

Dell Australia Pty Limited, BSAFE Product Team

Dell BSAFE™ Crypto Module for C

Module Version 3.0.1

FIPS 140-3 Security Policy

Document Version 1.13

BSAFE
FIPS 140 Validated

This document is a non-proprietary security policy for the Dell BSAFE™ Crypto Module for C, version 3.0.1 (BSAFE Crypto Module) from Dell Australia Pty Limited, BSAFE Product Team.

This document may be freely reproduced and distributed whole and intact including the copyright notice.

Contents:

Preface	3
1 General	4
2 Cryptographic Module Specification	5
3 Cryptographic Module Interfaces	18
4 Roles, Services and Authentication	19
5 Software/Firmware Security	25
6 Operational Environment	26
7 Physical Security	27
8 Non-invasive Security	27
9 Sensitive Security Parameters Management	28
10 Self-tests	35
11 Life-cycle Assurance	41
12 Mitigation of Other Attacks	48
13 Acronyms and terms	50

Preface

With the exception of the non-proprietary *Dell BSAFE™ Crypto Module for C Security Policy* document, the overall FIPS 140-3 Security Level 1 validation submission documentation is proprietary to Dell Australia Pty Limited and is releasable only under appropriate non-disclosure agreements. For access to the documentation, contact [Dell Support](#).

This security policy describes how the BSAFE Crypto Module meets the Security Level 1 requirements for all aspects of FIPS 140-3, and how to securely operate it.

Federal Information Processing Standards Publication 140-3 - Security Requirements for Cryptographic Modules (FIPS 140-3) details the U.S. Government requirements for cryptographic modules. More information about the FIPS 140-3 standard and validation program is available on the [NIST website](#).

This document deals only with operations and capabilities of the BSAFE Crypto Module in the technical terms of a FIPS 140-3 cryptographic module security policy. More information about BSAFE Crypto Module and the entire BSAFE product line is available from Dell Support.

Terminology

In this document, the term BSAFE Crypto Module denotes the BSAFE Crypto Module for C, version 3.0.1, FIPS 140-3 validated Cryptographic Module for Overall Security Level 1 for the C language.

The BSAFE Crypto Module for C is also referred to as:

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

1 General

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:

ISO/IEC 24759 Section 6	FIPS 140-3 Section Title	Security Level
1	General	1
2	Cryptographic Module Specification	1
3	Cryptographic Module Interfaces	1
4	Roles, Services, and Authentication	1
5	Software/Firmware Security	1
6	Operational Environment	1
7	Physical Security ¹	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

Table 1 Security Levels

¹The module relies on the physical security provided by the host on which it runs.

2 Cryptographic Module Specification

BSAFE Crypto Module 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 following diagram illustrates the cryptographic module boundary and logical interfaces:

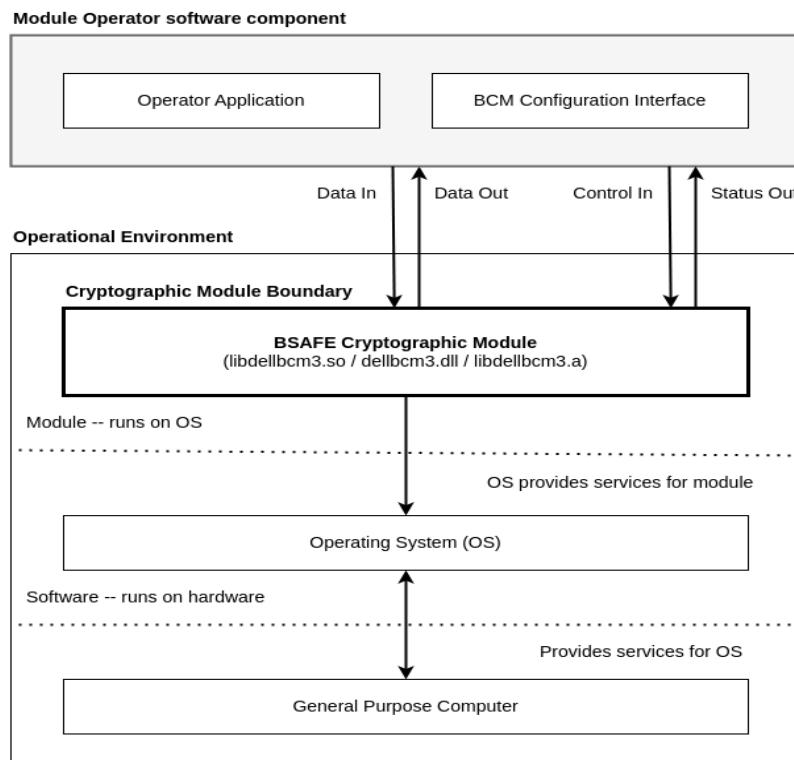


Figure 1 Cryptographic boundary

2.1 Module Description

The module is identified as *Dell BSAFE™ Crypto Module for C*, version 3.0.1. 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`

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

The FIPS 140-3 validation certificate can be located on the NIST Cryptographic Module Validation Program ([CMVP](#)) page using the module name and version reported.

2.2 Software Module Cryptographic Boundaries

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 containing the module's entire executable code. The module relies on the physical security provided by the host computer in which it runs.

The tested operational environment physical perimeter of the module is the case of 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 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 provides Control Input through the API calls. Data Input and Output are provided in the variables passed with the API calls. Status Output is provided through the return status codes documented for each call. This is illustrated in [Figure 1 Cryptographic boundary](#).

2.3 Operational Environments

For FIPS 140-3 validation, the module is tested by an accredited FIPS 140-3 testing laboratory on the following operational environments:

#	Operating System	Hardware Platform	Processor	PAA/ Acceleration
1	Dell PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon Gold 5218	Yes
2	Dell PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon Gold 5218	No
3	Dell PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon Gold 6240L	Yes
4	Dell PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon Gold 6240L	No
5	Dell PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon Gold 6254	Yes
6	Dell PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon Gold 6254	No
7	Dell PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon Platinum 8280L	Yes
8	Dell PowerMaxOS 10	PowerMax storage array compute node	Intel Xeon Platinum 8280L	No
9	Microsoft Windows 11	VMware ESXi 7.0.2 on Dell PowerEdge R630	Intel Xeon E5-2620 v4	Yes
10	Microsoft Windows 11	VMware ESXi 7.0.2 on Dell PowerEdge R630	Intel Xeon E5-2620 v4	No
11	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
12	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
13	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
14	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
15	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
16	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

Table 2 Tested Operational Environments

#	Operating System	Hardware Platform	Processor	PAA/ Acceleration
17	Microsoft Windows Server 2019 x86_64 (64-bit) (Visual Studio 2019)	VMware ESXi 6.7.0 on Dell PowerEdge R7425	AMD EPYC 7451	Yes
18	Microsoft Windows Server 2019 x86_64 (64-bit) (Visual Studio 2019)	VMware ESXi 6.7.0 on Dell PowerEdge R7425	AMD EPYC 7451	No
19	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
20	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
21	Microsoft Windows Server 2019 x86_64 (64-bit) (Visual Studio 2017)	VMware ESXi 6.7.0 on Dell PowerEdge R7425	AMD EPYC 7451	Yes
22	Microsoft Windows Server 2019 x86_64 (64-bit) (Visual Studio 2017)	VMware ESXi 6.7.0 on Dell PowerEdge R7425	AMD EPYC 7451	No
23	Red Hat Enterprise Linux 8.5 x86_64 (64-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	Yes
24	Red Hat Enterprise Linux 8.5 x86_64 (64-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	No
25	Red Hat Enterprise Linux 8.5 x86 (32-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	Yes
26	Red Hat Enterprise Linux 8.5 x86 (32-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	No
27	Red Hat Enterprise Linux 7.9 x86_64 (64-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	Yes
28	Red Hat Enterprise Linux 7.9 x86_64 (64-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	No
29	Red Hat Enterprise Linux 7.9 x86 (32-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	Yes
30	Red Hat Enterprise Linux 7.9 x86 (32-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	No
31	SUSE Linux Enterprise Server 15 SP3 x86_64 (64-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6246	Yes
32	SUSE Linux Enterprise Server 15 SP3 x86_64 (64-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6246	No
33	SUSE Linux Enterprise Server 15 SP3 x86_64 (64-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R7425	AMD EPYC 7451	Yes
34	SUSE Linux Enterprise Server 15 SP3 x86_64 (64-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R7425	AMD EPYC 7451	No

Table 2 Tested Operational Environments (continued)

#	Operating System	Hardware Platform	Processor	PAA/ Acceleration
35	SUSE Linux Enterprise Server 15 SP3 x86 (32-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6246	Yes
36	SUSE Linux Enterprise Server 15 SP3 x86 (32-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6246	No
37	SUSE Linux Enterprise Server 12 SP5 x86_64 (64-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	Yes
38	SUSE Linux Enterprise Server 12 SP5 x86_64 (64-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	No
39	SUSE Linux Enterprise Server 12 SP5 x86 (32-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	Yes
40	SUSE Linux Enterprise Server 12 SP5 x86 (32-bit)	VMware ESXi 6.7.0 on Dell PowerEdge R640	Intel Xeon Gold 6136	No

Table 2 Tested Operational Environments (continued)

Dell BSAFE affirms compliance for the following operational environments:

#	Operating System	Hardware Platform
1	Dell PowerProtect™ Data Domain™ OS 8	x86_64 (64-bit)
2	Dell PowerProtect Data Domain OS 7	x86_64 (64-bit)
3	Dell OneFS 9.8	x86_64 (64-bit)
4	Dell OneFS 9.7	x86_64 (64-bit)
5	Dell PowerStoreOS 4 (user)	x86_64 (64-bit)
6	Dell PowerStoreOS 4 (kernel)	x86_64 (64-bit)
7	Microsoft Windows 10 IoT Enterprise LTSC	x86_64 (64-bit)
8	Red Hat Enterprise Linux 9.4	x86_64 (64-bit)
9	Red Hat Enterprise Linux 8.5	x86_64 (64-bit)
10	SUSE Linux Enterprise Server 15 SP6	x86_64 (64-bit)
11	SUSE Linux Enterprise Server 15 SP6	x86 (32-bit)
12	SUSE Linux Enterprise Server 15 SP5	x86_64 (64-bit)
13	SUSE Linux Enterprise Server 15 SP4	x86_64 (64-bit)
14	SUSE Linux Enterprise Server 15 SP3	x86_64 (64-bit)

Table 3 Vendor Affirmed Operational Environments

#	Operating System	Hardware Platform
15	SUSE Linux Enterprise Server 15 SP3	x86 (32-bit)
16	SUSE Linux Enterprise Server 15 SP2	x86_64 (64-bit)
17	SUSE Linux Enterprise Server 15 SP2	x86 (32-bit)

Table 3 Vendor Affirmed Operational Environments (continued)

Note: When running the module on an affirmed platform, no assurances are made about the minimum strength of generated SSPs, such as keys.

2.4 Cryptographic Algorithms

The following table lists the BSAFE Crypto Module Approved algorithms, with the appropriate standards and CAVP validation certificate numbers:

CAVP Cert	Algorithm and Standard	Mode/Method	Description / Key Size(s) / Key Strength(s)	Use / Function
A2308	AES SP 800-38A	CBC, CBC-CS3, CFB 128-bit, CTR, ECB, and OFB	128, 192, 256-bit key sizes	Symmetric encryption
A2308	AES SP 800-38C	CCM	128, 192, 256-bit key sizes	Symmetric encryption
A2308	AES SP 800-38D	GCM with automatic IV ¹ generation and TLS partial IV ² generation.	128, 192, 256-bit key sizes	Symmetric encryption
A2308	AES SP 800-38E	XTS ³	128, 256-bit key sizes	Symmetric encryption
A2308	KTS SP 800-38F	AES Key Wrap, and Key Wrap with Padding.	128, 192, and 256-bit key sizes	Key wrapping
A2308	DSA FIPS 186-4	Domain parameter generation/validation	2048 and 3072-bit key sizes	Domain parameter generation/validation
A2308	DSA FIPS 186-4	Key generation. Signature generation and signature verification with SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256. Signature verification with SHA-1.	2048 and 3072-bit key sizes	Key generation, signature generation, and signature verification
A2308	ECDSA / FIPS 186-4	Key generation. Key validation. Signature generation and signature verification with SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512.	112 to 256 bits strength	Key generation, key validation, signature generation, and signature verification
A2308	RSA FIPS 186-2 (for legacy use) ⁴	Signature verification with SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512.	2048 to 4096-bit key size	Signature verification

Table 4 Approved Algorithms

CAVP Cert	Algorithm and Standard	Mode/Method	Description / Key Size(s) / Key Strength(s)	Use / Function
A2308	RSA FIPS 186-4	Key generation. Signature generation and signature verification with SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256.	2048 to 4096-bit key size	Key generation, signature generation, and signature verification
A2308	CVL FIPS 186-4	RSASP ⁵ component	2048 to 4096-bit key size	Signature generation
A2308	CVL SP 800-56B Rev. 2	RSADP ⁶ component	2048 to 4096-bit key size	Asymmetric encryption
A2308	SP 800-56A Rev. 3	Safe Primes Key generation	ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	Key generation
A2308	SP 800-56A Rev. 3	Safe Primes Key validation	ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192	Key validation
A2308	KDF SP 800-132	PBKDF2 ⁷	112 to 256 bits strength	Key derivation
A2308	KDF SP 800-108	KBKDF ⁸ with SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512	128 to 256 bits strength	Key derivation
A2308	CVL SP 800-135 Rev. 1	SSH-KDF ⁹ with SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224 ¹⁰ , SHA3-256 ¹⁰ , SHA3-384 ¹⁰ , SHA3-512 ¹⁰	128 to 256 bits strength	Key derivation

Table 4 Approved Algorithms (continued)

CAVP Cert	Algorithm and Standard	Mode/Method	Description / Key Size(s) / Key Strength(s)	Use / Function
A2308	CVL SP 800-135 Rev. 1 RFC 7627	TLS v1.2 PRF with SHA2-256, SHA2-384, SHA2-512	128 to 256 bits strength	Key derivation
A2308	CVL RFC 5246	TLS KDF with SHA2-256, SHA2-384, SHA2-512	128 to 256 bits strength	Key derivation
A2308	CVL RFC 8446	TLS v1.3 PRF with HMAC-SHA2-256, HMAC-SHA2-384	128 to 256 bits strength	Key derivation
A2308	CVL SP 800-135 Rev. 1	X9.63 KDF ¹¹ with SHA1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224 ¹⁰ , SHA3-256 ¹⁰ , SHA3-384 ¹⁰ , SHA3-512 ¹⁰	128 to 256 bits strength	Key derivation
A2308	DRBG ¹² SP 800-90A	AES-CTR	128, 192, 256 bits strength	Random bit generation
A2308	DRBG SP 800-90A	HMAC SHA2-512	256 bits strength	Random bit generation
A2308	SHS FIPS 180-4	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	112 to 256 bits strength	Message digesting
A2308	SHA3 FIPS 202	SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE-128, SHAKE-256	112 to 256 bits strength	Message digesting
A2308	HMAC ¹³ FIPS 198-1	HMAC with SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512	128 to 256 bits strength	MAC generation
A2308	AES SP 800-38B	CMAC with AES (128, 192, 256)	128, 192, 256 bits strength	MAC generation
A2308	AES SP 800-38D	GMAC (128, 192, 256)	128, 192, 256 bits strength	MAC generation

Table 4 Approved Algorithms (continued)

CAVP Cert	Algorithm and Standard	Mode/Method	Description / Key Size(s) / Key Strength(s)	Use / Function
A2308	KAS-ECC-SSC SP 800-56A Rev. 3	Schemes: Ephemeral Unified Model, One-Pass Diffie-Hellman Model Static Unified Model Curves: P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571	112 to 256 bits strength	Shared Secret Generation
A2308	KAS-FFC-SSC SP 800-56A Rev. 3	Schemes: Diffie-Hellman Ephemeral Only, Diffie-Hellman One Flow Diffie-Hellman Static Domain Parameter Generation methods: MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192 ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192	2048 to 8192-bit key size	Shared Secret Generation
A2308	KAS-IFC-SSC SP 800-56B Rev. 2	Scheme: KAS1 Key Generation methods: An RSA key pair with a private key in the basic format, with a random public exponent.	2048 to 8192-bit key size	Shared Secret Generation
A2308	KDA SP 800-56C Rev. 1	HKDF ¹⁴ with SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512	128 to 256 bits key strength	Key Derivation
A2308	KDA SP 800-56C Rev. 1	One-Step KDF with SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512	128 to 256 bits strength	Key Derivation
VA ¹⁵	CKG of symmetric keys/ SP 800-133 Rev. 2	Direct output of approved DRBG used to generate 128, 192, or 256 bit AES keys. FIPS 140-3 Implementation Guidance, IG D.H.	128-256 bits strength	Key Generation

Table 4 Approved Algorithms (continued)

CAVP Cert	Algorithm and Standard	Mode/Method	Description / Key Size(s) / Key Strength(s)	Use / Function
VA	CKG of asymmetric keys/ SP 800-133 Rev. 2	Direct output of approved DRBG used to generate prime number seeds and private key values. FIPS 140-3 Implementation Guidance, IG D.H.	128-256 bits strength	Key Generation

Table 4 Approved Algorithms (continued)

¹Initialization Vector (IV).

²The TLSv1.2 protocol uses the fragment number as part of the IV. The module generates 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.

³AES in XTS mode is approved only for hardware storage applications. The two keys concatenated to create the single double-length key must be checked to ensure they are different.

⁴Algorithms designated as “Legacy” can only be used on data that was generated prior to the Legacy Date specified in FIPS 140-3 IG C.M.

⁵RSA signature primitive 1 (RSASP1).

⁶RSA decryption primitive (RSADP). RSADP shall only be used within the context of an SP800-56B rev 2 Key Transport Scheme (KTS).

⁷Password-based key derivation function 2 (PBKDF2). As defined in SP 800-132, PBKDF2 can be used in the Approved Mode of Operation when used with Approved symmetric key and message digest algorithms. For more information, see [Crypto Officer Guidance](#).

⁸Key-based KDF (KBKDF), using pseudo-random functions HMAC-based Feedback Mode. As defined by the HKDF expand step.

⁹Secure Shell key derivation function (SSH-KDF).

¹⁰SHA3 options for SSH-KDF and X9.63 KDF are vendor affirmed since CAVP testing was not available at the time of module submission.

¹¹ANSI X9.63 KDF.

¹²Deterministic Random Bit Generator (DRBG).

¹³Hash-based Message Authentication Code (HMAC).

¹⁴HMAC-based Extract-and-Expand KDF (HKDF).

¹⁵Vendor-affirmed algorithms.

The following table lists the BSAFE Crypto Module Non-Approved algorithms, not allowed in the Approved Mode of Operation:

Algorithm / Function	Use / Function
AES in CFB in 64-bit mode	Symmetric encryption
DES3 ¹ (three key) in CBC, CFB, ECB and OFB modes	Symmetric encryption
MD5	Message Digesting
RSA-PKCS #1	Key Encapsulation
TLS v1.0/1.1 PRF	Key Derivation

Table 5 Non-Approved Algorithms Not Allowed in the Approved Mode of Operation

¹Triple Data Encryption Standard.

Note: BSAFE Crypto Module does not support any non-approved but allowed algorithms nor non-approved but allowed algorithms with no security claimed.

2.5 Certification Levels

BSAFE Crypto Module is validated with an overall Security Level 1 for FIPS 140-3, with Security Level 1 for each individual area. See [Table 1, Security Levels](#) for additional details.

2.6 Modes of Operation

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

- In the Approved mode (`BCM_MODE_FIPS`), the module allows only those cryptographic algorithms listed in [Table 4, Approved Algorithms](#). Non-Approved algorithms not allowed in the approved mode of operation are not available.
- 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 is available to the Crypto Officer.

Warning: 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 operating in Non-Approved mode.

2.6.1 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.

2.6.2 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.

2.7 Operating the Cryptographic Module

The module initializes itself automatically when it is loaded dynamically, or as part of the user software system, and runs the POST automatically, regardless of the mode of operation at startup.

See [Cryptographic Module Specification](#) for the module file format for the supported platforms.

The cryptographic algorithm self-tests are executed when the specific algorithm is accessed for the first time by the user software system. All the cryptographic algorithm self-tests can be run manually by calling `BCM_module_selftest()`.

The module does not support degraded operation. If any self-test fails, the cryptographic services of the module are disabled for both modes of operation.

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

The following table lists the ports and interfaces, and the data that passes over each:

Physical Port	Logical Interface	Data that passes over port/interface
N/A	Data input	Service inputs
N/A	Data output	Service outputs
N/A	Control input	Configuration parameters for the API <code>BCM_module_configure()</code> which sets the mode of operation.
N/A	Status output	Mode of operation indicator, from either the <code>BCM_ctx_is_fips()</code> , <code>BCM_param_is_fips()</code> , or <code>BCM_key_is_fips()</code> APIs. The state of the module, from the API <code>BCM_module_state()</code> . For other API status, refer to the Outputs column of Table 4, Approved Algorithms .

Table 6 Ports and Interfaces

Note: The module does not support a Control Output interface.

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 Crypto Officer Role

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

4.2 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*.

4.2.1 Roles, Services, Inputs and Outputs

The following is a list of services available to the single Crypto Officer (CO) role.

Role	Service	Input	Output
CO	AES Encryption	Plaintext	Ciphertext, Status
CO	AES Decryption	Ciphertext	Plaintext, Status
CO	Message Digest	Message	Digest, Status
CO	MAC Generation	Secret, Message	MAC, Status
CO	MAC Verification	Secret, Message, MAC	Verify Status, Status
CO	DRBG Initialization	-	-
CO	Random Number Generation	-	Random Bytes, Status
CO	Key Generation	-	Status
CO	Key Import	Key text	Status
CO	Key Export	-	Key text, Status
CO	Key Deletion	-	-

Table 7 Roles, Service Commands, Input and Output

Role	Service	Input	Output
CO	Key Derivation	Secret	Key text, Status
CO	Key Wrap	-	Wrapped key text, Status
CO	Key Unwrap	Wrapped key text	Status
CO	Key Encapsulation (decrypt)	Encapsulated key text	Status
CO	Digital Signature Generation	Message Digest	Signature, Status
CO	Digital Signature Verification	Message Digest, Signature	Verify Status, Status
CO	Key Agreement	Peer public key text	Shared Secret, Status
CO	Key Parameter Generation	-	Status
CO	Key Parameter Validation	-	Status
CO	Show Module Version Information	-	Module Version, Status
CO	Show Status	-	Module Status
CO	Self-test	-	Status

Table 7 Roles, Service Commands, Input and Output

4.2.2 Approved Services

The following is a list of approved services provided by the module:

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSPs	Indicator¹
AES Encryption	Encrypt with symmetric cipher	AES	AES keys	CO	E	K
AES Decryption	Decrypt with symmetric cipher	AES	AES keys	CO	E	K
Message Digest	Digest a message	SHS, SHA3	-	CO	-	C

Table 8 Approved Services

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSPs	Indicator¹
MAC Generation	Generate a Message Authentication Code	HMAC, AES (CMAC, GMAC)	MAC secret	CO	W, E, Z	C
MAC Verification	Verify a Message Authentication Code	HMAC, AES (CMAC, GMAC)	MAC secret	CO	W, E, Z	C
DRBG Initialization	Prepare for random number or key generation	DRBG	Entropy State, DRBG entropy input, DRBG seed, CTR DRBG key value, CTR DRBG V value, HMAC DRBG key value, HMAC DRBG V value	CO	G	C
Random Number Generation	Generate a random number	DRBG	DRBG entropy input, DRBG seed, CTR DRBG key value, CTR DRBG V value, HMAC DRBG key value, HMAC DRBG V value	CO	E	C
Key Generation	Generate a symmetric or asymmetric key	CKG RSA, ECDSA, DSA and Safe Primes Key generation	AES keys, RSA keys, DH keys, DSA keys, ECC keys	CO	G	P
Key Import	Import a key into the module	Safe Primes Key validation, DSA parameter validation	AES keys, RSA keys, DH keys, DSA keys, ECC keys	CO	W	C
Key Export	Export a key from the module	-	AES keys, RSA keys, DH keys, DSA keys, ECC keys	CO	R	K
Key Deletion	Delete a key from the module	-	AES keys, RSA keys, DH keys, DSA keys, ECC keys	CO	Z	K

Table 8 Approved Services (continued)

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSPs	Indicator¹
Key Derivation	Derive key text given input secret	KDF (PBKDF2, KBKDF) CVL (SSH-KDF, TLS v1.2 PRF, TLS v1.3 PRF, X9.63 KDF) KDA (HKDF, One-Step)	KDF secret Derived key text	CO	W, E, Z R, Z	C
Key Wrap	Encrypt an AES key with an AES key encryption key	KTS	AES key (wrapping key) AES key (wrapped key)	CO	E R	K
Key Unwrap	Decrypt an AES key with an AES key encryption key	KTS	AES key (wrapping key) AES key (unwrapped key)	CO	E W	K
Key Encapsulation (decrypt)	Decrypt an AES or RSA key with an RSA key encryption key	CVL (RSADP)	RSA key (key encryption key) AES key or RSA key (decrypted key)	CO	E W	K
Digital Signature Generation	Sign a message	RSA, DSA, ECDSA (signature generation) CVL (RSASP1)	RSA keys, DSA keys, ECC keys (private keys)	CO	E	K
Digital Signature Verification	Verify the signature for a message	RSA, DSA, ECDSA. RSA FIPS 186-2 (for legacy use) ² (signature verification)	RSA keys, DSA keys, ECC keys (public key)	CO	E	K

Table 8 Approved Services (continued)

Service	Description	Approved Security Functions	Keys and/or SSPs	Roles	Access rights to Keys and/or SSPs	Indicator¹
Key Agreement	Establish a shared secret	KAS-ECC-SSC KAS-FFC-SSC KAS-IFC-SSC	DH keys, ECC keys (peer public key)	CO	W, Z	K
			DH keys, ECC keys (peer public key, private key)		E	
Key Parameter Generation	Generate FFC parameters	DSA domain parameter generation	DSA parameters	CO	G	C
Key Parameter Validation	Validate FFC parameters	DSA domain parameter validation	DSA parameters	CO	R	P
Show Module Version Information	Provide module version to user	-	-	CO	-	-
Show Status	Provide module status to user	-	-	CO	-	-
Self-test	Run module self-tests on-demand	HMAC	Integrity test key	CO	-	-

Table 8 Approved Services (continued)

¹The 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.

²Algorithms designated as "Legacy" can only be used on data that was generated prior to the Legacy Date specified in FIPS 140-3 IG C.M.

4.2.3 Non-Approved Services

The following is a list of non-approved services provided by the module:

Service	Description	Algorithm Accessed	Role	Indicator¹
Key Derivation	Derive key text given input secret	TLS v1.0/1.1 PRF	CO	C

Table 9 Non-Approved Services

Service	Description	Algorithm Accessed	Role	Indicator¹
Key Encapsulation	Encrypt or decrypt a key with an RSA key encryption key	RSA-PKCS #1	CO	K
Message Digest	Digest a message	MD5	CO	C
Symmetric Decryption	Decrypt with symmetric cipher	AES with CFB 64-bit mode, DES3 with CBC, CFB, ECB and OFB modes	CO	K
Symmetric Encryption	Encrypt with symmetric cipher	AES with CFB 64-bit mode, DES3 with CBC, CFB, ECB and OFB modes	CO	K

Table 9 Non-Approved Services

¹The 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.

4.3 Operation Authentication

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

5 Software/Firmware Security

This section covers integrity measures to demonstrate protection of the software component of BSAFE Crypto Module, which is the whole of the module.

5.1 Approved 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.

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. 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 On Demand Integrity Test Method

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

5.3 Executable Module Form

The module is built as either a single shared library, a dynamic link library or an object file that exports symbols for the operations it supports.

6 Operational Environment

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

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

6.1 Compliance

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 Laboratory Validated Operational Environments

For FIPS 140-3 validation, the module is tested by an accredited FIPS 140-3 testing laboratory. For the tested operational environments, refer to [Table 2, Tested Operational Environments](#).

6.3 Affirmation of Compliance for Other Operational Environments

For the vendor affirmed operational environments, refer to [Table 3, Vendor Affirmed Operational Environments](#).

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

6.4 Configuration Restrictions

The module runs on a General Purpose Computer running one of the operational environments listed in **Tested Operational Environments** and **Vendor Affirmed Operational Environments**.

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

BSAFE Crypto Module does not implement any non-invasive mitigation techniques.

9 Sensitive Security Parameters Management

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

Name	Description	Type
VM	Memory in the Operational Environment of the module.	Volatile

Table 10 Storage Areas

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

Name	From	To	Format Type	Distribution Type	Entry Type
App Write	Operator Application in TOEPP ¹	VM	Plaintext	Manual	Electronic
App Read	VM	Operator Application in TOEPP	Plaintext	Manual	Electronic

Table 11 SSP Input-Output Methods

¹The module's Tested Operational Environment's Physical Perimeter. The TOEPP for the BSAFE Crypto Module is the host computer.

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

Method	Description	Rationale	Operator Initiated Capability
Immediate	Temporary SSPs are zeroized immediately after use	Intermediate values are zeroised at the end of a calculation	N/A
Implicit	SSPs are zeroized when the cryptographic object is deleted by the module	SSPs are zeroized when the BCM_CTX object is deleted	N/A
Explicit	SSPs are zeroized when the associated key or cryptographic object is deleted by the application	SSPs are zeroized when the application no longer needs them	API call to delete object

Table 12 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:
 - a. `BCM_CIPHER` cryptographic objects with a call to `BCM_cipher_delete()`.
 - b. `BCM_MAC` cryptographic objects with a call to `BCM_mac_delete()`.
 - c. `BCM_DIGEST` cryptographic objects with a call to `BCM_digest_delete()`.
 - d. `BCM_KEY` objects with a call to `BCM_key_delete()`.
 - e. `BCM_PARAM` objects with a call to `BCM_param_delete()`.
 - f. `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()`.

Key / SSP Name / Type	Strength	Security Function and Cert. Number	Generation	Import / Export	Establishment	Storage	Zeroisation	Use & related keys
RSA keys (public key / PSP; private key / CSP)	112 - 150 bits	RSA, KAS-IFC-SSC (A2308)	CKG SP 800-133 Rev. 2. FIPS 186-4 RSA key generation method	App Write, App Read	N/A	VM	Explicit	Signature generation and verification. Key agreement.
DH keys (public key / PSP; private key / CSP)	112 - 200 bits	KAS-FFC-SSC (A2308)	CKG SP 800-133 Rev. 2. SP 800-56A Rev. 3 Safe Primes Key Generation method	App Write, App Read	N/A	VM	Explicit	Shared secret generation
DSA parameters (CSP)	112 and 128 bits	DSA (A2308)	CKG SP 800-133 Rev. 2. FIPS 186-4 DSA domain parameter generation method	App Write, App Read	N/A	VM	Explicit	Key generation (DSA keys), Domain parameter validation
DSA keys (public key / PSP; private key / CSP)	112 and 128 bits	DSA (A2308)	CKG SP 800-133 Rev. 2. FIPS 186-4 DSA key generation method	App Write, App Read	N/A	VM	Explicit	Signature generation and verification
ECC keys (public key / PSP; private key / CSP)	112 - 256 bits	ECDSA, KAS-ECC-SSC (A2308)	CKG SP 800-133 Rev. 2. FIPS 186-4 method for ECDSA key generation, SP 800-56A Rev. 3 method for ECDH key generation	App Write, App Read	N/A	VM	Explicit	Signature generation and verification for ECDSA. Shared secret generation for ECDH.
AES keys (CSP)	128, 192, and 256 bits	AES, KTS (A2308)	CKG, SP 800-133 Rev. 2. Direct output of approved DRBG.	App Write, App Read	N/A	VM	Explicit	Symmetric encryption. Key wrapping.

Table 13 SSPs

Key / SSP Name (continued)/ Type	Strength	Security Function and Cert. Number	Generation	Import / Export	Establishment	Storage	Zeroisation	Use & related keys
MAC secret (CSP)	128 - 256 bits	HMAC, AES (CMAC), AES (GMAC) (A2308)	N/A (Application input)	App Write	N/A	VM	Immediate	MAC generation and verification
KDF secret (CSP)	112 - 256 bits	KDF (PBKDF2), KDF (KBKDF), CVL (SSH-KDF), CVL (TLS v1.2 PRF), CVL (TLS v1.3 PRF), CVL (X9.63 KDF), KDA (HKDF), KDA (One-step KDF) (A2308)	N/A (Application input)	App Write	N/A	VM	Immediate	Key derivation
Derived key text (CSP)	112 - 256 bits	KDF (PBKDF2), KDF (KBKDF), CVL (SSH-KDF), CVL (TLS v1.2 PRF), CVL (TLS v1.3 PRF), CVL (X9.63 KDF), KDA (HKDF), KDA (One-step KDF) (A2308)	PBKDF2, KBKDF, SSH-KDF, TLS v1.2 PRF, TLS v1.3 PRF, X9.63 KDF	App Read	N/A	VM	Immediate	Key derivation
CTR DRBG key value (CSP)	128, 192, and 256 bits	DRBG, CKG (A2308)	Obtained from SP 800-90B compliant entropy source	N/A	N/A	VM	Implicit	Random bit generation
CTR DRBG V value (CSP)	128 bits	DRBG, CKG (A2308)	Obtained from SP 800-90B compliant entropy source	N/A	N/A	VM	Implicit	Random bit generation
HMAC DRBG key value (CSP)	256 bits	DRBG, CKG (A2308)	Obtained from SP 800-90B compliant entropy source	N/A	N/A	VM	Implicit	Random bit generation
HMAC DRBG V value (CSP)	256 bits	DRBG, CKG (A2308)	Obtained from SP 800-90B compliant entropy source	N/A	N/A	VM	Implicit	Random bit generation
DRBG entropy input (CSP)	128, 192, and 256 bits	DRBG (A2308)	Obtained from SP 800-90B compliant entropy source	N/A	N/A	VM	Implicit	Random bit generation
DRBG seed (CSP)	128, 192, and 256 bits	DRBG (A2308)	Obtained from SP 800-90B compliant entropy source	N/A	N/A	VM	Implicit	Random bit generation
Entropy state (CSP)	256 bits	DRBG (A2308)	Internal state of SP 800-90B compliant entropy source. Obtained from noise source.	N/A	N/A	VM	Implicit	Random bit generation
Integrity test key (not considered an SSP)	144 bits	HMAC (A2308)	Built-in constant value.	N/A	N/A	Built into modul e image	N/A	Self-test

Table 13 SSPs (continued)

9.1 Deterministic Random Bit Generator

BSAFE Crypto Module provides the following approved DRBGs for use in both the Approved Mode of Operation and Non-Approved mode:

DRBG	Entropy Obtained (bits)
CTR DRBG with AES-128	128
CTR DRBG with AES-192	192
CTR DRBG with AES-256 ¹	256
HMAC DRBG with SHA2-512	256

Table 14 Approved Random Bit Generators

¹CTR DRBG with AES-256 is the default DRBG.

Use	Details
RSA key-pair generation	Prime generation, and Miller-Rabin prime number testing ¹
DSA and DH key-pair generation	Generation of public and private values ¹
FFC (DH, DSA) domain parameter generation	Prime generation, and Miller-Rabin prime number testing ¹
FFC, ECC and RSA key generation	Blinding of random values
RSA key validation	Prime recovery testing ²
ECC key generation	Generation of private value ¹
Symmetric key generation	Direct generation of symmetric keys
Initialization vector generation	Direct generation of IVs for symmetric encryption
RSA PKCS #1 v1.5 encryption	Generation of random value for message encoding
RSA PKCS #1 PSS signing	Generation of random value for message encoding

Table 15 DRBG Output Uses

¹All seeds for asymmetric key generation are generated using the direct output of the approved DRBG.

²For details refer to SP 800-56B Rev. 2, Appendix C.

9.2 Entropy Sources

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

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

Entropy Sources	Minimum number of bits of entropy	Details
Execution time jitter	256	Instantiation of HMAC DRBG
Execution time jitter	128	Instantiation of CTR DRBG with AES-128
Execution time jitter	192	Instantiation of CTR DRBG with AES-192
Execution time jitter	256	Instantiation of CTR DRBG with AES-256
Execution time jitter	DRBG instantiation bits	Application calls <code>BCM_random_seed()</code>
Execution time jitter	64	Application calls <code>BCM_secure_random_bytes()</code>

Table 16 Non-Deterministic Random Bit Generation Specification

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.

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 entropy 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 bits of entropy to the DRBG state before the DRBG generates output. The DRBG issues a GET call to the Entropy NDRBG for 64 bits of entropy.

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.

9.3 Transition periods

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 are 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 17 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. 1).

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

Table 18 Correspondence between Security Strength, Algorithms, and Key Size

Strength	Symmetric	RSA	EC	Hash	MAC and KDF
192	AES-192	7680	384	SHA2-384 SHA3-384	CMAC-AES GMAC-AES SHA2-224 SHA2-512/224 SHA3-224
256	AES-256	15360	521	SHA2-512 SHA3-512 SHAKE-256	CMAC-AES GMAC-AES SHA2-256 SHA2-512/256 SHA2-384 SHA2-512 SHA3-256 SHA3-384 SHA3-512

Table 18 Correspondence between Security Strength, Algorithms, and Key Size (continued)

10 Self-tests

BSAFE Crypto Module performs a number of 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_state()` control interface returns a state of `BCM_MODULE_STATE_NOT_READY`.

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, the cryptographic algorithm self-tests, and the key generation pair-wise consistency 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.

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	Test Properties	Type	Details
HMAC	HMAC-SHA2-256 signature:32 bytes HMAC secret:18 bytes	KAT	Pre-operational software integrity test executes automatically when the module is loaded into memory.
Entropy source	1024 noise samples	RCT and APT	Pre-operational critical functions test runs when a <code>BCM_CTX</code> objects creates an entropy source.

Table 19 Pre-operational Self-tests

10.1.1 Pre-operational Self-test Notes

If all POST pass, the cryptographic services of the module are enabled and the module can be used. The `BCM_module_state()` control 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 repetition count test (RCT) and adaptive proportion test (APT) are defined in SP 800-90B.

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

10.2 Conditional Self-tests

The following table lists the conditional self-tests:

Algorithm	Test	Type	Details	Condition
AES	128, 192 and 256-bit AES keys	KAT	Encrypt and decrypt self-tests	Before first use of algorithm
AES (CMAC)	128, 192 and 256-bit AES keys	KAT	MAC generation and verification self-tests	Before first use of algorithm
AES (GMAC)	128, 192 and 256-bit AES keys	KAT	MAC generation and verification self-tests	Before first use of algorithm
KTS	128, 192 and 256-bit AES keys	KAT	Wrap and unwrap self-tests	Before first use of algorithm
DRBG (AES-CTR)	128, 192 and 256-bit strength	KAT	Random bit generation self-tests	Before first use of algorithm
DRBG (AES-CTR)	128, 192 and 256-bit strength	Fault-Detection Test	SP 800-90A Rev. 1 health test. Instantiate, generate, reseed, uninstantiate tests.	Before first use of algorithm
DRBG (HMAC SHA2-512)	256-bit strength	KAT	Random bit generation self-test	Before first use of algorithm
DRBG (HMAC SHA2-512)	256-bit strength	Fault-Detection Test	SP 800-90A Rev. 1 health test. Instantiate, generate, reseed, uninstantiate tests.	Before first use of algorithm
KAS-FFC-SSC	Key generated from ffdhe2048	KAT	Shared secret calculation	Before first use of algorithm

Table 20 Conditional Self-tests

Algorithm	Test	Type	Details	Condition
DSA	2048-bit key	KAT	Signature generation and verification self-test	Before first use of algorithm
ECDSA	Key generated from P-224	KAT	Signature generation and verification self-test	Before first use of algorithm
KAS-ECC-SSC	Keys generated from P-224 and K-233	KAT	ECDH and ECDHC shared secret calculations	Before first use of algorithm
HMAC (SHA-1)	HMAC with SHA-1	KAT	MAC generation self-test	Before first use of algorithm
HMAC (SHA2)	HMAC with SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	KAT	MAC generation self-test	Before first use of algorithm
HMAC (SHA3)	HMAC with SHA3-224, SHA3-256, SHA3-384, SHA3-512	KAT	MAC generation self-test	Before first use of algorithm
KDA	HKDF with SHA2-256	KAT	Expand and extract self-test	Before first use of algorithm
KDA	One-step KDF with SHA2-256	KAT	Key derivation self-test	Before first use of algorithm
KDF	PBKDF2 with SHA2-256, 2 iterations, 32 byte output	KAT	Key derivation self-test	Before first use of algorithm
RSA	2048-bit RSA key	KAT	RSA encrypt and decrypt self-tests	Before first use of algorithm
RSA	2048-bit RSA key	KAT	RSA signature and verification self-tests	Before first use of algorithm
SHS (SHA-1)	SHA-1	KAT	Message digest generation self-test	Before first use of algorithm

Table 20 Conditional Self-tests (continued)

Algorithm	Test	Type	Details	Condition
SHS (SHA2)	SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256	KAT	Message digest generation self-test	Before first use of algorithm
SHA3	SHA3-224, SHA3-256, SHA3-384, SHA3-512	KAT	Message digest generation self-test	Before first use of algorithm
SHA3 (SHAKE)	SHAKE-128, SHAKE-256	KAT	XOF generation self-test	Before first use of algorithm
CVL (SSH-KDF)	SSH KDF with SHA-1	KAT	Key derivation self-test	Before first use of algorithm
CVL (TLS v1.2 PRF)	TLS v1.2 PRF with SHA2-256	KAT	Key derivation self-test	Before first use of algorithm
CVL (X9.63 KDF)	X9.63 KDF with SHA2-256	KAT	Key derivation self-test	Before first use of algorithm
SP 800-56A Rev. 3 (Safe primes key generation)	2048-bit to 8192-bit keys	PCT	Public key recalculation	Generation of DH key pair
DSA (key generation)	2048-bit and 3072-bit keys	PCT	Sign and verify	Generation of DSA key pair
RSA (key generation)	2048-bit to 4096-bit signing keys	PCT	RSA sign and verify	Generation of RSA signature key pair
RSA (key generation)	2048-bit to 4096-bit encryption keys	PCT	RSA encrypt and decrypt	Generation of RSA encryption key pair

Table 20 Conditional Self-tests (continued)

Algorithm	Test	Type	Details	Condition
ECC (FIPS 186-4 key generation)	Curves B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521	PCT	ECDSA sign and verify	Generation of EC signature key pair
ECC (SP 800-56A Rev. 3 key generation)	Curves B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521	PCT	ECDH public key recalculation	Generation of EC key exchange key pair
Entropy source	1 bit of entropy per byte	Fault-Detection Test	Entropy continuous RCT and APT testing, as defined in SP 800-90B	Creation of new DRBG instance Reseed of DRBG instance Generation of secure random bytes

Table 20 Conditional Self-tests (continued)

10.2.1 Conditional Self-test Notes

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-tests fail then the entropy collection operation returns an error indicator through a return status code. The error cannot be cleared from the NDRBG object. A new NDRBG object must be created to collect entropy.

10.3 Error States

The following table lists the error states:

State Name	Description	Indicator
Self-test Error	Module pre-operational integrity test failure	Cryptographic services return <code>BCM_ERROR_FIPS_INTEGRITY_FAILURE</code> error code. The <code>BCM_module_state()</code> control interface returns a state of <code>BCM_MODULE_STATE_INTEGRITY_FAILED</code> .
Self-test Error	CAST failure	Cryptographic services return <code>BCM_ERROR_FIPS_SELFTEST_FAILURE</code> error code. The <code>BCM_module_state()</code> control interface returns a state of <code>BCM_MODULE_STATE_SELFTEST_FAILED</code> .
Self-test Error	On-demand integrity test failure	Cryptographic services return <code>BCM_ERROR_FIPS_INTEGRITY_FAILURE</code> error code. The <code>BCM_module_state()</code> control interface returns a state of <code>BCM_MODULE_STATE_INTEGRITY_FAILED</code> .
Self-test Error	On-demand CAST failure	Cryptographic services return <code>BCM_ERROR_FIPS_SELFTEST_FAILURE</code> error code. The <code>BCM_module_state()</code> control interface returns a state of <code>BCM_MODULE_STATE_SELFTEST_FAILED</code> .
Self-test Error	On-demand asymmetric key PCT failure	Cryptographic services return <code>BCM_ERROR_FIPS_SELFTEST_FAILURE</code> error code. The <code>BCM_module_state()</code> control interface returns a state of <code>BCM_MODULE_STATE_SELFTEST_FAILED</code> .

Table 21 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.

11 Life-cycle Assurance

11.1 Installation, Initialization, Startup, Operation and Maintenance

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

There are no specific initialization steps required for the module.

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](#).

11.1.4 Maintenance

Maintenance applies only to the application maintainers. If modifications are made to the application, such as a new version or patch to the application, the module's pre-operational 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.

11.2 Crypto Officer Guidance

For details of the administrative functions, security parameters, and logical interfaces available to the Crypto Officer, refer to [Crypto Officer Role](#).

The following sections detail the requirements for algorithm use in the Approved Mode of Operation.

11.2.1 Module Management

General Crypto Officer Guidance

Users should take care to zeroize SSPs when they are no longer needed. BSAFE Crypto Module objects should be deleted when no longer needed, which will zeroize sensitive data managed by the module. User variables containing SSP data on the stack or heap should be zeroized after use or before going out of scope.

11.2.2 Random Number Generation Operations

Using the CTR DRBG with AES-256 is recommended as it is fast and because it provides 256 bits of cryptographic strength, it is suitable for all purposes. This is the module default DRBG.

11.2.3 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.

11.2.4 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.H 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.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.

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.

Triple DES

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

11.2.5 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 22 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 23 Supported DSA key pair sizes

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

Table 24 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 25 Approved RSA modulus length for digital signatures

11.2.6 Digital Signatures

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 22, 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 23, 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.

11.2.7 Message Authentication Code Operations

HMACs

The key length for an HMAC generation or verification must be between 112 and 256 bits, inclusive.

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

11.2.8 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.

11.2.9 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 22, 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 24, 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.

11.2.10 Key Transport Schemes

Key Wrapping using AES

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.

11.2.11 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.

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](#).

13 Acronyms and terms

The following table lists the acronyms and terms used with the module and their definitions:

Term	Definition
AES	Advanced Encryption Standard. A fast symmetric key algorithm with a 128-bit block, and keys of lengths 128, 192, and 256 bits. AES replaces DES as the US symmetric encryption standard.
API	Application Programming Interface.
Attack	An attempt, either a successful or unsuccessful, to break part or all of a cryptosystem. Attack types include an algebraic attack, birthday attack, brute force attack, chosen ciphertext attack, chosen plaintext attack, differential cryptanalysis, known plaintext attack, linear cryptanalysis, and middle person attack.
CAST	Cryptographic Algorithm Self-Tests. A test of all cryptographic functions of an approved cryptographic algorithm that must be performed prior to the first operational use of the cryptographic algorithm. A CAST may be a KAT, a comparison test or a fault-detection test.
CBC	Cipher Block Chaining. A mode of encryption in which each ciphertext depends upon all previous ciphertexts. Changing the IV alters the ciphertext produced by successive encryptions of an identical plaintext.
CCM	Counter with Cipher block chaining Message authentication code. A mode of encryption combining the Counter mode of encryption with Cipher Block Chaining Message Authentication Code (CBC-MAC) for authentication.
CFB	Cipher Feedback. A mode of encryption producing a stream of ciphertext bits rather than a succession of blocks. In other respects, it has similar properties to the CBC mode of operation.
CMAC	Cipher-based Message Authentication Code. An algorithm for calculating a MAC using a block cipher and a secret key, providing verification of both message integrity and authenticity.
CMVP	Cryptographic Module Validation Program.
CSP	Critical Security Parameters. Security related information, such as keys or passwords, whose disclosure or modification can compromise security.
CTR	Counter Mode. A mode of encryption which turns a block cipher into a stream cipher. It generates the next keystream block by encrypting successive values of a counter.
CTR DRBG	Counter mode Deterministic Random Bit Generator.
Decryption	The conversion of encrypted data, ciphertext, into its original form. Generally, the reverse of encryption.
DES	Data Encryption Standard. A symmetric encryption algorithm with a 56-bit key with eight parity bits.
DES3	Triple Data Encryption Standard. A symmetric encryption algorithm with three 56-bit keys, with eight parity bits. Also known as Triple-DES and TDEA.

Table 26 Acronyms and Definitions

Term	Definition
DH	Diffie-Hellman. An asymmetric key exchange algorithm. There are many variants, but typically two entities exchange some public information (for example, public keys or random values) and combine them with their own private keys to generate a shared session key. As private keys are not transmitted, eavesdroppers are not privy to all of the information comprising the session key.
DRBG	Deterministic Random Bit Generator.
DSA	Digital Signature Algorithm. An algorithm for creating digital signatures.
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography. A public-key cryptographic approach based on the algebraic structure of elliptic curves over finite fields. It has the advantage of allowing smaller keys to provide equivalent security.
ECB	Electronic Codebook. A mode of encryption which divides a message into blocks and encrypts each block separately.
ECDH	Elliptic Curve Diffie-Hellman. A key agreement protocol that provides the means for two parties, each with an elliptic-curve public/private key pair, to establish a shared secret over an insecure channel.
ECDHC	Elliptic Curve Diffie-Hellman with Cofactor key agreement algorithm. This algorithm employs the CDH primitive.
ECDSA	Elliptic Curve Digital Signature Algorithm. A variant of DSA that uses elliptic curve cryptography.
Encryption	The transformation of plaintext into an apparently less readable form, called ciphertext, using a mathematical process. The ciphertext can be read by anyone who has the key and decrypts (undoes the encryption) the ciphertext.
FFC	Finite Field Cryptography. The public-key cryptographic methods using operations in a multiplicative group of a finite field. See SP 800-56A Rev. 3 . Both DH and DSA are based on Finite Field Cryptography.
FIPS	Federal Information Processing Standards.
GCM	Galois/Counter Mode. A mode of encryption combining the Counter mode of encryption with Galois field multiplication for authentication.
GMAC	Galois Message Authentication Mode. An authentication-only variant of GCM.
HKDF	HMAC-based Extract-and Expand KDF. HKDF is a two-step key derivation function, where the first step, extraction, transforms a shared secret into a key-derivation key. The second step, expansion, uses the key-derivation key to derive an output key.
HMAC	Keyed-Hashing for Message Authentication Code.
HMAC DRBG	HMAC Deterministic Random Bit Generator.
IV	Initialization Vector. Used as a seed value for an encryption operation.
KAT	Known Answer Test.
KBKDF	Key-Based KDF.
KDF	Key Derivation Functions. Deterministic algorithms used to derive cryptographic keying material from a secret value such as a password.

Table 26 Acronyms and Definitions (continued)

Term	Definition
Key	A string of bits used in cryptography, allowing people to encrypt and decrypt data. Can be used to perform other mathematical operations as well. Given a cipher, a key determines the mapping of the plaintext to the ciphertext. The types of keys include distributed key, private key, public key, secret key, session key, shared key, subkey, symmetric key, and weak key.
Key wrapping	A method of encrypting key data for protection on untrusted storage devices or during transmission over an insecure channel.
MAC	Message Authentication Code. A piece of information is attached to a message for verification of the messages authenticity and data integrity.
MD5	A message digest algorithm, which hashes an arbitrary-length input into a 16-byte digest. Designed as a replacement for MD4.
NDRBG	Non-Deterministic Random Bit Generator.
NIST	National Institute of Standards and Technology. A division of the US Department of Commerce (formerly known as the NBS) which produces security and cryptography-related standards.
OAEP	Optimal Asymmetric Encryption Padding. A padding scheme often used with RSA encryption to convert the deterministic RSA encryption scheme into a probabilistic scheme.
OFB	Output Feedback. A mode of encryption in which the cipher is decoupled from its ciphertext.
OS	Operating System.
P_HASH	A function that uses the HMAC-HASH as the core function in its construction. Specified in RFC 2246 and RFC 5246.
PBKDF2	Password-based Key Derivation Function 2. A method of password-based key derivation, which applies a MAC algorithm to derive the key.
PKCS	Public Key Cryptography Standards. A group of public-key cryptography standards devised and published by RSA®.
POST	Pre-Operational Self-Tests.
PRF	Pseudo Random Function. A function that can be used to generate output from a random seed and a data variable such that the output is computationally indistinguishable from truly random output.
privacy	The state or quality of being secluded from the view or presence of others.
private key	The secret key in public key cryptography. Primarily used for decryption but also used for encryption with digital signatures.
PRNG	Pseudo-Random Number Generator.
PSP	Public Security Parameters. Security related public information, for example, public keys, whose modification can compromise the security of the cryptographic module.
PSS	Probabilistic Signature Scheme. A cryptographic signature scheme typically used with RSA signatures.
RNG	Random Number Generator.

Table 26 Acronyms and Definitions (continued)

Term	Definition
RSA	Public key, asymmetric, algorithm providing the ability to encrypt data and create and verify digital signatures. RSA stands for Rivest, Shamir, and Adleman, the developers of the RSA public key cryptosystem.
SHA	Secure Hash Algorithm. An algorithm which creates a unique hash value for each possible input. SHA takes an arbitrary input which is hashed into a 160-bit digest.
SHA-1	A revision to SHA to correct a weakness. It produces 160-bit digests. SHA-1 takes an arbitrary input, which is hashed into a 20-byte digest.
SHA-2	The NIST-mandated successor to SHA-1, to complement the Advanced Encryption Standard. It is a family of message digest algorithms (SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, and SHA2-512/256), which produce digests of 224, 256, 384, 512, 224, and 256 bits respectively.
SHA-3	A family of hash algorithms which includes SHA3-224, SHA3-256, SHA3-384 and SHA3-512 bits. It is an alternative to SHA-2, as no significant attacks on SHA-2 are currently known.
SHAKE	An extendable output function (XOF) that produces a variable digest size.
SSH-KDF	Secure Shell Key Derivation Function. Used by the SSH protocol to derive IVs, encryption keys and integrity keys.
SSP	Sensitive Security Parameters include both Critical Security Parameters (CSP) and Public Security Parameters (PSP).
TLS	Transport Layer Security.
Triple-DES	See DES3.
XTS	XEX-based Tweaked Codebook mode with ciphertext stealing. A mode of encryption used with AES.

Table 26 Acronyms and Definitions (continued)