Yes Supports Reseed - Yes Mode - SHA2-512 Entropy Input - Entropy Input: 256 Nonce - Nonce: 128 Personalization String Length - Personalization String Length: 0 Additional Input - Additional Input: 0 Returned Bits - 512	SP 800-90A Rev. 1
HMAC-SHA-1	A2308	MAC - MAC: 80-160 Increment 8 Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA2-224	A2308	MAC - MAC: 80-224 Increment 8 Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA2-256	A2308	MAC - MAC: 80-256 Increment 8 Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA2-384	A2308	MAC - MAC: 80-384 Increment 8 Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA2-512	A2308	MAC - MAC: 80-512 Increment 8 Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA2-512/224	A2308	MAC - MAC: 80-224 Increment 8 Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA2-512/256	A2308	MAC - MAC: 80-256 Increment 8 Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA3-224	A2308	MAC - MAC: 80-224 Increment 8 Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA3-256	A2308	MAC - MAC: 80-256 Increment 8 Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA3-384	A2308	MAC - MAC: 80-384 Increment 8 Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1
HMAC-SHA3-512	A2308	MAC - MAC: 80-512 Increment 8 Key Length - Key Length: 8-2048 Increment 8	FIPS 198-1

Algorithm	CAVP Cert	Properties	Reference
KAS-ECC-SSC Sp800-56Ar3	A2308	Domain Parameter Generation Methods - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Scheme - ephemeralUnified - KAS Role - initiator, responder onePassDh - KAS Role - initiator, responder staticUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3
KAS-FFC-SSC Sp800-56Ar3	A2308	Domain Parameter Generation Methods - FB, FC, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 Scheme - dhEphem - KAS Role - initiator, responder dhOneFlow - KAS Role - initiator, responder dhStatic - KAS Role - initiator, responder	SP 800-56A Rev. 3
KAS-IFC-SSC	A2308	Modulo - 2048, 3072, 4096, 6144, 8192 Key Generation Methods - rsakpg2-basic Scheme - KAS1 - KAS Role - initiator, responder	SP 800-56A Rev. 3
KDA HKDF Sp800-56Cr1	A2308	Fixed Info Pattern - context||uPartyInfo||vPartyInfo Fixed Info Encoding - concatenation Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-8192 Increment 8 HMAC Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512	SP 800-56C Rev. 2
KDA OneStep Sp800-56Cr1	A2308	Auxiliary Function Methods - Auxiliary Function Name - SHA-1 MAC Salting Methods - default Fixed Info Pattern - context||uPartyInfo||vPartyInfo Fixed Info Encoding - concatenation Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-8192 Increment 8	SP 800-56C Rev. 2
KDF ANS 9.63 (CVL)	A2308	Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512 Field Size - 224, 571 Shared Info Length - Shared Info Length: 0, 1024 Key Data Length - Key Data Length: 128, 4096	SP 800-135 Rev. 1

Algorithm	CAVP Cert	Properties	Reference
KDF SSH (CVL)	A2308	Cipher - AES-128, AES-192, AES-256 Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1
KDF TLS (CVL)	A2308	TLS Version - v1.2 Hash Algorithm - SHA2-256, SHA2-384	SP 800-135 Rev. 1
PBKDF	A2308	Iteration Count - Iteration Count: 1-10000 Increment 1 HMAC Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512 Password Length - Password Length: 8-128 Increment 1 Salt Length - Salt Length: 128-1024 Increment 8 Key Data Length - Key Data Length: 112-2048 Increment 8	SP 800-132
RSA Decryption Primitive (CVL)	A2308	Modulus Length - 2048	FIPS 186-4
RSA KeyGen (FIPS186-4)	A2308	Key Generation Mode - B.3.6 Modulo - 2048, 3072, 4096 Primality Tests - Table C.2 Info Generated By Server - Yes Public Exponent Mode - Random Private Key Format - Standard	FIPS 186-4
RSA SigGen (FIPS186-4)	A2308	Signature Type - ANSI X9.31, PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096 Hash Pair - Hash Algorithm - SHA2-224	FIPS 186-4
RSA SigVer (FIPS186-2)	A2308	Public Exponent Mode - Random Signature Type - ANSI X9.31, PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096 Hash Pair - Hash Algorithm - SHA-1	FIPS 186-4
RSA SigVer (FIPS186-4)	A2308	Signature Type - ANSI X9.31, PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096 Hash Pair - Hash Algorithm - SHA2-224 Public Exponent Mode - Random	FIPS 186-4
Safe Primes Key Generation	A2308	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	SP 800-56A Rev. 3
Safe Primes Key Verification	A2308	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	SP 800-56A Rev. 3

Algorithm	CAVP Cert	Properties	Reference
SHA-1	A2308	Message Length - Message Length: 0-8192 Increment 8 Function - SHA1	FIPS 180-4
SHA2-224	A2308	Message Length - Message Length: 0-8192 Increment 8 Function - SHA2	FIPS 180-4
SHA2-256	A2308	Message Length - Message Length: 0-8192 Increment 8 Function - SHA2	FIPS 180-4
SHA2-384	A2308	Message Length - Message Length: 0-8192 Increment 8 Function - SHA2	FIPS 180-4
SHA2-512	A2308	Message Length - Message Length: 0-8192 Increment 8 Function - SHA2	FIPS 180-4
SHA2-512/224	A2308	Message Length - Message Length: 0-8192 Increment 8 Function - SHA2	FIPS 180-4
SHA2-512/256	A2308	Message Length - Message Length: 0-8192 Increment 8 Function - SHA2	FIPS 180-4
SHA3-224	A2308	Message Length - Message Length: 0-8192 Increment 8	FIPS 202
SHA3-256	A2308	Message Length - Message Length: 0-8192 Increment 8	FIPS 202
SHA3-384	A2308	Message Length - Message Length: 0-8192 Increment 8	FIPS 202
SHA3-512	A2308	Message Length - Message Length: 0-8192 Increment 8	FIPS 202
SHAKE-128	A2308	Supports Bit-Oriented Messages - No Supports Empty Message - Yes Supports Bit-Oriented Output - No Output Length - Output Length: 16-1024 Increment 8	FIPS 202
SHAKE-256	A2308	Supports Bit-Oriented Messages - No Supports Empty Message - Yes Supports Bit-Oriented Output - No Output Length - Output Length: 16-1024 Increment 8	FIPS 202
TLS v1.2 KDF RFC7627 (CVL)	A2308	Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1

Table 6: Approved Algorithms

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
CKG-XTS	Key Type:Symmetric	N/A	Section 6.3, approved method 1

Name	Properties	Implementation	Reference
CKG-4	Key Type:Symmetric and Asymmetric	N/A	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
AES-CFB (64-bit)	Symmetric encryption
MD5	Message Digesting
RSA-PKCS #1	Key Encapsulation
TDES-CBC	Symmetric encryption
TDES-CFB	Symmetric encryption
TDES-ECB	Symmetric encryption
TDES-OFB	Symmetric encryption
TLS v1.0/1.1 PRF	Key Derivation

Table 8: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
AES-XTS	BC-UnAuth CKG	AES XTS Key generated to comply with the approved key generation guidelines of NIST SP 800-133rev2, Section 6.3		CKG-XTS: () Key Type: Symmetric AES-ECB: (A2308) AES-XTS Testing Revision 2.0: (A2308)
Authenticated Decryption	BC-AuthDecrypt	Decryption using authenticated modes		AES-CCM: (A2308) AES-GCM: (A2308)
Authenticated Encryption	BC-AuthEncrypt	Encryption using authenticated modes		AES-CCM: (A2308) AES-GCM: (A2308)

Name	Type	Description	Properties	Algorithms
Decryption	BC-UnAuthDecrypt	Decryption using unauthenticated modes		AES-CBC: (A2308) AES-CTR: (A2308) AES-ECB: (A2308) AES-CFB128: (A2308) AES-CBC-CS3: (A2308)
DSA Domain Parameter Generation	AsymKeyPair-DomPar	DSA Domain Parameter Generation		DSA PQGGen (FIPS186-4): (A2308) SHA2-224: (A2308) SHA2-256: (A2308)
DSA Domain Parameter Verification	AsymKeyPair-DomPar	DSA Domain Parameter Verification		DSA PQGVer (FIPS186-4): (A2308) SHA-1: (A2308) SHA2-224: (A2308) SHA2-256: (A2308)
FFC Key Generation	AsymKeyPair-KeyGen CKG	FFC Key Generation		CKG-4 : () DSA KeyGen (FIPS186-4): (A2308) Safe Primes Key Generation: (A2308)
DSA Signature Generation	DigSig-SigGen	DSA Signature Generation		DSA SigGen (FIPS186-4): (A2308) SHA2-224: (A2308) SHA2-256: (A2308) SHA2-384: (A2308) SHA2-512: (A2308) SHA2-512/224: (A2308) SHA2-512/256: (A2308) HMAC DRBG: (A2308)

Name	Type	Description	Properties	Algorithms
DSA Signature Verification	DigSig-SigVer	DSA Signature Verification	SHA-1:Per IG C.M, SHA-1 is approved only for legacy use in this signature verification algorithm.	DSA SigVer (FIPS186-4): (A2308) SHA-1: (A2308) SHA2-224: (A2308) SHA2-256: (A2308) SHA2-384: (A2308) SHA2-512: (A2308) SHA2-512/224: (A2308) SHA2-512/256: (A2308)
ECDSA Key Generation	AsymKeyPair-KeyGen CKG	ECDSA Key Generation		CKG-4 : () ECDSA KeyGen (FIPS186-4): (A2308)
ECDSA Key Verification	AsymKeyPair-KeyVer	ECDSA Key Verification		ECDSA KeyVer (FIPS186-4): (A2308)
ECDSA Signature Generation	DigSig-SigGen	ECDSA Signature Generation		ECDSA SigGen (FIPS186-4): (A2308) SHA2-224: (A2308) SHA2-256: (A2308) SHA2-384: (A2308) SHA2-512: (A2308) SHA2-512/224: (A2308) SHA2-512/256: (A2308) SHA3-224: (A2308) SHA3-256: (A2308) SHA3-384: (A2308) SHA3-512: (A2308) HMAC DRBG: (A2308)

Name	Type	Description	Properties	Algorithms
ECDSA Signature Verification	DigSig-SigVer	ECDSA Signature Verification		ECDSA SigVer (FIPS186-4): (A2308) SHA2-224: (A2308) SHA2-256: (A2308) SHA2-384: (A2308) SHA2-512: (A2308) SHA2-512/224: (A2308) SHA2-512/256: (A2308) SHA3-224: (A2308) SHA3-256: (A2308) SHA3-384: (A2308) SHA3-512: (A2308)
Encryption	BC-UnAuth	Encryption using unauthenticated modes		AES-CBC: (A2308) AES-CTR: (A2308) AES-CBC-CS3: (A2308) AES-ECB: (A2308) AES-CFB128: (A2308) AES-OFB: (A2308)
Entropy Conditioning	ENT-Cond	HMAC used for Entropy Conditioning		HMAC-SHA2-256: (A2308) SHA2-256: (A2308)
Entropy Source	ENT-ESV	Non Physical Entropy source		
HMAC DRBG	DRBG	Random Number Generation		CKG-4 : () Key Type: Symmetric and Asymmetric HMAC DRBG: (A2308) HMAC-SHA2-512: (A2308)

Name	Type	Description	Properties	Algorithms
				SHA2-512: (A2308)
KAS-ECC-SSC	KAS-SSC	Computation of shared secrets	IG:IG D.F Scenario 2 path (1)	KAS-ECC-SSC Sp800-56Ar3: (A2308)
KAS-FFC-SSC	KAS-SSC	Computation of shared secrets	IG:IG D.F Scenario 2 path (1)	KAS-FFC-SSC Sp800-56Ar3: (A2308) Safe Primes Key Verification: (A2308)
KAS-IFC-SSC	KAS-SSC	Computation of shared secrets	IG:IG D.F Scenario 1 path (1)	KAS-IFC-SSC: (A2308) RSA Decryption Primitive: (A2308)
Key derivation with ANS 9.63 KDF	KAS-135KDF	Key derivation with ANS 9.63 KDF		KDF ANS 9.63: (A2308) SHA2-224: (A2308) SHA2-256: (A2308) SHA2-384: (A2308) SHA2-512: (A2308) SHA3-224: (A2308) SHA3-256: (A2308) SHA3-384: (A2308) SHA3-512: (A2308)
Key derivation with HKDF	KAS-56CKDF	Key derivation with HKDF		KDA HKDF Sp800-56Cr1: (A2308) SHA-1: (A2308) SHA2-224: (A2308) SHA2-256: (A2308) SHA2-384: (A2308) SHA2-512: (A2308) SHA2-512/224: (A2308) SHA2-512/256:

Name	Type	Description	Properties	Algorithms
				(A2308) SHA3-224: (A2308) SHA3-256: (A2308) SHA3-384: (A2308) SHA3-512: (A2308) HMAC-SHA-1: (A2308) HMAC-SHA2-224: (A2308) HMAC-SHA2-256: (A2308) HMAC-SHA2-384: (A2308) HMAC-SHA2-512: (A2308) HMAC-SHA2-512/224: (A2308) HMAC-SHA2-512/256: (A2308) HMAC-SHA3-224: (A2308) HMAC-SHA3-256: (A2308) HMAC-SHA3-384: (A2308) HMAC-SHA3-512: (A2308)
Key derivation with OneStep KDA	KAS-56CKDF	Key derivation with OneStep KDA		KDA OneStep Sp800-56Cr1: (A2308) SHA-1: (A2308) SHA2-224: (A2308) SHA2-256: (A2308) SHA2-384: (A2308) SHA2-512: (A2308) SHA2-512/224: (A2308) SHA2-512/256: (A2308) SHA3-224:

Name	Type	Description	Properties	Algorithms
				(A2308) SHA3-256: (A2308) SHA3-384: (A2308) SHA3-512: (A2308) HMAC-SHA-1: (A2308) HMAC-SHA2-224: (A2308) HMAC-SHA2-256: (A2308) HMAC-SHA2-384: (A2308) HMAC-SHA2-512: (A2308) HMAC-SHA2-512/224: (A2308) HMAC-SHA2-512/256: (A2308) HMAC-SHA3-224: (A2308) HMAC-SHA3-256: (A2308) HMAC-SHA3-384: (A2308) HMAC-SHA3-512: (A2308)
Key derivation with SSH KDF	KAS-135KDF	Key derivation using SSH KDF		KDF SSH: (A2308) SHA-1: (A2308) SHA2-224: (A2308) SHA2-256: (A2308) SHA2-384: (A2308) SHA2-512: (A2308) SHA3-224: (A2308) SHA3-256: (A2308) SHA3-384: (A2308) SHA3-512: (A2308)

Name	Type	Description	Properties	Algorithms
Key derivation with TLS KDF	KAS-135KDF	Key derivation with TLS KDF	Publication:RFC 5246	KDF TLS: (A2308) SHA2-256: (A2308) SHA2-384: (A2308) HMAC-SHA2-256: (A2308) HMAC-SHA2-384: (A2308)
Key derivation with TLS v1.2 KDF RFC7627	KAS-135KDF	Key derivation with TLS v1.2 KDF RFC7627		TLS v1.2 KDF RFC7627: (A2308) SHA2-256: (A2308) SHA2-384: (A2308) HMAC-SHA2-256: (A2308) HMAC-SHA2-384: (A2308)
Key Unwrap	BC-Auth	Key Unwrapping		AES-ECB: (A2308) AES-KWP: (A2308) AES-KW: (A2308)
Key Wrap	BC-Auth	Key Wrapping		AES-ECB: (A2308) AES-KW: (A2308) AES-KWP: (A2308)
Legacy RSA Signature Verification	DigSig-SigVer	Legacy RSA Signature Verification	Caveat:Legacy Use Only	RSA SigVer (FIPS186-2): (A2308) SHA-1: (A2308) SHA2-224: (A2308) SHA2-256: (A2308) SHA2-384: (A2308) SHA2-512: (A2308)
MAC Generation	MAC	MAC Generation		HMAC-SHA-1: (A2308) HMAC-SHA2-224: (A2308)

Name	Type	Description	Properties	Algorithms
				HMAC-SHA2-256: (A2308) HMAC-SHA2-384: (A2308) HMAC-SHA2-512: (A2308) HMAC-SHA2-512/224: (A2308) HMAC-SHA2-512/256: (A2308) HMAC-SHA3-224: (A2308) HMAC-SHA3-256: (A2308) HMAC-SHA3-384: (A2308) HMAC-SHA3-512: (A2308) AES-CMAC: (A2308) AES-GMAC: (A2308)
MAC Verification	MAC	MAC Verification		HMAC-SHA-1: (A2308) HMAC-SHA2-224: (A2308) HMAC-SHA2-256: (A2308) HMAC-SHA2-384: (A2308) HMAC-SHA2-512: (A2308) HMAC-SHA2-512/224: (A2308) HMAC-SHA2-512/256: (A2308) HMAC-SHA3-224: (A2308) HMAC-SHA3-256: (A2308) HMAC-SHA3-384: (A2308) HMAC-SHA3-512: (A2308) AES-CMAC: (A2308)

Name	Type	Description	Properties	Algorithms
				AES-GMAC: (A2308)
Message Digest	SHA	Message Digest		SHA-1: (A2308) SHA2-224: (A2308) SHA2-256: (A2308) SHA2-384: (A2308) SHA2-512: (A2308) SHA2-512/224: (A2308) SHA2-512/256: (A2308) SHA3-224: (A2308) SHA3-256: (A2308) SHA3-384: (A2308) SHA3-512: (A2308) SHAKE-128: (A2308) SHAKE-256: (A2308)
Password Based Key Derivation	PBKDF	Password Based Key Derivation		PBKDF: (A2308) HMAC-SHA-1: (A2308) HMAC-SHA2-224: (A2308) HMAC-SHA2-256: (A2308) HMAC-SHA2-384: (A2308) HMAC-SHA2-512: (A2308) HMAC-SHA2-512/224: (A2308) HMAC-SHA2-512/256: (A2308) HMAC-SHA3-224: (A2308) HMAC-SHA3-256: (A2308) HMAC-SHA3-384: (A2308)

Name	Type	Description	Properties	Algorithms
				HMAC-SHA3-512: (A2308) SHA-1: (A2308) SHA2-224: (A2308) SHA2-256: (A2308) SHA2-384: (A2308) SHA2-512: (A2308) SHA2-512/224: (A2308) SHA2-512/256: (A2308) SHA3-224: (A2308) SHA3-256: (A2308) SHA3-384: (A2308) SHA3-512: (A2308)
RSA Key Generation	AsymKeyPair-KeyGen CKG	RSA Key Generation		CKG-4 : () RSA KeyGen (FIPS186-4): (A2308)
RSA Signature Generation	DigSig-SigGen	RSA Signature Generation		RSA SigGen (FIPS186-4): (A2308) SHA2-224: (A2308) SHA2-256: (A2308) SHA2-384: (A2308) SHA2-512: (A2308) SHA2-512/224: (A2308) SHA2-512/256: (A2308) HMAC DRBG: (A2308)
RSA Signature Verification	DigSig-SigVer	RSA Signature Verification		RSA SigVer (FIPS186-4): (A2308) SHA2-224: (A2308)

Name	Type	Description	Properties	Algorithms
				SHA2-256: (A2308) SHA2-384: (A2308) SHA2-512: (A2308) SHA2-512/224: (A2308) SHA2-512/256: (A2308)

Table 9: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 Symmetric Key Operations

GCM Mode Ciphers

An AES-GCM IV is constructed in compliance with either IG C.H scenario 1a or IG C.H scenario 2, depending on how the module is used.

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. It must be generated internally. IV generation operates in one of two ways:

IG C.G Scenario 2

- In regular use 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.

IG C.G Scenario 1a

- IG C.H scenario 1a: When used for TLSv1.2 protocol GCM cipher suites, as allowed by SP 800-52 Rev. 2 and defined in RFC 5288, the four-byte salt derived from the TLS handshake process must be input to the module to be used to form part of the IV.

The salt value must be passed as iv in the call to *BCM_cipher_new_AEAD()* and the *BCM_FLAG_CIPHER_PARTIAL_IV* flag must be provided. The salt is used as the first four bytes of the IV.

The remaining eight bytes of the IV, referred to as nonce_explicit in RFC 5288, are generated deterministically by the module using a 64-bit global counter within the module. The module uses the current system time to initialize the counter when it is first used. The system time must be valid to prevent repetition of IVs.

During a TLS connection, if the nonce_explicit part of the IV exhausts the maximum number of possible values for a given session key, a new handshake must be performed to establish a new key.

The TLSv1.2 protocol uses the fragment number as part of the IV. The module derives the remaining IV value using internally managed counter initialized at module load to ensure a probability of 2^{-32} or less, of reusing the same key and IV together. When using a partial IV, the module limits the use of a key and IV pair to 2^{64} - 1 bytes.

XTS Mode Ciphers

AES in XTS mode is approved only for hardware storage applications.

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.

TDES

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, for example, signing, encrypting or wrapping. *Process* refers to processing previously protected data, for example, verifying, decrypting or unwrapping.

Purpose	**Curves**
Protect and Process	B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521
Process only	B-163, B-233, B-283, B-409, B-571, K-163, K-233, K-283, K-409, K-571, P-192, P-224, P-256, P-384, P-521

Table 10: Supported Elliptic Curves

Purpose	**(prime, subprime)**
Protect and Process	(2048, 224), (2048, 256), (3072, 256)
Process only	(1024,160), (2048, 224), (2048, 256), (3072, 256)

Table 11: Supported DSA key pair sizes

Purpose	DH FFC Named domain parameter
Protect	FFDHE2048, FFDHE3072, FFDHE4096, FFDHE6144, FFDHE8192 MODP2048, MODP3072, MODP4096, MODP614

Table 12: Supported Diffie-Hellman named domain parameters

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 13: Approved RSA modulus length for digital signatures

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 one of signing, encryption or key exchange. 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.

Elliptic curve key pairs for digital signature generation must have a strength of 112 bits or stronger. Table 10: Supported Elliptic Curves, lists these in the Protect and Process row. For verification of digital signatures, use curves from the Process only row which includes legacy-use keys of less than 112 bits of strength.

For DSA signatures, only key pairs generated with parameters in the Protect and Process row of Table 11: Supported DSA key pair sizes are approved for signature generation. For verification, the sizes in the Process only row are allowed.

Additionally, parameters from which the keys are derived, or the keys must have been validated before use. An approved DRBG must be used for digital signature generation, and this is provided by the module.

The SHA-1 digest is disallowed for the generation of digital signatures. The digest must be an approved algorithm. 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

The security strength for an HMAC generation or verification must be between 112 and 256 bits.

For HMAC verification, a security strength greater than or equal to 80 and less than 112 is allowed for legacy-use.

2.7.5 Key Derivation Function Operations

The module does not implement the TLS or SSH protocol, and CAVP and CMVP have not tested TLS PRF or SSH KDF when used as part of such protocols.

HMAC-Based Extract-and-Expand Key Derivation Function

An approved HMAC must be used for extract and expand operations.

A particular key-derivation key must only be used for a single key-expansion step. For more information see SP 800-56C Rev. 1.

The derived key must be used only as a secret key.

The derived key shall not be used as a key stream for a stream cipher.

When selecting an HMAC hash, the digest size must be equal to or greater than the desired security strength of the derived key.

The pseudo-random key input to the expansion and the keying material output from the expansion must have lengths that are equal to or greater than the desired security strength of the derived key.

One-Step Key Derivation Function

An approved hash function must be used to derive key materials.

The hash function must meet the security strength required by the cryptographic function for which the keying material is being generated. The security strengths of approved hash functions used in KDFs can be found in SP 800-57 Part 1 Rev. 5.

When selecting a hash algorithm, the digest size must be equal to or greater than the desired security strength of the derived key.

The derived key must be used only as a secret key.

The derived key shall not be used as a key stream for a stream cipher.

The secret data input into this KDF must have a length equal to or greater than the desired security strength of the derived key.

The maximum length of a derived secret key is $(2^{32} - 1) * b$, where b is the digest size of the message digest function in bytes.

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.

Secure Shell Key Derivation

As defined in SP 800-135 Rev. 1, SSH-KDF can be used in the Approved Mode of Operation when it is used with an Approved hash function.

The hash function must meet the security strength required by the cryptographic function for which the keying material is being generated. The security strengths of approved hash functions used in KDFs can be found in SP 800-57 Part 1 Rev. 5.

The operation must be performed in the context of the SSH protocol.

TLS PRF Key Derivation Function

TLS v1.2 PRF KDF is allowed only when the following conditions are satisfied:

- The KDF is performed in the context of the TLS protocol.
- HMAC is as specified in FIPS 198-1.
- P_HASH uses either SHA2-256, SHA2-384, or SHA2-512.

For more information, see SP 800-135 Rev. 1.

X9.63 Key Derivation Function

As defined in SP 800-135 Rev. 1, X9.63 KDF can be used in the Approved Mode of Operation when it is used with an Approved hash function.

The hash function must meet the security strength required by the cryptographic function for which the keying material is being generated. The security strengths of approved hash functions used in KDFs can be found in SP 800-57 Part 1 Rev. 5.

When selecting a hash algorithm, the digest size must be equal to or greater than the desired security strength of the derived key.

The derived key must be used only as a secret key.

The derived key shall not be used as a key stream for a stream cipher.

The length of the derived key shall be less than $(2^{32} - 1) \times b$, where b is the digest size of the message digest function in bytes.

The operation must be performed in the context of ANSI X9.63-2001 key agreement scheme.

2.7.6 Key Agreement Schemes

Key pairs used in key agreement must not also be used to generate a digital signature. The shared secret must be:

- Used only as input to an approved KDF
- Treated as a CSP and destroyed after use. That is, after the secret has been used, the memory holding the secret must be zeroized.

For all schemes:

- Both parties must use validated parameters to generate a key pair. *BCM_param_validate()* provides the validation service.
- Key pairs generated with an approved method such as provided by the module are assumed to be valid.
- A nonce used in a key agreement scheme must be a random value that is generated by the module's approved DRBG. The DRBG that generates a nonce for a scheme must have a security strength greater than or equal to the targeted security strength of the scheme.
- A nonce must have a bit length that is greater than or equal to the targeted security strength of the scheme. Where possible, the bit length of a nonce should be at least twice the targeted security strength of the scheme.

For schemes that use ephemeral keys:

- The key pair is used only for a single transaction.
- The key pair is destroyed after use.

For schemes that use static key pairs:

- A public identifier must be authoritatively associated with the key pair.
- A public identifier must be associated with the public key to allow any peer to recognize the key pair.
- The receiving party must have assurance of the peer's ownership of the private key.
- It is noted that the scheme doesn't provide forward secrecy.

For schemes that use key confirmation:

- Both parties must use a common approved MAC to generate confirmation values.
- The MAC key will be generated as one of the key material elements.
- The input values for MAC tag generation must be formatted as per SP 800-56A Rev. 3.
- The MAC key must be zeroized after use.
- If confirmation fails then destroy all calculated values. All key material is destroyed to prevent it from being used for any other purpose.

ECC based DH key agreement schemes

Curves with at least 112 bits of security strength are allowed. Table 10: Supported Elliptic Curves, lists these in the Protect and Process row.

FFC based DH key agreement schemes

Keys used for DH key agreement should be generated from the Protect and Process row of Table 12: Supported Diffie-Hellman named domain parameters.

Keys from generated parameters should only be used for backwards compatibility with legacy applications and then must have passed parameter validation.

2.7.7 Key Transport Schemes

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.

2.7.8 Key Validation

Asymmetric keys and parameters 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, DSA, DH and ECC keys and parameters.

Named EC and DH parameters are treated as valid. Only named EC parameters are supported.

Key Parameter Generation

The generation of DSA parameters is in accordance with the FIPS 186-4 standard for the generation of probable primes.

2.8 RBG and Entropy

Cert Number	Vendor Name
E277	Dell Australia Pty Limited, BSAFE Product Team

Table 14: Entropy Certificates

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
Dell BSAFE™ Crypto Module Entropy Source	Non-Physical	Dell PowerMaxOS 10 on Intel Xeon Gold 5218, Dell PowerMaxOS 10 on Intel Xeon Gold 6254, Dell PowerMaxOS 10 on Intel Xeon Platinum 8280L, Microsoft Windows Server 2016 (64-bit) on VMware ESXi 6.7 on Intel Xeon Gold 6246, Microsoft Windows Server 2019 on VMware ESXi 6.7 on AMD EPYC 7451 24-Core, Red Hat Enterprise Linux 7 (32-bit) on VMware ESXi 6.7 on Intel Xeon Gold 6136, Red Hat Enterprise Linux 7 (64-bit) on VMware ESXi 6.7 on Intel Xeon Gold 6136	256 bits	255 bits	HMAC-SHA2-256 (A2308)

Table 15: Entropy Sources

The guaranteed amount of entropy for both the SSPs and the random strings generated by the module using the available entropy source(s).

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

Dell BSAFE™ Crypto Module Entropy Source utilizes the HMAC-SHA2-256 vetted conditioner and outputs 255 bits of entropy per each 256-bit output.

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 NDRBG for the number of bits of additional input 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 NDRBG 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.

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 default DRBG provides a security strength as great as that of any supported keys.

2.10 Key Establishment

Key Agreement
The module supports Key Agreement Schemes per SP 800-56A Rev. 3 and IG D.F Scenario 2 (path 2) and SP 800-56B Rev. 2 and IG D.F Scenario 1 (path 1).

Key Transport
The module supports the Key Transport per SP 800-38F and IG D.G (AES-KW/AES-KWP).

2.11 Industry Protocols

The Module conforms to IG D.C References to the Support of Industry Protocols: while it provides SP 800-56A Rev. 3 conformant schemes and APIs oriented to SSH and TLS usage, the Module does not contain the full implementation of SSH or TLS.

The following caveat applies:
No parts of the SSH and TLS protocols, other than the approved cryptographic algorithms and the KDFs, have been tested by the CAVP and CMVP

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 either the BCM_ctx_is_fips(), BCM_param_is_fips(), or BCM_key_is_fips() APIs. The state of the module, from the API BCM_module_state().

Table 16: 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

The module does not implement authentication. The Crypto Officer role is implicitly assumed once the module is loaded and cleared on module unload.

N/A for this module.

4.2 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()*.

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None

Table 17: Roles

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 keys: 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 keys: 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 keys: E - AES-XTS Key: E
Digital Signature Generation	Perform Digital Signature Generation	BCM_key_is_fips() return value	Message Digest	Signature, Status	DSA Signature Generation ECDSA Signature Generation RSA Signature Generation	Crypto Officer - DSA Private Keys: E - ECC Private Keys: E - RSA Private Keys: E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Digital Signature Verification	Perform Digital Signature Verification	BCM_key_is_fips() return value	Message Digest, Signature	Verification Status, Status	DSA Signature Verification ECDSA Signature Verification RSA Signature Verification Legacy RSA Signature Verification	Crypto Officer - DSA Public Keys: E - ECC Public Keys: E - RSA Public Keys: E
ECC Key Agreement	Establish a shared secret	BCM_key_is_fips() return value	Keys	Validation Status, Status	KAS-ECC-SSC	Crypto Officer - ECC Private Keys: R - ECC Public Keys: R
ECDSA Key Verification	Perform Key Verification	BCM_ctx_is_fips() return value	Key Parameters	Status	ECDSA Key Verification	Crypto Officer - ECC Private Keys: W - ECC Public Keys: W
Encryption	Perform unauthenticated encryption	BCM_key_is_fips() return value	Plaintext, key	Ciphertext, Status	AES-XTS Encryption	Crypto Officer - AES keys: E - AES-XTS Key: E
FFC Key Agreement	Establish a shared secret	BCM_key_is_fips() return value	Keys	Validation Status, Status	KAS-FFC-SSC	Crypto Officer - DH Private Keys: R - DH Public Keys: R
Key Derivation	Perform Key Derivation	BCM_key_is_fips() return value	Secret	Key text, Status	Key derivation with ANS 9.63 KDF	Crypto Officer - Derived key:

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					Key derivation with HKDF Key derivation with OneStep KDA Key derivation with SSH KDF Key derivation with TLS KDF Key derivation with TLS v1.2 KDF RFC7627 Password Based Key Derivation	G,R,Z - KDF secret: W,E,Z - Key Derivation Key: G,E - DH Shared Secret: W,E - ECC Shared Secret: W,E
Key Encapsulation	RSA key encapsulation	BCM_key_is_fips() return value	Keys	Validation Status, Status	KAS-IFC-SSC	Crypto Officer - RSA Private Keys: R - RSA Public Keys: R
Key Export	Perform Key Export	BCM_key_is_fips() return value	Key	Key text, Status		Crypto Officer
Key Generation	Generate a symmetric or asymmetric key	BCM_ctx_is_fips() return value	Key Parameters	Key text, Status	HMAC DRBG KAS-ECC-SSC KAS-FFC-SSC FFC Key Generation ECDSA Key Generation RSA Key Generation	Crypto Officer - DH Private Keys: G - DH Public Keys: G - DSA Private Keys: G - DSA Public

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					Entropy Source Entropy Conditioning	Keys: G - ECC Private Keys: G - ECC Public Keys: G - Entropy input: W - HMAC DRBG Key: G,E - HMAC DRBG Seed: G,E - HMAC DRBG V: G,E - RSA Private Keys: G - RSA Public Keys: G
Key Import	Perform Key Import	BCM_key_is_fips() return value	Key text	Status		Crypto Officer - AES keys: W - DH Private Keys: W - DH Public Keys: W - DSA Private Keys: W - DSA Public Keys: W - ECC Private Keys: W - ECC Public Keys: W - RSA Private

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						Keys: W - RSA Public Keys: W
Key Parameter Generation	DSA domain parameter generation	BCM_key_is_fips() return value	Key Parameters	Domain Parameter, Status	DSA Domain Parameter Generation	Crypto Officer - DSA parameters: G
Key Parameter Validation	DSA domain parameter validation	BCM_key_is_fips() return value	Domain Parameter	Domain Parameter	DSA Domain Parameter Verification	Crypto Officer - DSA parameters: W,E
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: R,E
Key Zeroization	Perform Key Zeroization	N/A				Crypto Officer
MAC Generation	Perform MAC Generation	BCM_ctx_is_fips() return value	Secret, Message, key	Verify Status, 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	HMAC DRBG Entropy Source Entropy	Crypto Officer - Entropy input: G,E - HMAC

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					Conditioning	DRBG Key: G,E - HMAC DRBG Seed: G,E - HMAC DRBG V: G,E
Self-test	Perform Self-test	BCM_ctx_is_fips() return value	Command	Command	None	Crypto Officer
Show Status	Show module status		API Call	Module Status	None	Crypto Officer
Show Version	Show module name and version		API Call	Module version, Status	None	Crypto Officer

Table 18: Approved Services

For each service, the Approved Mode indicator is obtained by checking the service status and the Approved mode of the Context, Key, or Key Parameters. A return status code indicates the service status. For information about individual functions that implement each service, see the Dell BSAFE™ Crypto Module for C Developers Guide.

The indicator for the service is one of the following:

- **C**: The Approved Mode indicator is obtained by checking the service return status code and the mode of the operation's context by calling *BCM_ctx_is_fips()*. For approved services, the FIPS indicator function will return 1.
- **K**: The Approved Mode indicator is obtained by checking the service return status code and the mode of the operation's key by calling *BCM_key_is_fips()*. For approved services, the FIPS indicator function will return 1.
- **P**: The Approved Mode indicator is obtained by checking the service return status code and the mode of the operation's key parameters by calling *BCM_param_is_fips()*. For approved services, the FIPS indicator function will return 1.

4.4 Non-Approved Services

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

Name	Description	Algorithms	Role
Key Derivation	Derive key text given input secret	TLS v1.0/1.1 PRF	CO
Key Encapsulation	Encrypt or decrypt a key with an RSA key encryption key	RSA-PKCS #1	CO

Name	Description	Algorithms	Role
Message Digest	Digest a message	MD5	CO
Symmetric Decryption	Decrypt with symmetric cipher	AES-CFB (64-bit) TDES-CBC TDES-CFB TDES-ECB TDES-OFB	CO
Symmetric Encryption	Encrypt with symmetric cipher	AES-CFB (64-bit) TDES-CBC TDES-CFB TDES-ECB TDES-OFB	CO

Table 19: Non-Approved Services

The indicator for the service is one of the following:

- **C**: The Approved Mode indicator is obtained by checking the service return status code and the mode of the operation's context by calling *BCM_ctx_is_fips()*. For these Non-Approved services, the FIPS indicator function will return 0.
- **K**: The Approved Mode indicator is obtained by checking the service return status code and the mode of the operation's key by calling *BCM_key_is_fips()*. For these Non-Approved services, the FIPS indicator function will return 0.

While in the Non-Approved mode the approved services are still available as the module changes modes depending on the services used.

4.5 External Software/Firmware Loaded

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

5 Software/Firmware Security

5.1 Integrity Techniques

Depending on the platform, the module is either a shared library, dynamic link library or an object file. When the module 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. The built-in Integrity Test Key is used as the MAC secret.

The constituents of the module are platform-specific:

- For a Linux platform, the module consists of a shared library, *libdellbcm3.so*
- For a PowerMaxOS platform, the module consists of a static library, *libdellbcm3.a*
- For a Windows platform, the module consists of a dynamic library, *dellbcm3.dll*

During the pre-operational software 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 pre-operational software 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 operation attempted will return the *BCM_ERROR_FIPS_INTEGRITY_FAILURE* error status code.

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 (Cert. # A2308). The Cryptographic Algorithm Self-Test (CAST) for HMAC is run prior to the pre-operational software integrity test. This is done to ensure that the MAC implementation used in the integrity test has been self-tested before it is used in the pre-operational software integrity test.

5.2 Initiate on Demand

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

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

How Requirements are Satisfied:

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

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 be accessed or modified only 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 operational environments listed in Tested Operational Environments table and Vendor Affirmed Operational Environments table.

Each supported operational environment manages its own processes and memory in a logically separated manner. The process management setting is not configurable on the supported operational 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 20: 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
App Read	VM	Operator Application in TOEPP	Plaintext	Manual	Electronic	
App Write	Operator Application in TOEPP	VM	Plaintext	Manual	Electronic	

Table 21: 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
Explict	SSPs are zeroized when the cryptographic object is deleted by the module. Zeroization is performed by overwriting the data with zero valued bytes SSPs are zeroized when the associated key or cryptographic object is deleted by the module. 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 power has been removed 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 for the DRBG's internal state are immediately zeroized.	N/A

Table 22: SSP Zeroization Methods

BSAFE Crypto 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:
   - *BCM_CIPHER* cryptographic objects with a call to *BCM_cipher_delete()*.
   - *BCM_MAC* cryptographic objects with a call to *BCM_mac_delete()*.
   - *BCM_DIGEST* cryptographic objects with a call to *BCM_digest_delete()*.
   - *BCM_KEY* objects with a call to *BCM_key_delete()*.
   - *BCM_PARAM* objects with a call to *BCM_param_delete()*.
   - *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 Wrap Key	AES key wrapping and unwrapping	128, 192, 256 bits - 128, 192, 256 bits	Symmetric Key - CSP	CKG-4		Key Wrap Key Unwrap
AES keys	Encryption and Decryption	128, 192, 256 bits - 128, 192, 256 bits	Symmetric Key - CSP	CKG-4		Authenticated Decryption Authenticated Encryption Decryption Encryption
AES-GCM IV	AES-GCM Initialization Vector	96 bits - 96 bits	Initialization Vector - PSP	HMAC DRBG		Authenticated Decryption Authenticated Encryption

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
						Decryption Encryption
AES-XTS Key	Encryption and Decryption	128, 256 bits - 128, 256 bits	Symmetric Key - CSP	CKG-XTS		AES-XTS
Derived key	The keying material produced by the KDF	128 - 256 bits - 112 - 256 bits	Derived Key - CSP	Key derivation with ANS 9.63 KDF Key derivation with HKDF Key derivation with OneStep KDA Key derivation with SSH KDF Key derivation with TLS KDF Key derivation with TLS v1.2 KDF RFC7627 Password Based Key Derivation		Key derivation with ANS 9.63 KDF Key derivation with HKDF Key derivation with OneStep KDA Key derivation with SSH KDF Key derivation with TLS KDF Key derivation with TLS v1.2 KDF RFC7627 Password Based Key Derivation
DH Private Keys	Shared secret computation	2048 - 8192 bits - 112 - 200 bits	Private Key - CSP	FFC Key Generation		KAS-FFC-SSC
DH Public Keys	Shared secret computation	2048 - 8192 bits - 112 - 200 bits	Public Key - PSP	FFC Key Generation		KAS-FFC-SSC
DH Shared Secret	Computed shared secret	2048-8192 bits -	Shared Secret - CSP	KAS-FFC-SSC		Key derivation with TLS

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	(Includes the Pre-Master Secret)	112 - 200 bits				KDF Key derivation with TLS v1.2 KDF RFC7627
DSA parameters	Domain parameter validation	2048 / 3072 bits - 112 / 128 bits	Domain Parameters - CSP	DSA Domain Parameter Generation		FFC Key Generation DSA Domain Parameter Verification DSA Signature Generation DSA Signature Verification
DSA Private Keys	Signing	2048 / 3072 bits - 112 / 128 bits	Private Key - PSP	FFC Key Generation		DSA Signature Generation
DSA Public Keys	Verification	2048 / 3072 bits - 112 / 128 bits	Public Key - CSP	FFC Key Generation		DSA Signature Verification
ECC Private Keys	Shared secret computation and signing	P-224, P-256, P-384, P-521 - 112 - 256 bits	Private Key - CSP	ECDSA Key Generation		KAS-ECC-SSC
ECC Public Keys	Shared secret computation and signature verification	P-224, P-256, P-384, P-521 - 112 - 256 bits	Public Key - PSP	ECDSA Key Generation		KAS-ECC-SSC
ECC Shared Secret	Computed shared secret (Includes the Pre-Master Secret)	P-224, P-256, P-384, P-521 - 112 - 256 bits	Shared Secret - CSP	KAS-ECC-SSC		Key derivation with TLS KDF Key derivation with TLS

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
						v1.2 KDF RFC7627
Entropy input	Entropy input for DRBG	256 bits - 256 bits	Entropy - CSP	Entropy Source Entropy Conditioning		HMAC DRBG
HMAC DRBG Key	Random number generation	256 bits - 256 bits	DRBG State - CSP	Entropy Source		HMAC DRBG
HMAC DRBG Seed	Random number generation	256 bits - 256 bits	DRBG State - CSP	Entropy Source		HMAC DRBG
HMAC DRBG V	Random number generation	256 bits - 256 bits	DRBG State - CSP	Entropy Source		HMAC DRBG
HMAC Keys	MAC generation and verification	160, 256, 384, 512 bits - 160, 256, 384, 512 bits	Symmetric Key - CSP	CKG-4		MAC Generation MAC Verification
KDF secret	Key derivation	128 - 256 bits - 112 - 256 bits	Symmetric Key - CSP			Key derivation with ANS 9.63 KDF Key derivation with HKDF Key derivation with OneStep KDA Key derivation with SSH KDF Key derivation with TLS KDF Key derivation with TLS

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
						v1.2 KDF RFC7627 Password Based Key Derivation
Key Derivation Key	Key Derivation key (Includes the Master Secret)	160 - 512 bits - 112 - 256 bits	Symmetric Key - CSP			Key derivation with HKDF Key derivation with TLS KDF Key derivation with TLS v1.2 KDF RFC7627
RSA Private Keys	Signing; key transport (KTS-IFC-SSC) private key operations	2048 - 4096- bits - 112 - 150 bits	Private Key - CSP	RSA Key Generation		RSA Signature Generation
RSA Public Keys	Signature verification; key transport (KTS-IFC-SSC) public key operations	2048 - 4096 bits - 112 - 150 bits	Public Key - PSP	RSA Key Generation		RSA Signature Verification

Table 23: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES Key Wrap Key	App Write App Read	VM:Plaintext	Duration of the service	Implicit	
AES keys	App Write App Read	VM:Plaintext	Duration of the service	Explict	
AES-GCM IV		VM:Plaintext	Duration of the service	Implicit Explict	

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES-XTS Key	App Write App Read	VM:Plaintext	Duration of the service	Explict	
Derived key	App Read	VM:Plaintext	Duration of the service	Immediate	Key Derivation Key:Derived From
DH Private Keys	App Write App Read	VM:Plaintext	Duration of the service	Explict	DH Public Keys:Paired With
DH Public Keys	App Write App Read	VM:Plaintext	Duration of the service	Explict	DH Private Keys:Paired With
DH Shared Secret	App Read	VM:Plaintext	Duration of the service	Explict Immediate	DH Private Keys:Derived From DH Public Keys:Derived From
DSA parameters	App Write App Read	VM:Plaintext	Duration of the service	Explict	
DSA Private Keys	App Write App Read	VM:Plaintext	Duration of the service	Explict	DSA Public Keys:Paired With
DSA Public Keys	App Write App Read	VM:Plaintext	Duration of the service	Explict	DSA Private Keys:Paired With
ECC Private Keys	App Write App Read	VM:Plaintext	Duration of the service	Explict	ECC Public Keys:Paired With
ECC Public Keys	App Write App Read	VM:Plaintext	Duration of the service	Explict	ECC Private Keys:Paired With
ECC Shared Secret	App Read	VM:Plaintext	Duration of the service	Explict Immediate	ECC Private Keys:Derived From ECC Public Keys:Derived From
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

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
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	App Write App Read	VM:Plaintext	Duration of the service	Explict	
KDF secret	App Write	VM:Plaintext	Duration of the service	Immediate	
Key Derivation Key	App Write	VM:Plaintext	Duration of the service	Immediate	
RSA Private Keys	App Write App Read	VM:Plaintext	Duration of the service	Explict	RSA Public Keys:Paired With
RSA Public Keys	App Write App Read	VM:Plaintext	Duration of the service	Explict	RSA Private Keys:Paired With

Table 24: SSP Table 2

9.5 Transitions

The module addresses the requirements of FIPS 140-3. Transitioning the use of cryptographic algorithms and key lengths (SP 800-131A Rev. 2) provides more specific guidance concerning transition periods and 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 (SP 800-57 Part 1 Rev. 5) specifies security strengths that are acceptable for protecting data going forward. Application writers should consider the specified acceptable use dates with respect to the expected deployment lifetime of the application and the lifespan of the data being protected.

The lifespan depends on the type of key and use, but is from 1-3 years. For more information, refer to SP 800-57, Part 1.

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

Table 25: Security Strength Time Frames

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

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

Refer to the latest NIST specifications for up-to-date recommendations on Security Strength.

Strength	Symmetric	RSA	EC	Hash	MAC and KDF
< 80		1024	160	SHA-1	
112	3DES	2048	224	SHA2-224; SHA2-512/224; SHA3-224	
128	AES-128	3072	256	SHA2-256; SHA2-512/224; SHA3-256; SHAKE-128	CMAC-AES; GMAC-AES; SHA-1
192	AES-192	7680	384	SHA2-382; SHA3-384	CMAC-AES; GMAC-AES; SHA2-224; SHA2-512/224; SHA3-224
256	AES-256	15360	521	SHA2-512; SHA3-512; SHAKE-128	CMAC-AES; GMAC-AES; SHA2-256; SHA2-512/256; SHA2-384; SHA2-512; SHA3-256; SHA3-384; SHA3-512

Table 26: Correspondence between Security Strength, Algorithms, and Key Size

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 (A2308)	HMAC-SHA2-256 signature:32 bytes HMAC secret:18 bytes	Integrity Test	SW/FW Integrity	API return code	Pre-operational software integrity test executes automatically when the module is loaded into memory.

Table 27: 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 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 KW Unwrap	128, 192 and 256-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Unwrap	Module startup

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES KW Wrap	128, 192 and 256-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Wrap	Module startup
AES KWP Unwrap	128, 192 and 256-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Unwrap	Module startup
AES KWP Wrap	128, 192 and 256-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Wrap	Module startup
AES-CBC Decrypt	128, 192 and 256-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Decrypt	Module startup
AES-CBC Encrypt	128, 192 and 256-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Encrypt	Module startup
AES-CCM Decrypt	128, 192 and 256-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Decrypt	Module startup
AES-CCM Encrypt	128, 192 and 256-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Encrypt	Module startup
AES-CMAC Generation (A2308)	128, 192 and 256-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Generation	Module startup
AES-CMAC Verification (A2308)	128, 192 and 256-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Verification	Module startup
AES-CTR Decrypt	128, 192 and 256-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Decrypt	Module startup
AES-CTR Encrypt	128, 192 and 256-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Encrypt	Module startup
AES-ECB Decrypt	128, 192 and 256-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Decrypt	Module startup
AES-ECB Encrypt	128, 192 and 256-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Encrypt	Module startup
AES-GCM Decrypt	128, 192 and 256-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Decrypt	Module startup

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-GCM Encrypt	128, 192 and 256-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Encrypt	Module startup
AES-GMAC Verify (A2308)	128, 192 and 256-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Verify	Module startup
AES-GMAC Generate (A2308)	128, 192 and 256-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Generate	Module startup
AES-XTS Decrypt	128 and 256-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Decrypt	Module startup
AES-XTS Encrypt	128 and 256-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Encrypt	Module startup
AES-XTS Key Test	K1 != K2 Test	Critical Function	Critical Function	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Compare	On cipher initialization
DSA KeyGen (FIPS186-4) (A2308)	2048-bit and 3072-bit keys	PCT	PCT	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Key Generation	Key Pair Generation
DSA SigGen (FIPS186-4) (A2308)	2048-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Generation	Module startup
DSA SigVer (FIPS186-4) (A2308)	2048-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Verification	Module startup
ECDSA KeyGen (FIPS186-4) (A2308)	Curves B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224,	PCT	PCT	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Key Pair Generation	Key Pair Generation

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
	P-256, P-384, P-521					
ECDSA SigGen (FIPS186-4) (A2308)	P-224	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Generation	Module startup
ECDSA SigVer (FIPS186-4) (A2308)	P-224	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Verification	Module startup
ENT (APT)	Adaptive Proportion Test	Fault Detection Test	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Health Test runs when a BCM_CTX object creates an entropy source	Continuous
ENT (RCT)	Repetition Count Test	Fault Detection Test	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Health Test runs when a BCM_CTX object creates an entropy source	Continuous
HMAC DRBG (A2308)	HMAC-SHA2-512	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Instantiate, Reseed, Generate	Module startup
HMAC-SHA-1 (A2308)	HMAC with SHA1	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	MAC	Module startup
HMAC-SHA2-224 (A2308)	HMAC with SHA2-224	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	MAC	Module startup

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA2-256 (A2308)	HMAC with SHA2-256	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	MAC	Module startup
HMAC-SHA2-384 (A2308)	HMAC with SHA2-384	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	MAC	Module startup
HMAC-SHA2-512 (A2308)	HMAC with SHA2-512	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	MAC	Module startup
HMAC-SHA2-512/224 (A2308)	HMAC with SHA2-512/224	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	MAC	Module startup
HMAC-SHA2-512/256 (A2308)	HMAC with SHA2-512/224	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	MAC	Module startup
HMAC-SHA3-224 (A2308)	HMAC with SHA3-224	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	MAC	Module startup
HMAC-SHA3-256 (A2308)	HMAC with SHA3-256	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	MAC	Module startup
HMAC-SHA3-384 (A2308)	HMAC with SHA3-384	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	MAC	Module startup
HMAC-SHA3-512 (A2308)	HMAC with SHA3-512	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	MAC	Module startup
KAS-ECC-SSC Sp800-56Ar3 (A2308)	P-224 and K-233	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Shared secret computation	Module startup
KAS-FFC-SSC Sp800-	ffdhe2048	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Shared secret computation	Module startup

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
56Ar3 (A2308)						
KAS-IFC-SSC (A2308)	2048-bit	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	RSA Primitive Computation	Module startup
KDA HKDF Sp800-56Cr1 (A2308)	HMAC-SHA2-256	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Key Derivation	Module startup
KDA OneStep Sp800-56Cr1 (A2308)	SHA2-256	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Key Derivation	Module startup
KDF ANS 9.63 (A2308)	SHA2-256	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Key Derivation	Module startup
KDF SSH (A2308)	SHA-1	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Key Derivation	Module startup
PBKDF (A2308)	HMAC-SHA2-256	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Key Derivation	Module startup
RSA KeyGen (FIPS186-4) (A2308)	2048 to 4096-bit	PCT	PCT	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Key Pair Generation	Key Pair Generation
RSA SigGen (FIPS186-4) (A2308)	2048-bit RSA X9.31 padding SHA2-256 hash	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Sign	Module startup
RSA SigVer (FIPS186-2) (A2308)	Legacy; 2048-bit RSA X9.31 padding SHA2-256 hash	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Verify	Module startup

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
RSA SigVer (FIPS186-4) (A2308)	2048-bit RSA X9.31 padding SHA2-256 hash	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Verify	Module startup
SHA-1 (A2308)	SHA-1	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Hash	Module startup
SHA2-224 (A2308)	SHA2-224	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Hash	Module startup
SHA2-256 (A2308)	SHA2-256	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Hash	Module startup
SHA2-384 (A2308)	SHA2-384	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Hash	Module startup
SHA2-512 (A2308)	SHA2-512	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Hash	Module startup
SHA2-512/224 (A2308)	SHA2-512/224	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Hash	Module startup
SHA2-512/256 (A2308)	SHA2-512/256	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Hash	Module startup
SHA3-224 (A2308)	SHA3-224	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Hash	Module startup
SHA3-256 (A2308)	SHA3-256	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Hash	Module startup
SHA3-384 (A2308)	SHA3-384	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Hash	Module startup
SHA3-512 (A2308)	SHA3-512	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Hash	Module startup
SHAKE-128 (A2308)	SHAKE-128	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Hash	Module startup
SHAKE-256 (A2308)	SHAKE-256	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Hash	Module startup

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
SP 800-56A Rev. 3 (Safe primes key generation)	2048-bit to 8192-bit keys	PCT	PCT	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Safe Prime Key Generation	Key Pair Generation
TLS v1.2 KDF RFC7627 (A2308)	SHA2-256	KAT	CAST	Pass: API return code Fail: BCM_ERROR_FIPS_SELFTEST_FAILURE	Key Derivation	Module startup

Table 28: 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 Entropy Source self-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.

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A2308)	Integrity Test	SW/FW Integrity	On Demand	Module load

Table 29: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES KW Unwrap	KAT	CAST	On-Demand	On power on or reset
AES KW Wrap	KAT	CAST	On-Demand	On power on or reset

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES KWP Unwrap	KAT	CAST	On-Demand	On power on or reset
AES KWP Wrap	KAT	CAST	On-Demand	On power on or reset
AES-CBC Decrypt	KAT	CAST	On-Demand	On power on or reset
AES-CBC Encrypt	KAT	CAST	On-Demand	On power on or reset
AES-CCM Decrypt	KAT	CAST	On-Demand	On power on or reset
AES-CCM Encrypt	KAT	CAST	On-Demand	On power on or reset
AES-CMAC Generation (A2308)	KAT	CAST	On-Demand	On power on or reset
AES-CMAC Verification (A2308)	KAT	CAST	On-Demand	On power on or reset
AES-CTR Decrypt	KAT	CAST	On-Demand	On power on or reset
AES-CTR Encrypt	KAT	CAST	On-Demand	On power on or reset
AES-ECB Decrypt	KAT	CAST	On-Demand	On power on or reset
AES-ECB Encrypt	KAT	CAST	On-Demand	On power on or reset
AES-GCM Decrypt	KAT	CAST	On-Demand	On power on or reset
AES-GCM Encrypt	KAT	CAST	On-Demand	On power on or reset
AES-GMAC Verify (A2308)	KAT	CAST	On-Demand	On power on or reset
AES-GMAC Generate (A2308)	KAT	CAST	On-Demand	On power on or reset
AES-XTS Decrypt	KAT	CAST	On-Demand	On power on or reset
AES-XTS Encrypt	KAT	CAST	On-Demand	On power on or reset
AES-XTS Key Test	Critical Function	Critical Function	On-Demand	When K1 and K2 are imported into the module
DSA KeyGen (FIPS186-4) (A2308)	PCT	PCT	On-Demand	When a key pair is generated
DSA SigGen (FIPS186-4) (A2308)	KAT	CAST	On-Demand	On power on or reset

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
DSA SigVer (FIPS186-4) (A2308)	KAT	CAST	On-Demand	On power on or reset
ECDSA KeyGen (FIPS186-4) (A2308)	PCT	PCT	On-Demand	When a key pair is generated
ECDSA SigGen (FIPS186-4) (A2308)	KAT	CAST	On-Demand	On power on or reset
ECDSA SigVer (FIPS186-4) (A2308)	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 (A2308)	KAT	CAST	On-Demand	On power on or reset
HMAC-SHA-1 (A2308)	KAT	CAST	On-Demand	On power on or reset
HMAC-SHA2-224 (A2308)	KAT	CAST	On-Demand	On power on or reset
HMAC-SHA2-256 (A2308)	KAT	CAST	On-Demand	On power on or reset
HMAC-SHA2-384 (A2308)	KAT	CAST	On-Demand	On power on or reset
HMAC-SHA2-512 (A2308)	KAT	CAST	On-Demand	On power on or reset
HMAC-SHA2-512/224 (A2308)	KAT	CAST	On-Demand	On power on or reset
HMAC-SHA2-512/256 (A2308)	KAT	CAST	On-Demand	On power on or reset
HMAC-SHA3-224 (A2308)	KAT	CAST	On-Demand	On power on or reset
HMAC-SHA3-256 (A2308)	KAT	CAST	On-Demand	On power on or reset
HMAC-SHA3-384 (A2308)	KAT	CAST	On-Demand	On power on or reset
HMAC-SHA3-512 (A2308)	KAT	CAST	On-Demand	On power on or reset
KAS-ECC-SSC Sp800-56Ar3 (A2308)	KAT	CAST	On-Demand	On power on or reset
KAS-FFC-SSC Sp800-56Ar3 (A2308)	KAT	CAST	On-Demand	On power on or reset
KAS-IFC-SSC (A2308)	KAT	CAST	On-Demand	On power on or reset

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
KDA HKDF Sp800-56Cr1 (A2308)	KAT	CAST	On-Demand	On power on or reset
KDA OneStep Sp800-56Cr1 (A2308)	KAT	CAST	On-Demand	On power on or reset
KDF ANS 9.63 (A2308)	KAT	CAST	On-Demand	On power on or reset
KDF SSH (A2308)	KAT	CAST	On-Demand	On power on or reset
PBKDF (A2308)	KAT	CAST	On-Demand	On power on or reset
RSA KeyGen (FIPS186-4) (A2308)	PCT	PCT	On-Demand	When a key pair is generated
RSA SigGen (FIPS186-4) (A2308)	KAT	CAST	On-Demand	On power on or reset
RSA SigVer (FIPS186-2) (A2308)	KAT	CAST	On-Demand	On power on or reset
RSA SigVer (FIPS186-4) (A2308)	KAT	CAST	On-Demand	On power on or reset
SHA-1 (A2308)	KAT	CAST	On-Demand	On power on or reset
SHA2-224 (A2308)	KAT	CAST	On-Demand	On power on or reset
SHA2-256 (A2308)	KAT	CAST	On-Demand	On power on or reset
SHA2-384 (A2308)	KAT	CAST	On-Demand	On power on or reset
SHA2-512 (A2308)	KAT	CAST	On-Demand	On power on or reset
SHA2-512/224 (A2308)	KAT	CAST	On-Demand	On power on or reset
SHA2-512/256 (A2308)	KAT	CAST	On-Demand	On power on or reset
SHA3-224 (A2308)	KAT	CAST	On-Demand	On power on or reset
SHA3-256 (A2308)	KAT	CAST	On-Demand	On power on or reset
SHA3-384 (A2308)	KAT	CAST	On-Demand	On power on or reset
SHA3-512 (A2308)	KAT	CAST	On-Demand	On power on or reset
SHAKE-128 (A2308)	KAT	CAST	On-Demand	On power on or reset

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
SHAKE-256 (A2308)	KAT	CAST	On-Demand	On power on or reset
SP 800-56A Rev. 3 (Safe primes key generation)	PCT	PCT	On-Demand	When a key pair is generated
TLS v1.2 KDF RFC7627 (A2308)	KAT	CAST	On-Demand	On power on or reset

Table 30: 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	Reloading the module	Cryptographic services return BCM_ERROR_FIPS_SELFTEST_FAILURE error code. The BCM_module_state() control interface returns a state of BCM_MODULE_STATE_SELFTEST_FAILED.
ODCAST	Failure when doing on demand Cryptographic Algorithm Self-tests	CAST failure	Reloading the module	Cryptographic services return BCM_ERROR_FIPS_SELFTEST_FAILURE error code. The BCM_module_state() control interface returns a state of BCM_MODULE_STATE_SELFTEST_FAILED.
ODPOST	On-demand integrity test failure	On-demand integrity test failure	Reloading the module	Cryptographic services return BCM_ERROR_FIPS_INTEGRITY_FAILURE error code. The BCM_module_state() control interface returns a state of BCM_MODULE_STATE_INTEGRITY_FAILED.
POST	Module pre-operational integrity test failure	Module pre-operational integrity test failure	Reloading the module	Cryptographic services return BCM_ERROR_FIPS_INTEGRITY_FAILURE error code. The BCM_module_state() control interface returns a state of BCM_MODULE_STATE_INTEGRITY_FAILED.

Table 31: Error States

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

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

For the PowerMaxOS platform, the module is linked into the application at compile time, and installed as part of the target application. For Linux and Windows platforms, the module must be installed with the target application.

The Crypto Officer should follow a secure installation procedure to install the module. For all target platforms the installation process should check the integrity of the installed files by checking the hash value of the original files and installed files.

A minimum privileged user on the operating system should be created solely to execute the application. This user should not share any other roles in the operating system. The installation process should set the minimum execution permission to the installed files for the user.

If the module is dynamically loaded into the application, the installation process should ensure that the path to the module cannot be modified by other OS users.

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_V3)
        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_512;
}
```

11.1.3 Startup

The module is started by initiating the application that statically or dynamically linked 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 software integrity test. These ensure that the application has made no modification to the module as part of its development or installation.

Before cryptographic services are made available by the module, the pre-operational software integrity test must complete successfully. For more information about the pre-operational software integrity test, see Software/Firmware Security section.

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

11.4 Maintenance Requirements

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 software integrity test ensures that the module contained within, or dynamically loaded, is

unaltered. Application writers should not attempt to modify the module library or object file as the module will refuse to load or perform cryptographic operations.

12 Mitigation of Other Attacks

RSA, EC and DSA key operations implement blinding, a reversible way of modifying the input data, 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.