Cryptographic Module for BIG-IP

version 1.0.2za-fips

FIPS 140-3 Non-Proprietary Security Policy

Last update: November 2024

Prepared by:
atsec information security corporation
4516 Seton Center Parkway, Suite 250
Austin, TX 78759
www.atsec.com

Table of Contents

Copyrights and Trademarks

F5® and BIG-IP® are registered trademarks of F5, Inc.

VMware ESXi™ is a registered trademark of VMware®, Inc.

Intel® Xeon® is a registered trademark of Intel® Corporation.

Dell is a registered trademark of Dell, Inc.

Azure and Hyper-V are registered trademarks of Microsoft

AWS is a trademark of Amazon.com, Inc.

1 General

1.1 Description

This document is the non-proprietary FIPS 140-3 Security Policy that contains the security rules under which the Cryptographic Module for BIG-IP must operate and describes how this module meets the requirements as specified in FIPS PUB 140-3 (Federal Information Processing Standards Publication 140-3) for an Overall Security Level 1 module.

1.2 Security Levels

ISO/IEC 24759 Section 6. [Number Below]	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	Not Applicable
8	Non-invasive Security	Not Applicable
9	Sensitive Security Parameter Management	1
10	Self-tests	1
11	Life-cycle Assurance	1
12	Mitigation of Other Attacks	Not Applicable
Overall Level		1

Table 1 - Security Levels

2 Cryptographic Module Specification

2.1 Description

Purpose of Use:

The Cryptographic Module for BIG-IP (hereafter referred to as “the module”) is a cryptographic library offering various cryptographic mechanisms to be used by OpenSSL application running on BIG-IP Virtual Edition.

The module provides cryptographic services to applications through an Application Program Interface (API). The module also interacts with the underlying operating system via system calls.

Module Type: software

Module Embodiment: multiple-chip standalone

Cryptographic boundary:

The block diagram in Figure 1 shows the module, its interfaces with the operational environment and the delimitation of its cryptographic boundary with red lines.

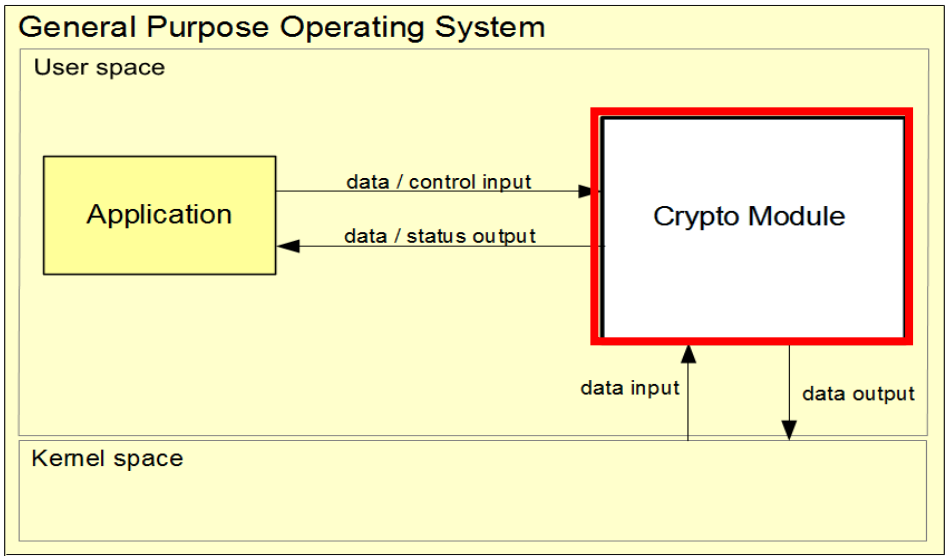

Figure 1 Block Diagram

The cryptographic module boundary consists of two components:

- the shared library, the binary for cryptographic implementations (libcrypto.so.1.0.2za)
- the file that holds the pre-computed integrity check value (.libcrypto.so.1.0.2za.hmac)

Tested Operational Environment’s Physical Perimeter (TOEPP):

The module is aimed to run on a general-purpose computer; the physical perimeter is the surface of the case of the target platform, as shown with orange dotted lines in the diagram Figure 2.

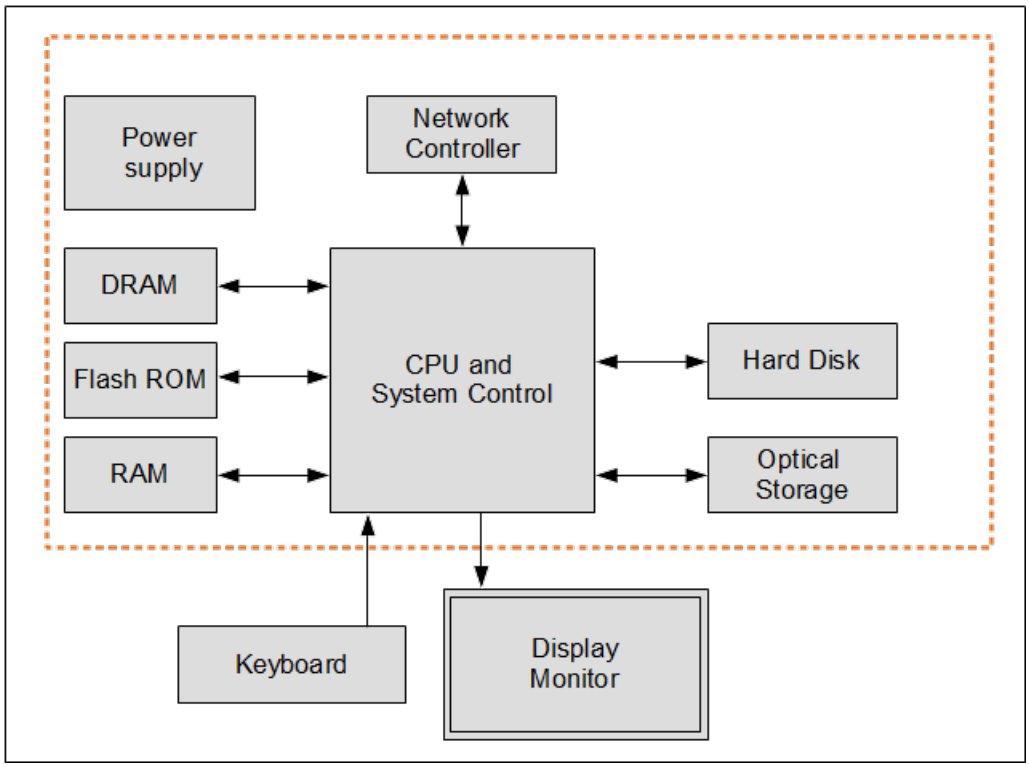

Figure 2 – Cryptographic Module Tested Operational Environment's Physical Perimeter (TOEPP)

2.2 Tested and Vendor Affirmed Module Version and Identification

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

The module version is 1.0.2za-fips.

Tested Operational Environments - Software, Firmware, Hybrid:

#	Operating System (Guest OS)	Hardware Platform	Processors	PAA / PAI	Hypervisor and Host OS
1	BIG-IP 17.1.0.1	Dell PowerEdge R650	Intel® Xeon® Gold Ice Lake 6330N	With and without PAA	VMware ESXi™ 8.0.0.10100 (Build: 20920323)
2	BIG-IP 17.1.0.1	Dell PowerEdge R450	Intel® Xeon® Silver Ice Lake 4309Y	With and without PAA	Hyper-V version 10.0.20348.1 Windows Server 2022 Standard
3	BIG-IP 17.1.0.1	Dell PowerEdge R450	Intel® Xeon® Silver Ice Lake 4309Y	With and without PAA	KVM Ubuntu 22.04.1 LTS

Table 2 - Operational Environments -Software

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

#	Operating System	Hardware Platform
1	BIG-IP 17.1.0.1	Microsoft Corporation Hyper-V Virtual Machine running on Azure CLI 2.48.1 with Intel Xeon Platinum 8272CL

2	BIG-IP 17.1.0.1	Xen 4.2.amazon running on AWS CLI 2.11.19 with Intel Xeon Scalable Processor Cascade Lake 8259CL

Table 3 - Vendor Affirmed Operational Environments

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

2.3 Excluded Components

There are no components within the cryptographic boundary excluded from the FIPS 140-3 requirements.

2.4 Modes of Operation

Modes List and Description:

Name	Description	Type	Status Indicator
Approved mode	Automatically entered whenever an approved service is requested	Approved	Equivalent to the indicator of the requested service
Non-Approved	Automatically entered whenever a non-approved service is requested	Non-Approved	Equivalent to the indicator of the requested service

Table 4 - Modes of Operation

The module becomes operational and enters the approved mode after pre-operational self-tests and conditional algorithm self-tests succeed. No operator intervention is required to reach this point. Once the module is operational, the mode of operation is implicitly assumed depending on the security function invoked and the security strength of the cryptographic keys. Using any non-approved algorithms from Table 8 will put the module in non-approved mode implicitly.

2.5 Algorithms

Approved algorithms:

CAVP Cert[1]	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strengths	Use / Function
A3947, A3948	AES [FIPS 197, SP800-38A, SP800-38D]	ECB, CBC, GCM	128 / 192 / 256-bit AES key / strength from 128 to 256-bits	Symmetric encryption and decryption
A3947	AES [FIPS 197, SP800-38A]	CTR	128 / 192 / 256-bit AES key / strength from 128 to 256-bits	Symmetric encryption and decryption
A3947, A3948	AES [FIPS 197, SP800-38D]	GMAC	128 / 192 / 256-bit AES key / strength from 128 to 256-bits	MAC generation / verification

[1] There are algorithms, modes, and key/moduli sizes that have been CAVP-tested but are not used by any approved service of the module. Only the algorithms, modes/methods, and key lengths/curves/moduli shown in this table are used by an approved service of the module.

CAVP Cert[1]	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strengths	Use / Function
A3947, A3948	KTS (AES) [FIPS 197, SP800-38F, SP800-38D]	GCM	128 / 256-bit AES key / strength from 128 and 256-bits.	Key wrapping
A3947	CTR_DRBG [SP800-90Ar1]	AES-256 in CTR mode, with / without derivation function, prediction resistance enabled and disabled	Entropy input, seed, V and key values / strength is 256-bits	Random number generation
A3948	CTR_DRBG [SP800-90Ar1]	AES 256 in CTR mode, with derivation function, prediction resistance enabled	Entropy input, seed, V and key values / strength is 256-bits	Random number generation
A3947	RSA KeyGen [FIPS 186-4]	Appendix B.3.3 Probable primes with standard key format	2048 / 3072/ 4096-bit modulus size / strength from 112 to 150-bits	RSA key generation
A3947	RSA SigGen [FIPS 186-4]	PKCS 1.5 with SHA-256, SHA-384	2048 / 3072/ 4096-bit modulus / strength from 112 to 150-bits	RSA signature generation
A3947	RSA SigVer [FIPS 186-4]	PKCS 1.5 with SHA2-256, SHA2-384	2048 / 3072/ 4096-bit modulus / strength from 112 to 150-bits	RSA signature verification
A3947	ECDSA KeyGen [FIPS 186-4]	Appendix B.4.2: Testing candidates	ECDSA / ECDH key pair P-256 and P-384 curves / strength 128 and 192-bits	ECDSA/ ECDH key pair generation
A3947	ECDSA KeyVer [FIPS 186-4]	N/A	ECDSA / ECDH key pair with P-256 and P-384 curves / strength 128 and 192-bits	ECDSA/ ECDH public key verification
A3947	ECDSA SigGen [FIPS 186-4]	SHA2-256, SHA2-384	ECDSA P-256, P- 384 curves / strength 128 and 192-bits	ECDSA signature generation
A3947	ECDSA SigVer [FIPS 186-4]	SHA2-256, SHA2-384	ECDSA P-256, P- 384 curves / strength 128 and 192-bits	ECDSA signature verification
A3947, A3948	SHA [FIPS180-4]	SHA-1	N/A	Message digest
A3947	SHA [FIPS180-4]	SHA2-256, SHA2-384	N/A	Message digest
A3947, A3948	HMAC [FIPS 198]	HMAC-SHA-1	112 to 1024-bits with key strengths 112-bits	MAC generation/ verification
A3947	HMAC [FIPS 198]	HMAC-SHA2-256, HMAC-SHA2-384	112 to 1024-bits with key strengths 112 to 192-bits	MAC generation/ verification

CAVP Cert[1]	Algorithm and Standard	Mode / Method	Description / Key Size(s) / Key Strengths	Use / Function
A3947	KAS-ECC-SSC [SP800-56Ar3]	Ephemeral Unified: KAS Role: initiator, responder	P-256, P-384 / strength 128 and 192-bits	EC Diffie-Hellman shared secret computation used in Key Agreement Scheme (KAS) IG D.F scenario 2 (path 1)
A3947	Safe primes key generation [SP800 – 56Ar3]	Safe prime	Safe Prime Groups: ffdhe2048, ffdhe3072, ffdhe4096 / strength from 112 to 150-bits	Safe primes key generation
A3947	KAS-FFC-SSC [SP800-56Ar3]	dhEphemeral: KAS Role: initiator, responder	ffdhe2048, ffdhe3072, ffdhe4096 / strength from 112 to 150-bits	Diffie-Hellman shared secret computation used in KAS IG D.F scenario 2 (path 1)
A3947 (CVL)	SSH KDF [SP800-135r1]	AES-128, AES-256 with SHA2-256, SHA2-384	256-bit keys with 256-bit key strength	Key derivation
A3947 (CVL)	TLS KDF [SP800-135r1]	TLS version 1.0 / 1.1	256-bits	Key derivation
A3947 (CVL)	TLS KDF [SP800-135r1] RFC7627	TLS v1.2	256-bits	Key derivation

Table 5 - Approved Algorithms

Vendor Affirmed Algorithms:

Name	Properties	Implementation	Reference
CKG	Key Type: Asymmetric	Cryptographic Module for BIG-IP	SP800-133r2, Section 4, example 1 and IG D.I.

Table 6 - Vendor Affirmed Algorithms

Non-Approved, Allowed Algorithms:

The module does not implement any Non-Approved Allowed algorithms in the Approved mode of operation.

Non-Approved, Allowed Algorithms with No Security Claimed:

Algorithm	Caveat	Use / Function
MD5	Allowed per IG 2.4.A	Message digest used in TLS 1.0 / 1.1 KDF only

Table 7 - Non-Approved, Allowed Algorithms with No Security Claimed

Non-Approved, Not Allowed Algorithms

Algorithm / Function	Use / Function
AES with OFB, CCM, CFB, XTS, KW modes, Blowfish, Camellia, CAST5, DES, IDEA, RC2, RC4, SEED, SM2, SM4, Triple-DES	Symmetric encryption and decryption
SHA2-224, SHA2-512, SM3, MD4, MDC2, RIPEMD, Whirlpool	Message digest
HMAC-SHA2-224, HMAC-SHA2-512, AES CMAC, Triple-DES CMAC	MAC generation/ verification
RSA key generation	RSA with 1024 and greater than 4096 up to 16384 modulus
RSA signature generation and verification	PKCS #1 v1.5 scheme with 1024 and greater than 4096 up to 16384 modulus, for all SHA sizes
	Probabilistic Signature Scheme (PSS), ANSI X9.31 schemes
	Signature generation/ Signature verification using PKCS #1 v1.5 scheme with modulus size 2048, 3072, 4096-bits with SHA-1, SHA2-224, SHA2-512
ECDSA	Key pair generation using P-224, P-521 curves
	Public key verification using P-224, P-521 curves
	Signature generation / Signature verification using curves P-256, P-384 with SHA-1, SHA2-224, SHA2-512; using P-224, P-521 curves with any SHA sizes
	Signature generation and signature verification using SM2
RSA encrypt / decrypt	RSA with modulus sizes up to 16384-bits
Safe primes key verification	Public key verification using safe prime groups
DSA domain parameter generation, domain parameter verification, key pair generation, signature generation and verification	DSA with all key and SHA sizes
HMAC_DRBG and Hash_DRBG using all SHA sizes, CTR_DRBG with AES-128, AES-192, ANSI X9.31 RNG	Random number generation
Diffie-Hellman	Shared secret computation with groups other than ffdhe2048, ffdhe3072, ffdhe4096
EC Diffie-Hellman	Shared secret computation: - Ephemeral Unified with curves other than P-256, P-384 without KDF. - one PassDh and StaticUnified without KDF.
TLS KDF SSH KDF	Key Derivation function in the context of: - TLS using SHA2-224 / SHA2-512 - SSH using SHA-1 / SHA2-224 / SHA2-512

Table 8 - Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	SF Properties	Algorithms/ CAVP Cert
Diffie-Hellman shared secret computation	KAS	[SP800-56Ar3] shared secret computation used in KAS IG D.F scenario 2 (path 1)	ffdhe2048, ffdhe3072, ffdhe4096 providing from 112 to 150-bits of encryption strength	KAS-FFC-SSC/ A3947
EC Diffie-Hellman shared secret computation	KAS	[SP800-56Ar3] shared secret computation used in KAS IG D.F scenario 2 (path 1)	P-256, P-384 providing 128 and 192-bits of encryption strength	KAS-ECC-SSC/ A3947
AES key wrapping	KTS	[FIPS 197, SP800-38F], IG D.G comment 8, key wrapping and unwrapping, in the context of the TLS protocol, are provided by the TLS record layer using AES-GCM as the approved authenticated encryption mode.	128 / 256-bit AES-GCM keys providing 128 and 256-bits of encryption strength	AES GCM/ A3947, A3948

Table 9 - Security Function Implementations

2.7 Algorithm Specific Information

AES-GCM IV: The Crypto Officer shall consider the following requirements and restrictions when using the module. The IV for AES-GCM is constructed in compliance with IG C.H scenario 1a (TLS 1.2). The module is compliant with SP800-52r2 section 3.3.1 and the mechanism for IV generation is compliant with RFC5288.
The module does not implement the TLS protocol. The module's implementation of AES-GCM is used together with an application that runs outside the module's cryptographic boundary. The design of the TLS protocol implicitly ensures that the counter (the nonce_explicit part of the IV) does not exhaust the maximum number of possible values for a given session key. In the event the module's power is lost and restored, the consuming application must ensure that a new key for use with the AES-GCM key encryption or decryption under this scenario shall be established.

SP800-56Ar3 assurances (IG D.F): To comply with the assurances found in Section 5.6.2 of SP800-56Ar3, the keys for KAS-FFC-SSC and KAS-ECC-SSC must be generated using the approved key generation services specified in section 9.2. For KAS-FFC-SSC the module generates keys using Safe Primes Key Generation with Safe Prime Groups: ffdhe2048, ffdhe3072, ffdhe4096. For KAS-ECC-SSC, the module generates keys using ECDSA KeyGen, with curves P-384 and P-256. The module performs full public key validation on the generated public keys. Additionally, the module performs full public key validation on the received public keys.

RSA modulus size (IG C.F): The module implements FIPS 186-4 RSA KeyGen, SigVer and RSA SigGen with modulus lengths of 2048, 3072, 4096 bits. All these modulus lengths have been CAVP tested.

2.8 RNG and Entropy

The module entropy source specified in Table 10 uses jitter variations caused by executing instructions and memory accessed (see details in Public Use Document referenced in section 11.2). The CPU Jitter 3.4.0 entropy source with SHA-3 vetted conditioning component is located within the physical perimeter of the module but outside the cryptographic boundary of the module.

Name	Type	Operational Environment	Sample size	Entropy per sample	Conditioning Components
CPU Jitter Entropy Source for F5 cryptographic module (SHA_3) #E74	Non-Physical	OEs listed in Table 2.	256 bits	256 bits	SHA3-256 (A2621)

Table 10 - Entropy Source

The module employs a Deterministic Random Bit Generator (DRBG) based on [SP800-90Ar1] for the generation of random value used in asymmetric keys, and for providing a RNG service to calling applications. The approved DRBG provided by the module is the counter DRBG with AES-256.

The output of entropy sources provides 256-bits of entropy to seed and reseed SP800-90Ar1 DRBG during initialization (seed) and reseeding (reseed).

2.9 Key Generation

The module implements RSA, ECDSA, ECDH and DH asymmetric key generation services with the following methods:

- RSA and ECDSA key pairs are generated in accordance with [SP800-133r2] section 5.1 which maps to [FIPS186-4] Appendix B.3.3 (RSA) or Appendix B.4.2 (ECDSA). A seed (i.e. the random value) used in asymmetric key generation is directly obtained from the [SP800-90Ar1] DRBG.
- Diffie-Hellman and EC Diffie-Hellman key pairs are generated in accordance with [SP800-133r2] section 5.2 i.e. key generation method specified in [SP800-56Ar3] section 5.6.1.1.1 (DH) or section 5.6.1.2.1 (ECDH) used by approved key-establishment schemes which maps to [FIPS 186-4].

2.10 Key Establishment

See Table 9, "KAS" row.

2.11 Industry Protocol

The TLS v1.0 / 1.1 / 1.2 protocol has not been reviewed or tested by the CAVP or CMVP. See section 2.7 for details.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

The logical interfaces are the API through which the applications request services. The following table summarizes the logical interfaces.

Physical Port	Logical Interface[2]	Data that passes over port / interface
As a software-only module, the module does not have physical ports. Physical Ports are interpreted to be the physical ports of the hardware platform on which it runs.	Data Input	Data inputs are provided in the variables passed in the API and callable service invocations, generally through caller-supplied buffers
	Data Output	Data outputs are provided in the variables passed in the API and callable service invocations, generally through caller-supplied buffers
	Control Input	Control inputs which control the mode of the module are provided through dedicated parameters.
	Status Output	Status output is provided in return codes and through messages. Documentation for each API lists possible return codes. A complete list of all return codes returned by the C language APIs within the module is provided in the header files and the API documentation. Messages are also documented in the API documentation.

Table 11 - Ports and Interfaces

[2] The module does not implement Control Output interface.

4 Roles, Services, and Authentication

4.1 Authentication Methods

FIPS 140-3 does not require an authentication mechanism for level 1 modules. Therefore, the module does not implement an authentication mechanism for Crypto Officer. The Crypto Officer role is implicitly assumed when accessing all services provided by the module (see Table - Approved Services and Table - Non-Approved Services below).

4.2 Roles

The module supports the Crypto Officer role only. No support is provided for multiple concurrent operators.

Name	Type	Operator Type	Authentication Method
Crypto Officer	Role	CO	N/A

Table 12 - Roles

Table below describes the authorized role in which the service can be performed with specification of the service input parameters and associated service output parameters.

Role	Service	Input	Output
Crypto Officer	Encryption and decryption	Plaintext and key / ciphertext and key	Ciphertext / plaintext
	Key wrapping	Wrapping key, key to be wrapped / Unwrapping key, key and key to be unwrapped	Wrapped key / unwrapped key
	Random number generation	Number of bits	Random numbers
	RSA key pair generation	Key size	Public key, private key
	ECDSA/ ECDH key pair generation	Elliptic curve	Private key, public key
	ECDSA/ ECDH Public key verification	Public key	Pass / fail result of public key verification
	RSA/ ECDSA / DSA/ signature generation	Private key, message, hashing algorithm	Computed signature
	RSA/ ECDSA / DSA signature verification	Public key, digital signature, message, hashing algorithm	Pass / fail result of digital signature verification
	RSA encryption and decryption	Message, key	Ciphertext / plaintext
	DSA key pair generation	Domain parameters	Public key, private key
	DSA domain parameter generation	Prime length and seed length	Domain parameters
	DSA domain parameter verification	Domain parameters	Return codes / log messages
	Safe primes key generation/ verification	Group	Private key, public key
	EC Diffie-Hellman / Diffie-Hellman shared secret computation	Public key, private key	Shared secret

Role	Service	Input	Output
	Message digest	Message, Hashing algorithm	Hashed message
	MAC generation	Message, HMAC key or GMAC key, MAC algorithm, MAC length	Authenticated message
	MAC verification	Authenticated message, HMAC key or GMAC key, MAC algorithm	Pass/fail result of MAC verification
	Key derivation	PRF algorithm, TLS pre-primary secret, TLS primary secret, SSH shared secret, SSH derived key	Derived key
	Show version	N/A	Name and Version information
	Show Status	N/A	Status output
	Self-Tests	N/A	Pass / fail results of self-tests
	Zeroization	Unencrypted SSPs listed in Table 17	None

Table 13 - Roles, Service Commands, Input and Output

4.3 Approved Services

The status output from the FIPS_set_indicator_status service indicator's call is provided in Indicator column in Table 14. To read this indicator, the calling application must register a callback function using 'FIPS_register_indicator_callback'. The callback function shall take the input of the form "char *" which is the form of the indicator being output by the module.

Service Name	Description	Security Function Implementations	Roles SSPs Access	Roles	Indicator
Encryption and decryption	Executes AES- mode encrypt or decrypt operation	AES-ECB, AES-CBC, AES-CTR	128 / 192 / 256-bit AES key: W, E:	Crypto Officer	AES-ECB, AES-CBC, AES-CTR
Key wrapping	Executes AES-GCM key wrapping or unwrapping operation, per IG D.G	AES-GCM encrypt / decrypt	128 / 256-bit AES key: W, E	Crypto Officer	AES-GCM
Random number generation	Generate random number	Counter DRBG	Entropy input: E DRBG seed, V and key values: G, W, E	Crypto Officer	CTR-DRBG-AES-256
RSA key pair generation	Generate RSA key pair	CKG [SP800-133r2] RSA KeyGen [FIPS 186-4] Counter DRBG	RSA private key, RSA public key (2048 / 3072 / 4096-bits): G, R	Crypto Officer	RSA-KEY-GEN-2048, RSA-KEY-GEN-3072 RSA-KEY-GEN- 4096
RSA signature generation	Sign a message with a specified RSA private key.	RSA SigGen [FIPS 186-4] with SHA2-256, SHA2-384	RSA private key (2048 / 3072 / 4096-bits): W, E	Crypto Officer	RSA-SIG
RSA signature verification	Verify the signature of a message with a specified RSA public key.	RSA SigVer [FIPS 186-4] with SHA2-256, SHA2-384	RSA public key (2048 / 3072 / 4096-bits): W, E	Crypto Officer	RSA-VER

Service Name	Description	Security Function Implementations	Roles SSPs Access	Roles	Indicator
ECDSA / EC Diffie-Hellman key pair generation	Generate a key pair for a requested elliptic curve	CKG [SP800-133r2] ECDSA KeyGen [FIPS 186-4] Counter DRBG	ECDSA / EC Diffie-Hellman private key, ECDSA / EC Diffie-Hellman public key (P-256 and P-384 curves, key pair): G, R	Crypto Officer	EC-KEYGEN-P-256, EC-KEYGEN-P-384
ECDSA / EC Diffie-Hellman public key verification	Public key verification	ECDSA KeyVer [FIPS 186-4]	ECDSA / EC Diffie-Hellman public key (P-256 and P-384 curves): E, W	Crypto Officer	EC-KEY-VERIFY-P-256, EC-KEY-VERIFY-P-384
ECDSA signature generation	Sign a message with a specified ECDSA private key.	ECDSA SigGen [FIPS 186-4] (SHA2-256, SHA2-384)	ECDSA private key (P-256 and P-384 curves): W, E	Crypto Officer	ECDSA-SIGN-P-256, ECDSA-SIGN-P-384
ECDSA signature verification	Verify the signature of a message with a specified ECDSA public key.	ECDSA ECDSA SigVer [FIPS 186-4] (SHA2-256, SHA2-384)	ECDSA public key (P-256 and P-384 curves): W, E	Crypto Officer	ECDSA-VERIFY-P-256, ECDSA-VERIFY-P-384
EC Diffie-Hellman shared secret computation IG D.F scenario 2, path 1	Calculate a shared secret via the EC Diffie-Hellman algorithm.	KAS-ECC-SSC SP800-56Ar3	EC Diffie-Hellman private key (P-256 and P-384 curves): W, E EC Diffie-Hellman shared secret: G, R EC Diffie-Hellman public key (remote peer public key) (P-256 and P-384 curves): W, E	Crypto Officer	ECDH-COMPUTE-KEY-P-256, ECDH-COMPUTE-KEY-P-384
Safe primes key generation	Generate a key pair	Safe primes key generation	Diffie-Hellman public key (ffdhe2048, ffdhe3072, ffdhe4096): G, R Diffie-Hellman private key (ffdhe2048, ffdhe3072, ffdhe4096): G, R	Crypto Officer	FFDHE2048-KEYGEN, FFDHE3072-KEYGEN, FFDHE4096-KEYGEN
Diffie-Hellman shared secret computation IG D.F scenario 2, path 1	Calculate a shared secret via the Diffie-Hellman algorithm.	KAS-FFC-SSC SP800-56Ar3	Diffie-Hellman private key (ffdhe2048, ffdhe3072, ffdhe4096): W, E Diffie-Hellman shared secret: G, R Diffie-Hellman public key (remote peer public key) (ffdhe2048, ffdhe3072, ffdhe4096): W, E	Crypto Officer	FFDHE2048-COMPUTE, FFDHE3072- COMPUTE, FFDHE4096- COMPUTE

Service Name	Description	Security Function Implementations	Roles SSPs Access	Roles	Indicator
Message digest	Generate a digest for the requested algorithm	SHA-1, SHA2-256, SHA2-384	N/A	Crypto Officer	MESSAGE-DIGEST-SHA-1 MESSAGE-DIGEST--SHA-256 MESSAGE-DIGEST--SHA-384
MAC generation / verification	Generate / verify an HMAC or GMAC digest using the requested SHA algorithm or AES algorithm as appropriate	HMAC-SHA-1, HMAC-SHA2-256, HMAC-SHA2-384, AES-GMAC	HMAC key, AES-key: W, E	Crypto Officer	MSG-AUTH-HMAC-SHA-1, MSG-AUTH-HMAC-SHA-256 MSG-AUTH-HMAC-SHA-384 AES-GMAC
Key derivation	Deriving TLS keys	TLS KDF v1.0 / 1.1 / 1.2	TLS pre-primary secret: W, E TLS primary secret: W, E, G TLS derived key: G	Crypto Officer	TLS-P-HASH-DERIVATION-SHA-1 TLS-P-HASH-DERIVATION-SHA-256 TLS-P-HASH-DERIVATION SHA-384
Key derivation	Deriving SSH keys	SSHv2 KDF	SSH shared secret: W, E SSH derived key: G	Crypto Officer	SSH-KEY-HASH-DERIVATION-SHA-256 SSH-KEY-HASH-DERIVATION SHA-384
Show version	Return the SW version and the module's name	N/A	N/A	Crypto Officer	None
Show Status	Return the module status	N/A	N/A	Crypto Officer	None
Self-tests	Execute integrity test. Execute the CASTs	Integrity test, CASTs from Table 19	N/A (key for self-tests are not SSPs)	Crypto Officer	None
Zeroization	Zeroize all non-protected SSPs	N/A	All SSPs: Z	Crypto Officer	None

Table 14 - Approved Services

G = **Generate**: The module generates or derives the SSP.

R = **Read**: The SSP is read from the module (e.g. the SSP is output).

W = **Write**: The SSP is updated, imported, or written to the module.

E = **Execute**: The module uses the SSP in performing a cryptographic operation.

Z = **Zeroise**: The module zeroises the SSP.

4.4 Non-Approved Services

Service Name	Description	Algorithms Accessed	Role	Indicator
Symmetric encryption and decryption	Encryption / decryption	AES with OFB, CFB, CCM, XTS, KW modes; Triple-DES; Blowfish; Camellia; CAST5; DES; IDEA; RC2; RC4; SEED; SM2; SM4	Crypto Officer	None
Message digest	Generating message digest	SHA2-224, SHA2-512, SM3, MD4, MDC2, RIPEMD, Whirlpool		

Service Name	Description	Algorithms Accessed	Role	Indicator
Message authentication code generation and verification	MAC computation	HMAC-SHA2-224, HMAC-SHA2-512 AES CMAC, Triple-DES CMAC		
RSA key pair generation	Generating key pair	RSA KeyGen with 1024, greater than 4096 and up to 16384 modulus		
RSA signature generation and verification	Generating signature, Verifying signature	RSA SigGen / SigVer PKCS #1 v1.5 with key other than the one listed in Table 5		
		RSA SigGen / SigVer PSS, X9.31 schemes		
		RSA SigGen/ SigVer PKCS #1 v1.5 scheme with modulus size 2048, 3072, 4096-bits with SHA-1, SHA2-224, SHA2-512		
ECDSA key pair generation	Generating key pair	ECDSA KeyGen using P-224, P-521 curves		
ECDSA public key verification	Verifying public key	ECDSA KeyVer using P-224, P-521 curves		
ECDSA signature generation & verification	Generating signature, Verifying signature	ECDSA SigGen and ECDSA SigVer using P-256, P-384 curves with SHA-1, SHA2-224 and SHA2-512		
		ECDSA SigGen and ECDSA SigVer using SM2 algorithm		
		ECDSA SigGen and ECDSA SigVer using P-224, P-521 curves with any SHA sizes		
RSA encryption / decryption	Encryption / decryption	RSA with modulus size up to 16384-bits		
Safe primes key verification	Public key verification	Public key verification using safe prime groups		
DSA domain parameter generation, domain parameter verification, key pair generation, signature generation and verification	Generating domain parameter, Verifying domain parameters, Generating key pair, Generating signature, Verifying signature	DSA with all key and SHA sizes		
Random number generation	Generating deterministic random number	Using HMAC_DRBG and Hash_DRBG for all SHA sizes		
		CTR_DRBG with AES-128 or AES-192		
		ANSI X9.31 RNG		
Shared secret computation	Calculating shared secret	Diffie-Hellman key agreement without KDF with groups other than ffdhe2048, ffdhe3072, ffdhe4096 EC Diffie-Hellman Ephemeral Unified with P-224, P-521 curves without KDF EC Diffie-Hellman one Pass and Static without KDF		
Key derivation (TLS)	Deriving TLS key	Key Derivation using SHA2-224 / SHA2-512		
Key derivation (SSH)	Deriving SSH key	Key Derivation function using SHA-1 / SHA2-224 / SHA2-512		

Table 15 - Non-Approved Services

5 Software / Firmware Security

5.1 Integrity Techniques

The integrity of the module is verified by comparing a HMAC value calculated at run time on the libcrypto.so.1.0.2za file, using HMAC key embedded in the same file, with the HMAC-SHA2-256 value stored in the module file .libcrypto.so.1.0.2za.hmac that was computed at build time.

Integrity tests are performed as part of the Pre-Operational Self-Tests.

5.2 Initiate on Demand

The on-demand integrity test is performed by power-cycling or reloading the module.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Operational Environment Type:

The module operates in a modifiable operational environment. BIG-IP consists of a Linux based operating system customized for performance that runs directly on the hardware or in virtual environment.

Operational Environment Requirements:

The module runs on a BIG-IP 17.1.0.1 operating system executing on the hardware and hypervisor specified in Table 2.

Configuration Settings and Restrictions:

The module should be installed as stated in section 11. The operator should confirm that the module is installed correctly by section following guidance in section 11.1 and section 11.2.

7 Physical Security

The module is a software and therefore this section is Not Applicable (N/A).

8 Non-invasive Security

Per IG 12.A: Until requirements of SP800-140F are defined, non-invasive mechanisms fall under ISO / IEC 19790:2012 Section 7.12 Mitigation of other attacks.

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Name	Description	Persistence Type
RAM	The memory occupied by SSPs is allocated by regular memory allocation operating system calls. SSPs are stored in plaintext	Dynamic

Table 16 - Storage Areas

9.2 SSP Input-Output Methods

The module does not support manual SSP entry or intermediate key generation output. The module does not support entry and output of SSPs beyond the physical perimeter of the operational environment (TOEPP). The SSPs can be provided to the module in plaintext form via API parameters, to and from the calling application running on the same operational environment. This is allowed by [FIPS 140-3_IG] IG 9.5.A Table 1, according to the “CM Software to/from App via TOEPP Path” entry which refers to keys communicated within the physical perimeter of the GPC.

9.3 SSP Zeroization Methods

The application is responsible for calling the appropriate destruction functions provided in the module's API. The destruction functions (listed in Table 17) overwrite the memory occupied by keys with “zeros” and deallocate the memory with the regular memory deallocation operating system call.

9.4 SSPs

SSP Name / Type	Strength	Security Function / Cert. #	Generated By	Import / Export	Established By	Storage	Zeroization	Use and related keys
AES key / CSP / symmetric	128 to 256-bits	ECB, CBC, CTR, GCM A3947 ECB, CBC, GCM A3948	N/A	Input as an API parameter; No export	N/A	RAM	EVP_CIPHER_CTX_cleanup	**Use**: Encryption / decryption Related keys: N/A
AES key / CSP / symmetric	128 to 256-bits	GMAC, A3947 A3948	N/A	Input as an API parameter; No export	N/A	RAM	EVP_CIPHER_CTX_cleanup	**Use**: MAC generation and verification Related keys: N/A
AES key / CSP / symmetric	128 and 256-bits	AES GCM key wrapping A3947 A3948	N/A	Input as an API parameter; No export	N/A	RAM	FIPS_cipher_ctx_cleanup()	**Use**: Key wrapping **Related keys**: N/A
HMAC key / CSP /	112 to 192-bits	HMAC-SHA-1, HMAC-	N/A	Input as an API parameter; No export	N/A	RAM	HMAC_CTX_	**Use**: MAC generation/ verification

SSP Name / Type	Strength	Security Function / Cert. #	Generated By	Import / Export	Established By	Storage	Zeroization	Use and related keys
symmetric		SHA-256, HMAC-SHA-384: A3947 HMAC-SHA-1 A3948					clean up()	Related keys: N/A
RSA private key / CSP / asymmetric	112 to 150-bits	RSA SigGen: A3947	Generated conformant to SP800-133r2 section 4 example 1 (CKG)	Import / Export: CM to / from TOEPP Path. Passed to / from the module via API parameters in plaintext format.	N/A	RAM	FIPS_rsa_free()	**Use**: Digital signature generation **Related keys**: RSA public key, DRBG internal state
RSA public key / PSP/ asymmetric		RSA SigVer: A3947						**Use**: Digital signature verification **Related keys**: RSA private key, DRBG internal state
ECDSA private key / CSP / asymmetric	128 and 192-bits	ECDSA SigGen: A3947	Generated conformant to SP800-133r2 section 4 example 1 (CKG)	Import / Export: CM to / from TOEPP Path. Passed to / from the module via API parameters in plaintext format.	N/A	RAM	EC_KEY_free()	**Use**: Digital signature generation **Related keys**: ECDSA public key, DRBG internal state
ECDSA public key / PSP / asymmetric		ECDSA SigVer: A3947						**Use**: Digital signature verification **Related keys**: ECDSA private key, DRBG internal state
EC Diffie-Hellman private key / CSP / asymmetric	128 and 192-bits	KAS-ECC-SSC Sp800-56Ar3 A3947	Generated conformant to SP800-133r2 section 4 example 1	Import / Export: CM to / from TOEPP Path. Passed to / from the module via API parameters in plaintext format.	N/A	RAM	EC_KEY_free(); EC_POINT_free()	**Use**: EC Diffie-Hellman shared secret computation **Related keys**: EC Diffie-Hellman public key, DRBG internal state, EC Diffie-Hellman shared secret
EC Diffie-Hellman public key / PSP / asymmetric								**Use**: EC Diffie-Hellman shared secret computation **Related keys**: EC Diffie-Hellman private key, DRBG internal state, EC Diffie-Hellman shared secret
EC Diffie-Hellman shared secret /	128 and 192-bits	KAS-ECC-SSC Sp800-56Ar3: A3947	N/A	No import; Export: CM to TOEPP Path. Passed from the module via API parameters	SP800-56Ar3 KAS in Table 9	RAM	EC_KEY_free() EC_POINT_free()	**Use**: EC Diffie-Hellman shared secret computation **Related keys**: EC Diffie-Hellman private key, EC

SSP Name / Type	Strength	Security Function / Cert. #	Generated By	Import / Export	Established By	Storage	Zeroization	Use and related keys
CSP / asymmetric				in plaintext format.				Diffie-Hellman public key
Diffie-Hellman private key / CSP / asymmetric	112 to 150-bits	KAS-FFC-SSC Sp800-56Ar3: A3947	Generated conformant to SP800-133r2 section 4 example 1 (CKG) Table 6	Import / Export: CM to / from TOEPP Path. Passed to / from the module via API parameters in plaintext format.	N/A	RAM	DH_free	**Use**: DH shared secret computation **Related keys**: Diffie-Hellman public key, DRBG internal state
Diffie-Hellman public key / PSP / asymmetric								**Use**: DH shared secret computation **Related keys**: Diffie-Hellman private key, DRBG internal state
Diffie-Hellman shared secret / CSP / asymmetric	112 to 150-bits	KAS-FFC-SSC Sp800-56Ar3: A3947	N/A	No import; Export: CM to TOEPP Path. Passed from the module via API parameters in plaintext format.	SP800-56Ar3 KAS in Table 9	RAM	DH_free	**Use**: DH shared secret computation **Related keys**: Diffie-Hellman private key, Diffie-Hellman public key
TLS pre-primary secret	DH ffdhe2048, ffdhe3072, ffdhe4096 / 112, 128, 150-bits ECDH: P-256, P-384/ 128 and 192-bits	TLS KDF A3947	N/A	Import; as an API parameter No export	SP800-56Ar3 KAS in Table 9	RAM	OPENSSL_cleanse	**Use**: TLS KDF **Related SSPs**: EC Diffie-Hellman private and public keys and Diffie-Hellman private and public keys; TLS primary secret
TLS primary secret	256-bits	TLS KDF A3947	SP800-135r1 TLS KDF	No import; No export	N/A	RAM	OPENSSL_cleanse	**Use**: TLS KDF **Related SSPs**: TLS pre-primary secret; TLS derived key
TLS derived key (AES HMAC)	128 and 256-bits (AES) 112 and 256-bits (HMAC)	AES HMAC A3947	SP800-135r1 TLS KDF	No import; Export: Passed from the module via API parameters in plaintext format	N/A	RAM	OPENSSL_free	**Use**: TLS protocol **Related SSPs**: TLS pre-primary secret, TLS primary secret
SSH shared secret	ECDH: P-256, P-384 / 128	SSH KDF A3947	N/A	Import; as an API parameter No export	SP800-56Ar3	RAM	OPENSSL_free	**Use**: Key derivation; **Related SSPs**: EC Diffie-Hellman

SSP Name / Type	Strength	Security Function / Cert. #	Generated By	Import / Export	Established By	Storage	Zeroization	Use and related keys
	and 192-bits				KAS in Table 9			private and public keys; SSH derived key
SSH derived key (AES, HMAC)	128 and 256-bits (AES) 112 and 256-bits (HMAC)	AES HMAC A3947	SP800-135r1 SSH KDF	No import; Export: Passed from the module via API parameters in plaintext format	N/A	RAM	OPENSSL_free	**Use**: data encryption / decryption and MAC calculations in SSH protocol **Related SSPs**: SSH shared secret
Entropy input / CSP (IG D.L)	256-bits	ESV E74	Obtained from ESV E74 (reference in section 11.2)	Import from the OS within the TOEPP; No Export	N/A	RAM	When the system is powered down	**Use**: Random number generation Related keys: DRBG seed
DRBG seed / CSP (IG D.L)	256-bits	Counter DRBG A3947 A3948	Derived from the entropy string as defined by [SP800-90Ar1]	No import: it remains within the cryptographic boundary; No Export	N/A	RAM	FIPS_drbg_uninstantiate	**Use**: Random number generation. **Related keys**: Entropy input, DRBG, Internal state
DRBG internal state (V and Key values) / CSP (IG D.L)	256-bits	Counter DRBG A3947 A3948	Derived from the DRBG seed as defined by [SP800-90Ar1]	No import: it remains within the cryptographic boundary; No Export	N/A	RAM	FIPS_drbg_uninstantiate	**Use**: Random Number Generation. **Related keys**: Entropy input, DRBG seed

Table 17 - SSPs

10 Self-tests

While the module is executing the pre-operational self-test, CASTs and PCTs, services are not available, and data/control input and data output are inhibited. The module does not return control to the calling application until the self-tests are completed. On successful completion of the CASTs and pre-operational self-tests (integrity test), the module enters operational mode and cryptographic services are available. If the module fails any of the tests, it will return an error code and enter into the Error state to prohibit any further cryptographic operations.

10.1 Pre-Operational Self-Tests

Pre-operational self-tests are performed automatically when the module is loaded into memory; the pre-operational self-tests ensure that the module is not corrupted and that the cryptographic algorithms work as expected.

Algorithm	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA-256	112-bit key	Message Authentication	Integrity	Module becomes operational	Integrity of the module is verified by comparing the HMAC-SHA2-256 value calculated at runtime with the HMAC-SHA2-256 value stored in the module that was computed at build time

Table 18 - Pre-Operational Self-Tests

10.2 Conditional Self-Tests

Algorithm	Test Properties	Test Method	Type	Indicator	Details	Condition
CTR_DRBG	AES 256-bits with derivation function	KAT	CAST	Module becomes operational	SP800-90Ar1 section 11.3 health tests	Test runs at Power-on before the integrity test
AES-GCM; AES-CBC	128-bit key	KAT	CAST	Module becomes operational	Encryption / decryption	Test runs at Power-on before the integrity test
RSA PKCS#1 v1.5	2048-bit key and SHA2-256	KAT	CAST	Module becomes operational	Signature generation / signature verification	Test runs at Power-on before the integrity test
RSA	2048-bit key and SHA2-256	PCT	PCT	Asymmetric algorithm is performed	Calculation and verification of a digital signature	Test runs at first algorithm use
ECDSA	P-256 and SHA2-256	KAT	CAST	Module becomes operational	Signature generation / signature verification	Test runs at Power-on before the integrity test
ECDSA	P-256 and SHA2-256	PCT	PCT	Asymmetric algorithm is performed	Calculation and verification of a digital signature	Test runs at first algorithm use

Algorithm	Test Properties	Test Method	Type	Indicator	Details	Condition
KAS-ECC-SSC	P-256	KAT	CAST	Module becomes operational	Shared secret computation	Test runs at Power-on before the integrity test
Diffie Hellman	ffdhe2048	PCT	PCT	Asymmetric algorithm is performed	Section 5.6.2.1.4 of SP800-56Ar3	Test runs at first algorithm use
KAS-FFC-SSC	ffdhe2048	KAT	CAST	Module becomes operational	Shared secret computation	Test runs at Power-on before the integrity test
EC Diffie Hellman	P-256	PCT	PCT	Asymmetric algorithm is performed	Covered by ECDSA PCT (IG 10.3.A)	Test runs at first algorithm use
HMAC-SHA	HMAC-SHA-1, HMAC-SHA2-256 HMAC-SHA2-384	KAT	CAST	Module becomes operational	MAC	Test runs at Power-on before the integrity test
SHA	SHA-1, SHA2-256, SHA2-384	KAT	CAST	Module becomes operational	Covered by respective HMAC KATs (allowed per IG 10.3.B)	Test runs at Power-on before the integrity test
[SP800-135r1] KDF	TLS	KAT	CAST	Module becomes operational	Key derivation used in the TLS v 1.0 / 1.1 / 1.2 protocols	Test runs at Power-on before the integrity test
[SP800-135r1] KDF	SSH	KAT	CAST	Module becomes operational	Key derivation used in the SSHv2 protocol	Test runs at Power-on before the integrity test

Table 19 - Conditional Self-Tests

The entropy source performs its required self-tests; those are not listed in Table 19, as the entropy source is not part of the cryptographic boundary of the module.

10.3 Periodic Self-Test Information

The periodic self-tests can be invoked by powering-off and reloading the module. This service performs Conditional Cryptographic Algorithm Self-Tests (CASTs) and integrity test. During the execution of the periodic self-tests, crypto services are not available and no data output or input is possible.

10.4 Error States

State Name	Description	Conditions	Recovery Method	Indicator
Halt Error	The data output is inhibited.	HMAC-SHA2-256 integrity test failure	The module must be re-loaded	Module will not load
	The data output is inhibited.	Failure of any of the CASTs	The module must be re-loaded	Error message related to the crypto function listed in Table 19 and the flag 'fips_selftest_fail' is set.

State Name	Description	Conditions	Recovery Method	Indicator
	The data output is inhibited.	Failure of any of the PCTs	The module must be re-loaded	Error message a PCT failure for RSA, DH, ECDH or ECDSA pairwise consistency test and the flag 'fips_selftest_fail' is set.

Table 20 - Error States

The module must be re-loaded in order to clear the error condition.

10.5 Operator Initiation of Self-Tests

The on demand self-tests is performed by powering-off and reloading the module. During the execution of the on demand self-tests, Conditional Cryptographic Algorithm Self-Tests (CASTs) and integrity test are performed. The crypto services are not available and no data output or input is possible.

11 Life-cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

Startup Procedures: Before the Crypto Officer can configure and use the BIG-IP system, the Crypto Officer must activate a valid license on the system. The Crypto Officer chooses the license to buy by selecting the hypervisor from Table 2, and the throughput needed. The following procedures described in "K77552 Licensing the BIG-IP system" on my.F5.com (https://my.f5.com/manage/s/article/K7752#reg) are performed:

- Obtaining a registration key
- Obtaining a dossier
- Activating the license

Installation Process: The Crypto Officer downloads the BIG-IP VE image (ie the module i.e. 1.0.2za-fips binary and its integrity check file) and deploys it. After the FIPS validated module license is installed, the command prompt will change to ‘REBOOT REQUIRED’. The Crypto Officer must reboot the BIG-IP for all FIPS-compliant changes to take effect.

11.2 Administrator Guidance

The Crypto Officer should verify the validity of the BIG-IP software license by running the command: *‘tmsh show sys license*' which should output 'FIPS 140, BIG-IP VE-1G to 10G,’ under the ‘Active Modules’ list.

On the BIG-IP product the Crypto Officer should call the dedicated Show version API, fips_get_f5fips_module_version, to ensure that the module identifier and version are shown as: Cryptographic Module for BIG-IP OpenSSL 1.0.2za-fips.

If the module has passed all self-tests then it is operating in the Approved mode. The Approved mode of operation can only transition into the non-Approved mode by calling one of the non-Approved services listed in Table - Non-Approved Services.

The ESV Public Use Document (PUD) reference for non-physical entropy source is as follows: https://csrc.nist.gov/projects/cryptographic-module-validation-program/entropy-validations/certificate/74

11.3 Design and Rules

The Crypto Officer shall consider the following requirements and restrictions when using the module.

- AES GCM IV see Section 2.7
- SP800-56Ar3 assurances see Section 2.7
- RSA modulus size see Section 2.7

12 Mitigation of Other Attacks

The module does not implement security mechanisms to mitigate other attacks.

Appendix A. Glossary and Abbreviations

AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
CAST	Cryptographic Algorithm Self-Test
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cipher Feedback
CKG	Cryptographic Key Generation
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CPU	Central Processing Unit
CSP	Critical Security Parameter
CTR	Counter Mode
DES	Data Encryption Standard
DSA	Digital Signature Algorithm
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ESV	Entropy Source Validation
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standards Publication
GCM	Galois Counter Mode
GMAC	Galois Message Authentication Code
HMAC	Hash Message Authentication Code
IV	Initialization Vector
KAS	Key Agreement Schema
KAT	Known Answer Test
KDF	Key Derivation Function
KTS	Key Transport Scheme
KW	AES Key Wrap
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
OFB	Output Feedback
PAA	Processor Algorithm Acceleration
PCT	Pairwise Consistency Test
PSP	Public Security Parameter
PSS	Probabilistic Signature Scheme
RAM	Random-Access Memory
RNG	Random Number Generator
RSA	Rivest, Shamir, Adleman
SHA	Secure Hash Algorithm
SSH	Secure Shell
TDES	Triple-DES
TLS	Transport Layer Security
XTS	XEX-based Tweaked-codebook mode with cipher text Stealing

Appendix B. References

FIPS140-3	**FIPS PUB 140-3 - Security Requirements For Cryptographic Modules** March 2019 https://doi.org/10.6028/NIST.FIPS.140-3
FIPS140-3_IG	**Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program** https://csrc.nist.gov/Projects/cryptographic-module-validation-program/fips-140-3-ig-announcements
FIPS180-4	**Secure Hash Standard (SHS)** August 2015 http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
FIPS186-4	**Digital Signature Standard (DSS)** July 2013 http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
FIPS197	**Advanced Encryption Standard** November 2001 http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
FIPS198-1	**The Keyed Hash Message Authentication Code (HMAC)** July 2008 http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf
FIPS202	**SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions** August 2015 http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf
PKCS#1	**Public Key Cryptography Standards (PKCS) #1: RSA Cryptography** Specifications Version 2.1 February 2003 http://www.ietf.org/rfc/rfc3447.txt
RFC3394	**Advanced Encryption Standard (AES) Key Wrap Algorithm** September 2002 http://www.ietf.org/rfc/rfc3394.txt
SP800-38A	**NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques** December 2001 http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf
SP800-38B	**NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication** May 2005 http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf
SP800-38C	**NIST Special Publication 800-38C - Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality** May 2004 http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf
SP800-38D	**NIST Special Publication 800-38D - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC** November 2007 http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf

SP800-38E	**NIST Special Publication 800-38E - Recommendation for Block Cipher Modes of Operation: The XTS AES Mode for Confidentiality on Storage Devices** January 2010 http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf
SP800-38F	**NIST Special Publication 800-38F - Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping** December 2012 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf
SP800-38G	**NIST Special Publication 800-38G - Recommendation for Block Cipher Modes of Operation: Methods for Format - Preserving Encryption** March 2016 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38G.pdf
SP800-52r2	**Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations** August 2019 https://doi.org/10.6028/NIST.SP.800-52r2
SP800-56Ar3	**NIST Special Publication 800-56A Revision 3 - Recommendation for Pair Wise Key Establishment Schemes Using Discrete Logarithm Cryptography** April 2018 https://doi.org/10.6028/NIST.SP.800-56Ar3
SP800-56Cr2	**Recommendation for Key Derivation through Extraction-then-Expansion** August 2020 https://doi.org/10.6028/NIST.SP.800-56Cr2
SP800-57	**NIST Special Publication 800-57 Part 1 Revision 4 - Recommendation for Key Management Part 1: General** January 2016 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf
SP800-90Ar1	**NIST Special Publication 800-90A - Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 http://dx.doi.org/10.6028/NIST.SP.800-90Ar1
SP800-90B	**(Second DRAFT) NIST Special Publication 800-90B - Recommendation for the Entropy Sources Used for Random Bit Generation** January 2018 https://doi.org/10.6028/NIST.SP.800-90B
SP800-131Ar2	**NIST Special Publication 800-131A Revision 2- Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths** March 2019 https://doi.org/10.6028/NIST.SP.800-131Ar2
SP800-132	**NIST Special Publication 800-132 - Recommendation for Password-Based Key Derivation - Part 1: Storage Applications** December 2010 http://csrc.nist.gov/publications/nistpubs/800-132/nist-sp800-132.pdf
SP800-133r2	**NIST Special Publication 800-133 - Recommendation for Cryptographic Key Generation** June 2020 https://doi.org/10.6028/NIST.SP.800-133r2
SP800-135r1	**NIST Special Publication 800-135 Revision 1 - Recommendation for Existing Application-Specific Key Derivation Functions** December 2011 http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-135r1.pdf

SP800-140B **NIST Special Publication 800-140B - CMVP Security Policy Requirements**
March 2020
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-140B.pdf