F5, Inc.

OpenSSL Cryptographic Module

FIPS 140-3 Non-Proprietary Security Policy

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

Table of Contents

List of Tables

List of Figures

Copyrights and Trademarks

1 General

1.1 Overview

This document is the non-proprietary FIPS 140-3 Security Policy that contains the security rules under which the OpenSSL Cryptographic Module 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

Section	Title	Security Level
1	General	1
2	Cryptographic module specification	1
3	Cryptographic module interfaces	1
4	Roles, services, and authentication	1
5	Software/Firmware security	1
6	Operational environment	1
7	Physical security	N/A
8	Non-invasive security	N/A
9	Sensitive security parameter management	1
10	Self-tests	1
11	Life-cycle assurance	1
12	Mitigation of other attacks	N/A
	Overall Level	1

Table 1: Security Levels

2 Cryptographic Module Specification

2.1 Description

Purpose and Use: The OpenSSL Cryptographic Module (hereafter referred to as "the module") is a cryptographic library offering various cryptographic mechanisms to be used by OpenSSL application running on F5 VELOS system controller and blade. 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: MultiChipStand

Cryptographic Boundary: The software block diagram Figure 1 shows the module, its interfaces with the operational environment and the delimitation of its cryptographic boundary with bold black perimeter. The software components of the cryptographic module are listed in Table - Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets).

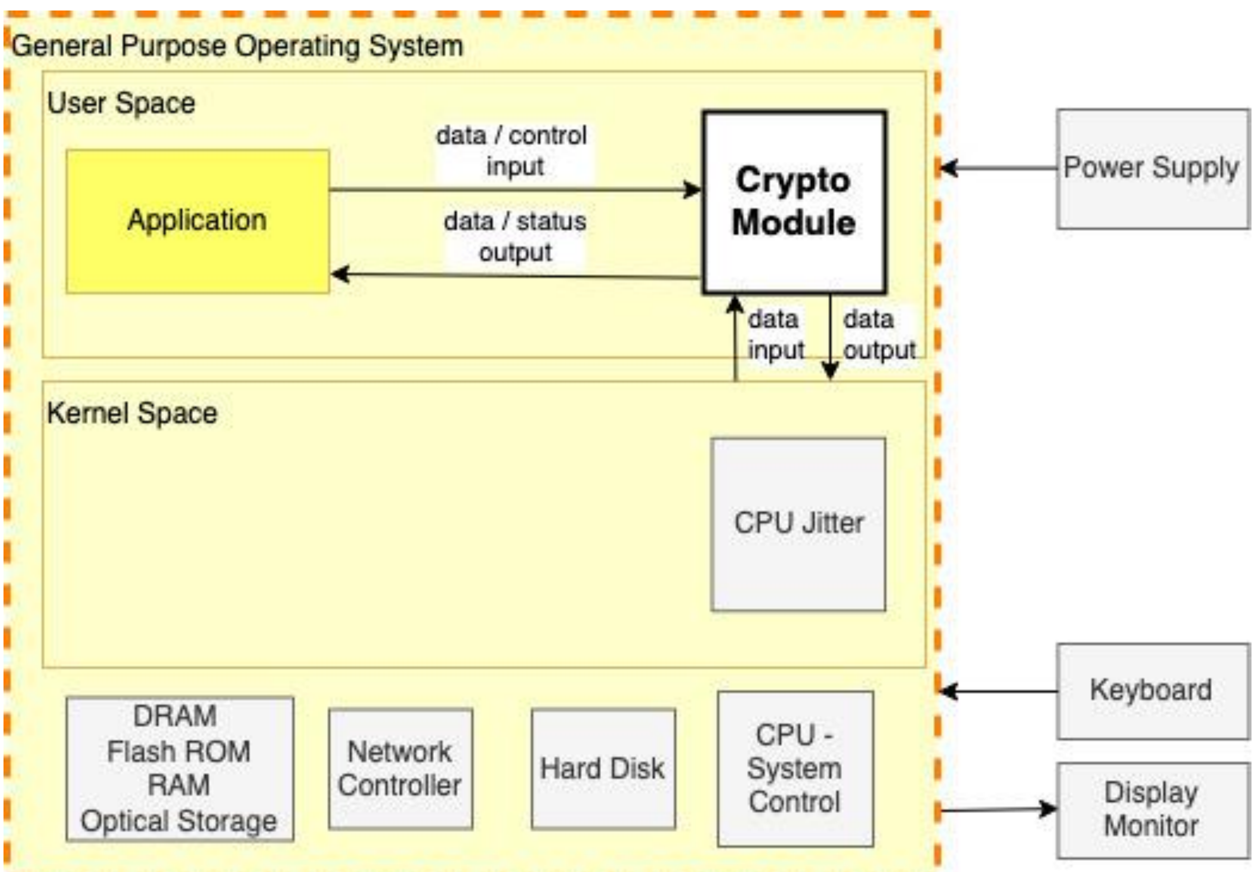

Figure 1: Block Diagram

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 1. The components of the TOEPP are listed in Table - Tested Operational Environments - Software, Firmware, Hybrid.

The entropy source located within the module's physical perimeter is outside of the module's cryptographic boundary (see Figure 1).

2.2 Tested and Vendor Affirmed Module Version and Identification

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

Package or File Name	Software/ Firmware Version	Features	Integrity Test
libcrypto.so.1.0.2zc and .libcrypto.so.1.0.2zc.hmac	1.0.2zc-fips	N/A	HMAC-SHA2-256

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

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
F5OS-C 1.6.0	VELOS Controller CX410	Intel Atom C3758 Denverton-NS	Yes	N/A	1.0.2zc-fips
F5OS-C 1.6.0	VELOS Controller CX410	Intel Atom C3758 Denverton-NS	No	N/A	1.0.2zc-fips
F5OS-C 1.6.0	VELOS Blade BX110	Intel Xeon D-2177NT Skylake-D	Yes	N/A	1.0.2zc-fips
F5OS-C 1.6.0	VELOS Blade BX110	Intel Xeon D-2177NT Skylake-D	No	N/A	1.0.2zc-fips

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

2.3 Excluded Components

None

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved mode	Only approved security functions or vendor affirmed	Approved	The status output from the FIPS_set_indicator_status service call is provided. To read this indicator, the calling application must register a callback function using `FIPS_register_indicator_callback'. The

Mode Name	Description	Type	Status Indicator
	security functions can be used.		callback function should take the input of the form "char *" which is the form of the indicator being output by the module.
Non-Approved mode	Only non-approved security functions can be used	Non-Approved	No service indicator

Table 4: Modes List and Description

Mode Change Instructions and Status:

The module enters the approved mode after pre-operational self-tests succeed. The module automatically switches between the approved and non-approved modes depending on the services requested by the operator. The status indicator of the mode of operation is equivalent to the indicator of the service that was requested.

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A4782	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CBC	A4783	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CTR	A4782	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A4782	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A4783	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A4782	Direction - Decrypt, Encrypt IV Generation - Internal IV Generation Mode - 8.2.1 Key Length - 128, 192, 256	SP 800-38D
AES-GCM	A4783	Direction - Decrypt, Encrypt IV Generation - Internal IV Generation Mode - 8.2.1 Key Length - 128, 192, 256	SP 800-38D
AES-GMAC	A4782	Direction - Decrypt, Encrypt IV Generation - Internal	SP 800-38D

Algorithm	CAVP Cert	Properties	Reference
		IV Generation Mode - 8.2.1 Key Length - 128, 192, 256	
AES-GMAC	A4783	Direction - Decrypt, Encrypt IV Generation - Internal IV Generation Mode - 8.2.1 Key Length - 128, 192, 256	SP 800-38D
Counter DRBG	A4782	Prediction Resistance - No, Yes Mode - AES-256 Derivation Function Enabled - No, Yes	SP 800-90A Rev. 1
Counter DRBG	A4783	Prediction Resistance - No Mode - AES-256 Derivation Function Enabled - Yes	SP 800-90A Rev. 1
ECDSA KeyGen (FIPS186-4)	A4782	Curve - P-256, P-384	FIPS 186-4
ECDSA KeyVer (FIPS186-4)	A4782	Curve - P-256, P-384	FIPS 186-4
ECDSA SigGen (FIPS186-4)	A4782	Component - No Curve - P-256, P-384	FIPS 186-4
ECDSA SigVer (FIPS186-4)	A4782	Component - No Curve - P-256, P-384	FIPS 186-4
HMAC-SHA-1	A4782	Key Length - Key Length: 8, 16, 64, 128, 1024	FIPS 198-1
HMAC-SHA-1	A4783	Key Length - Key Length: 8, 16, 64, 128, 1024	FIPS 198-1
HMAC-SHA2-256	A4782	Key Length - Key Length: 8, 16, 64, 128, 1024	FIPS 198-1
HMAC-SHA2-384	A4782	Key Length - Key Length: 8, 16, 64, 128, 1024	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A4782	Domain Parameter Generation Methods - P-256, P-384 Scheme - staticUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3
KDF SSH (CVL)	A4782	Cipher - AES-128, AES-256	SP 800-135 Rev. 1
KDF TLS (CVL)	A4782	TLS Version - v1.0/1.1	SP 800-135 Rev. 1
RSA KeyGen (FIPS186-4)	A4782	Key Generation Mode - B.3.3 Modulo - 2048, 3072, 4096 Primality Tests - Table C.2 Private Key Format - Standard	FIPS 186-4

Algorithm	CAVP Cert	Properties	Reference
RSA SigGen (FIPS186-4)	A4782	Signature Type - PKCS 1.5 Modulo - 2048, 3072, 4096	FIPS 186-4
RSA SigVer (FIPS186-4)	A4782	Signature Type - PKCS 1.5 Modulo - 2048, 3072, 4096	FIPS 186-4
SHA-1	A4782	-	FIPS 180-4
SHA-1	A4783	-	FIPS 180-4
SHA2-256	A4782	-	FIPS 180-4
SHA2-384	A4782	-	FIPS 180-4
TLS v1.2 KDF RFC7627 (CVL)	A4782	-	SP 800-135 Rev. 1

Table 5: Approved Algorithms

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.

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
Cryptographic Key Generation (CKG)	Key Type:Asymmetric	N/A	Random bit strings required for generating the cryptographic keys is compliant with section 4 example 1 of SP800-133r2

Table 6: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

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

Non-Approved, Allowed Algorithms with No Security Claimed:

Name	Caveat	Use and 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:

Name	Use and Function
AES with OFB, CCM, CFB, XTS, KW modes	Symmetric encryption and decryption
Blowfish, Camellia, CAST5, DES, IDEA, RC2, RC4, SEED, SM2, SM4, Triple-DES	Symmetric encryption and decryption
SHA2-224, SHA2-512, SM3, MD4, MD5 (outside of TLS), MDC2, RIPEMD, Whirlpool	Message digest
HMAC-SHA2-224, HMAC-SHA2-512, AES CMAC, Triple-DES CMAC	Message authentication
PKCS #1 v1.5 scheme with 1024 and greater than 4096 up to 16384 modulus, for all SHA sizes	RSA signature generation and verification
Probabilistic Signature Scheme (PSS), ANSI X9.31 schemes	RSA signature generation and verification
PKCS #1 v1.5 scheme with modulus size 2048, 3072, 4096 bits with SHA-1, SHA2-224, SHA2-512	RSA signature generation
PKCS #1 v1.5 scheme with modulus size 2048, 3072, 4096 bits with SHA2-224, SHA2-512	RSA signature verification
ECDSA with P-224, P-521	ECDSA key generation / verification
ECDSA with curves P-256, P-384 with SHA-1 SHA2-224, SHA2-512	ECDSA signature generation / verification
ECDSA using SM2	Digital signature generation and verification
RSA with modulus sizes up to 16384 bits	RSA encrypt / decrypt
DSA with all key and SHA sizes	DSA domain parameter generation, domain parameter verification, key pair generation, signature generation and verification
HMAC_DRBG and Hash_DRBG for all SHA sizes	Random number generation
CTR_DRBG with AES-128, AES-192	Random number generation
ANSI X9.31 RNG	Random number generation
Diffie-Hellman	Shared secret computation
EC Diffie-Hellman Ephemeral Unified with curves other than P-256, P-384, without KDF. EC Diffie-Hellman without KDF or using onePassDH / StaticUnified schemes	Shared secret computation
Key Derivation function in the context of TLS using SHA2-224 / SHA2-512	TLS KDF
Key Derivation function in the context of SSH using SHA-1 / SHA2-224 / SHA2-512	SSH KDF

Name	Use and Function
PKCS #1 v1.5 with keys other than 2048 / 3072 / 4096-bit using SHA2-256, SHA2-384	RSA signature generation and verification

Table 8: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
EC Diffie-Hellman Shared Secret Computation	KAS-SSC	[SP800-56ARev3] Shared Secret Computation used in Key Agreement Scheme (KAS) IG D.F scenario 2 (path 1)	Curves:P-256, P-384 with strength 128 and 192-bits	KAS-ECC-SSC Sp800-56Ar3: (A4782)
AES-Key Wrapping	KTS-Wrap	FIPS [197, SP800-38F],IG D.G. key wrapping and unwrapping, in the context of the TLS protocol, are provided by the TLS record layer using an approved authenticated encryption mode.	Keys:128 / 256-bit AES key with security strength from 128 and 256-bits	AES-GCM: (A4783, A4782)
Encryption with AES	BC-UnAuth	Encryption using AES	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CBC: (A4783, A4782) AES-ECB: (A4783, A4782) AES-CTR: (A4782)
Decryption with AES	BC-UnAuth	Decryption using AES	Keys:128, 192, 256 bits with 128-256 bits of key strength	AES-CBC: (A4783, A4782) AES-ECB: (A4783, A4782) AES-CTR: (A4782)
ECC key pair generation	AsymKeyPair-KeyGen	ECDSA / ECDH key pair generation	Curves:P-256 and P-384 curves with security strength 128 and 192-bits	ECDSA KeyGen (FIPS186-4): (A4782)
ECC public key verification	AsymKeyPair-KeyVer	[FIPS 186-4] key verification using ECDSA and EC Diffie-Hellman keys	Curves:P-256 and P-384 with strength 128 and 192-bits	ECDSA KeyVer (FIPS186-4): (A4782)

Name	Type	Description	Properties	Algorithms
ECDSA signature generation	DigSig-SigGen	[FIPS 186-4] Digital signature generation using ECDSA	Curves:P-256, P- 384 with security strength 128 and 192-bits Hashes:SHA2-256, SHA2-384	ECDSA SigGen (FIPS186-4): (A4782) ECDSA / ECDH key pair : P-256, P-384
ECDSA signature verification	DigSig-SigVer	[FIPS 186-4] Signature verification using ECDSA	Curves:P-256 and P-384 with security strength 128 and 192-bits Hashes:SHA2-256, SHA2-384	ECDSA SigVer (FIPS186-4): (A4782)
Message digest	SHA	[FIPS180-4] Message digest using SHA	N/A:N/A	SHA-1: (A4783, A4782) SHA2-256: (A4782) SHA2-384: (A4782)
Message authentication generation with HMAC	MAC	Message authentication generation using HMAC	SHA algorithm:SHA-1, SHA2-256, SHA2-384,	HMAC-SHA-1: (A4783, A4782) HMAC-SHA2-256: (A4782) HMAC-SHA2-384: (A4782)
Message authentication verification with HMAC	MAC	Message authentication verification using HMAC	SHA algorithm:SHA-1, SHA2-256, SHA2-384,	HMAC-SHA-1: (A4783, A4782) HMAC-SHA2-256: (A4782) HMAC-SHA2-384: (A4782)
Key derivation	KAS-135KDF	Key derivation using protocol KDF	Derived keys:112 to 256 bits	KDF SSH: (A4782) KDF TLS: (A4782) TLS v1.2 KDF RFC7627: (A4782)
RSA key generation	AsymKeyPair-KeyGen	[FIPS 186-4] B.3.3 Probable primes with standard key format	Keys:2048 / 3072 / 4096-bit with security strength from 112 to 150-bits	RSA KeyGen (FIPS186-4): (A4782)
Message authentication generation with AES	MAC	Message authentication generation using AES	Keys:128 /192 / 256 bits with security strength from 128 to 256 bits	AES-GMAC: (A4783, A4782)

Name	Type	Description	Properties	Algorithms
Message authentication verification with AES	MAC	Message authentication verification using AES	Keys:128 /192 / 256 bits with security strength from 128 to 256 bits	AES-GMAC: (A4783, A4782)
Authenticated encryption with AES GCM	BC-Auth	Authenticated encryption using AES	Keys:128 or 256 bits with 128 or 256 bits of strength Authenticated Encryption: Internal IV Mode 8.2.1	AES-GCM: (A4783, A4782)
Authenticated decryption with AES GCM	BC-Auth	Authenticated decryption using AES	Keys:128 or 256 bits with 128 or 256 bits of strength. Authenticated Decryption: External IV	AES-GCM: (A4783, A4782)
Random Number Generation	DRBG	Random number generation using DRBG with AES-236 in CTR mode	Seed, V and key values :Security strength 256-bits	Counter DRBG: (A4783, A4782)
RSA signature generation	DigSig-SigGen	PKCS 1.5 digital signature generation using RSA with SHA-256, SHA-384	Keys:2048 / 3072 / 4096-bit with security strength from 112 to 150-bits Hashes:SHA2-256, SHA2-384	RSA SigGen (FIPS186-4): (A4782)
RSA signature verification	DigSig-SigVer	PKCS 1.5 digital signature verification using RSA with SHA-256, SHA-384	Keys:2048 / 3072 / 4096-bit with security strength from 112 to 150-bits Hashes:SHA2-256, SHA2-384	RSA SigVer (FIPS186-4): (A4782)
RSA signature verification (legacy)	DigSig-SigVer	PKCS 1.5 digital signature verification using RSA with SHA-1	Publications:FIPS 140-3 IG C.M legacy algorithms Keys:2048 / 3072 / 4096-bit with security strength from 112 to 150-bits Hashes: SHA-1	RSA SigVer (FIPS186-4): (A4782)

Table 9: Security Function Implementations

2.7 Algorithm Specific Information

AES GCM Use:

The IV for AES-GCM is constructed in compliance with IG C.H scenario 1a (TLS 1.2) and scenario 1d (SSHv2).

- For TLS 1.2, the module offers the AES-GCM implementation and uses the context of Scenario 1 of IG C.H. The module is compliant with SP800-52Rev2 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.
- For SSHv2, the IV for the module AES-GCM implementation is only used in the context of the AES-GCM mode encryptions. The module is compliant with RFCs 4252, 4253 and 5647.
  The module does not implement SSH protocol and 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 SSH protocol implicitly ensures that the counter does not exhaust the maximum number of possible values for a given session key and that the no more than 2^{64} -1 ASE-GCM encryptions are performed.
  When a session is terminated, as new key and a new initial IV shall be derived.
- For both TLSv1.2 and SSHv2 protocols, 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.

SHA-1 Use:

SHA-1 from Message Digest is only approved for non-digital-signature uses (see Table 8 of SP 800-131A rev2).

Legacy Use

- RSA Digital signature verification using SHA-1 is allowed for legacy use only.
- Algorithms 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.

2.8 RBG and Entropy

Cert Number	Vendor Name
E85	F5

Table 10: Entropy Certificates

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
CPU Jitter 3.4.1	Non-Physical	OEs listed in Table 3	256 bits	256 bits	SHA-3 vetted conditioning component. ACVP Cert. A4093

Table 11: Entropy Sources

The module entropy source specified in Table Entropy Sources uses jitter variations caused by executing instructions and memory accessed. The operator does not have the ability to modify the F5 entropy source (ES) configuration settings (see details in Public Use Document referenced in section 11.2).

The module employs a Deterministic Random Bit Generator (DRBG) based on [SP800-90ARev1] 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 CTR_DRBG with AES-256.

The output of entropy sources provides 256-bits of entropy to seed and reseed SP800-90ARev1 DRBG during initialization (seed) and reseeding (reseed).

2.9 Key Generation

The module implements asymmetric key generation methods according to SP 800-133r2 section 5. The key generation methods are specified in the *Security Function Implementations* table.

The module does not implement symmetric key generation.

2.10 Key Establishment

The module implements SSP agreement, compliant with IG D.F scenario 2 (path 1). Additionally, the module implements SSP transport, compliant with IG D.G. The Key Establishment methods are specified in the *Security Function Implementations* table.

2.11 Industry Protocols

GCM with internal IV generation in the approved mode is compliant with version 1.2 of the TLS protocol (RFC 5288) and shall only be used in conjunction with the TLS protocol.

Additionally, the module implements the TLS 1.2 and SSH key derivation functions for use in the TLS protocol and SSH protocol (RFC 4253 and RFC 6668) respectively. The strength of the derived session key is based on the shared secret and the SHA function used as follows:

For deriving session key with 192 bit strength, the TLS/SSH key derivation functions with shared secret based on P-384 curve using SHA-384 should be used.

For deriving session key with 128 bit strength,

- TLS key derivation functions with shared secret based on P-256 curve using SHA-256, SHA-384 or P-384 curve using SHA-256 should be used.
- SSH key derivation functions with shared secret based on P-256 curve using SHA-1, SHA-256, SHA-384 or P-384 curve using SHA-1, SHA-256 should be used.

The TLS v1.0 / 1.1 / 1.2 and SSHv2 protocols have not been reviewed or tested by the CAVP or CMVP.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	Data inputs are provided in the variables passed in the API and callable service invocations, generally through caller-supplied buffers
N/A	Data Output	Data outputs are provided in the variables passed in the API and callable service invocations, generally through caller-supplied buffers
N/A	Control Input	Control inputs which control the mode of the module are provided through dedicated parameters.
N/A	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 12: Ports and Interfaces

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

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

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None

Table 13: Roles

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Encryption	Executes AES-mode encrypt operation	AES-ECB, AES-CBC, AES-CTR	Plaintext and key	Ciphertext	Encryption with AES	Crypto Officer - AES key : W,E
Decryption	Executes AES-mode decrypt operation	AES-ECB, AES-CBC, AES-CTR	Ciphertext and key	Plaintext	Decryption with AES	Crypto Officer - AES key : W,E
Key wrapping	Executes AES-key wrapping or unwrapping operation	AES-GCM encrypt / decrypt	Key wrapping key and key to be wrapped	Wrapped key	AES-Key Wrapping	Crypto Officer - AES key : W,E - GCM IV in TLS context: G,W,E - GCM IV in SSH context: G,W,E
Random number generation	Generate Random number	CTR-DRBG-AES-256	Number of bits	Random numbers	Random Number Generation	Crypto Officer - Entropy input string : W,E - DRBG seed : G - DRBG internal state (V and key values) : G
RSA key pair generation	Generate RSA key pair	RSA-KEY-GEN-2048, RSA-KEY-GEN-3072,	Key size	Key pair	RSA key generation	Crypto Officer - RSA private key: G,R

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		RSA-KEY-GEN-4096				- RSA public key: G,R
RSA signature generation	Sign a message with a specified RSA private key	RSA-SIG	Private key, Message, Hashing algorithm	Computed signature	RSA signature generation	Crypto Officer - RSA private key: W,E
Authenticated Encryption	Authenticated Encryption	AES-GCM	AES key, plaintext	Ciphertext	Authenticated encryption with AES GCM	Crypto Officer - AES key : W,E - GCM IV in TLS context: G,W,E - GCM IV in SSH context: G,W,E
Authenticated Decryption	Authenticated Decryption	AES-GCM	AES key, ciphertext	Plaintext	Authenticated decryption with AES GCM	Crypto Officer - AES key : W,E - GCM IV in TLS context: W,E - GCM IV in SSH context: W,E
Message Authentication Generation with AES	MAC computation	AES-GMAC	AES key, message	MAC tag	Message authentication generation with AES	Crypto Officer - AES key : W,E
Message Authentication Generation with HMAC	MAC computation	MSG-AUTH-HMAC-SHA-1, MSG-AUTH-HMAC-SHA-256 MSG-AUTH-HMAC-SHA-384	HMAC key, message	MAC tag	Message authentication generation with HMAC	Crypto Officer - HMAC key : W,E
Message Authentication Verification with AES	MAC computation	AES-GMAC	AES key, Authenticated message, MAC algorithm	Message	Message authentication verification with AES	Crypto Officer - AES key : W,E
Message Authentication	MAC computation	MSG-AUTH-HMAC-SHA-1, MSG-AUTH-HMAC-SHA-	HMAC key, Authenticated message,	Message	Message authentication	Crypto Officer - HMAC key : W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Verification with HMAC		256 MSG-AUTH-HMAC-SHA-384	MAC algorithm		verification with HMAC	
Message Digest	Generating message digest	MESSAGE-DIGEST-SHA-1 MESSAGE-DIGEST-SHA-256 MESSAGE-DIGEST-SHA-384	Message	Message digest	Message digest	Crypto Officer
ECDSA key pair generation	Generate ECDSA key pair	EC-KEYGEN-P-256, EC-KEYGEN-P-284	Curve	ECDSA key pair	ECC key pair generation	Crypto Officer - ECDSA private key: G,R - ECDSA public key: G,R - EC Diffie-Hellman private key: G,R - EC Diffie-Hellman public key: G,R
ECDSA key pair verification	Verify ECDSA key pair	EC-KEY-VERIFY-P-256, EC-KEY-VERIFY-P-384	Public key	Success/ error	ECC public key verification	Crypto Officer - ECDSA public key: W - EC Diffie-Hellman public key: W
RSA signature verification	Verify the signature of a message with a specified RSA public key.	RSA-VER	RSA public key, digital signature, message, Hashing algorithm	Pass / fail result of digital signature verification	RSA signature verification RSA signature verification (legacy)	Crypto Officer - RSA public key: W,E
ECDSA signature generation	Sign a message with a specified ECDSA private key.	ECDSA-SIGN-P-256, ECDSA-SIGN-P-384	ECDSA private key, Message, Hashing algorithm	Computed signature	ECDSA signature generation	Crypto Officer - ECDSA private key: W,E
ECDSA signature verification	Verify the signature of a message with a specified	ECDSA-VERIFY-P-256, ECDSA-VERIFY-P-384	ECDSA public key, digital signature, message,	Digital signature verification result	ECDSA signature verification	Crypto Officer - ECDSA public key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	ECDSA public key		Hashing algorithm			
EC Diffie-Hellman shared secret computation	Calculate a shared secret via the ECDH algorithm.	ECDH-COMPUTE-KEY-P-256, ECDH-COMPUTE-KEY-P-384	EC public key, EC private key	Shared Secret	EC Diffie-Hellman Shared Secret Computation	Crypto Officer - EC Diffie-Hellman private key: W - EC Diffie-Hellman shared secret: G,R
Key derivation using TLS pre-primary secret	Deriving TLS keys	TLS-P-HASH-DERIVATION-SHA-1 TLS-P-HASH-DERIVATION-SHA-256 TLS-P-HASH-DERIVATION SHA-384	TLS pre-primary secret	TLS primary secret	Key derivation	Crypto Officer - TLS pre-primary secret : W,E - TLS primary secret: G,R
Key derivation using TLS primary secret	Deriving TLS keys	TLS-P-HASH-DERIVATION-SHA-1 TLS-P-HASH-DERIVATION-SHA-256 TLS-P-HASH-DERIVATION SHA-384	TLS primary secret	TLS Derived Key	Key derivation	Crypto Officer - TLS primary secret: W,E - TLS derived session key : G,R
Key derivation using SSH shared secret	Deriving SSH keys	SSH-KEY-HASH-DERIVATION-SHA-256 SSH-KEY-HASH-DERIVATION SHA-384	Shared secret, Key length	SSH derived Key	Key derivation	Crypto Officer - SSH derived session key : G,R - SSH shared secret: W,E
Show version	Return the SW version and the module's name	N/A	N/A	Module name and version	None	Unauthenticated Crypto Officer
Show Status	Return the module status	N/A	N/A	Module status	None	Unauthenticated Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Zeroization	Zeroize all non-protected SSPs	N/A	N/A	All SSPs in the SSPs table	None	Crypto Officer - AES key : Z - HMAC key : Z - RSA private key: Z - RSA public key: Z - ECDSA private key: Z - ECDSA public key: Z - EC Diffie-Hellman private key: Z - EC Diffie-Hellman public key: Z - EC Diffie-Hellman shared secret: Z - TLS pre-primary secret : Z - TLS primary secret: Z - TLS derived session key : Z - SSH shared secret: Z - SSH derived session key : Z - Entropy input string : Z - DRBG seed : Z - DRBG internal state (V and key values) : Z
Self-tests	Execute integrity test. Execute the CASTs	Integrity test, CASTs from sections 10.1 and 10.2	N/A	Pass or fail	EC Diffie-Hellman Shared Secret Computation AES-Key Wrapping Encryption with AES Decryption with AES	Unauthenticated Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
					ECC key pair generation ECC public key verification ECDSA signature generation ECDSA signature verification Message digest Message authentication generation with HMAC Message authentication verification with HMAC Key derivation RSA key generation Message authentication generation with AES Message authentication verification with AES Authenticated encryption with AES GCM Authenticated decryption with AES GCM Random Number Generation RSA signature generation RSA signature verification	

Table 14: Approved Services

For the above table, the convention below applies when specifying the access permissions (types) that the service has for each SSP.

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

Name	Description	Algorithms	Role
Symmetric encryption and decryption	Encryption / decryption	AES with OFB, CCM, CFB, XTS, KW modes Blowfish, Camellia, CAST5, DES, IDEA, RC2, RC4, SEED, SM2, SM4, Triple-DES	Crypto Officer
Message digest	Generating message digest	SHA2-224, SHA2-512, SM3, MD4, MD5 (outside of TLS), MDC2, RIPEMD, Whirlpool	Crypto Officer
Message authentication code generation and verification	MAC computation	HMAC-SHA2-224, HMAC-SHA2-512, AES CMAC, Triple-DES CMAC	Crypto Officer
RSA key generation	Generating key pair	PKCS #1 v1.5 scheme with 1024 and greater than 4096 up to 16384 modulus, for all SHA sizes	Crypto Officer
RSA signature generation and verification	Generating signature, verifying signature	Probabilistic Signature Scheme (PSS), ANSI X9.31 schemes PKCS #1 v1.5 scheme with modulus size 2048, 3072, 4096 bits with SHA-1, SHA2-224, SHA2-512 PKCS #1 v1.5 scheme with modulus size 2048, 3072, 4096 bits with SHA2-224, SHA2-512 PKCS #1 v1.5 with keys other than 2048 / 3072 / 4096-bit using SHA2-256, SHA2-384	Crypto Officer
Key generation / verification	Generating key pair	ECDSA with P-224, P-521	Crypto Officer
ECDSA signature generation & verification	Generating signature, verifying signature	ECDSA with P-224, P-521 ECDSA with curves P-256, P-384 with SHA-1 SHA2-224, SHA2-512 ECDSA using SM2	Crypto Officer
RSA encrypt / decrypt	Encryption / decryption	RSA with modulus sizes up to 16384 bits	Crypto Officer

Name	Description	Algorithms	Role
DSA domain parameter generation, domain parameter verification, key pair generation, signature generation and verification	Generating key pair, generating signature, verifying signature	DSA with all key and SHA sizes	Crypto Officer
Random number generation	Generating deterministic random number	HMAC_DRBG and Hash_DRBG for all SHA sizes CTR_DRBG with AES-128, AES-192 ANSI X9.31 RNG	Crypto Officer
Diffie-Hellman shared secret computation	Calculate a shared secret via the DH algorithm.	Diffie-Hellman	Crypto Officer
ECDH shared secret computation	Calculating shared secret	EC Diffie-Hellman Ephemeral Unified with curves other than P-256, P-384, without KDF. EC Diffie-Hellman without KDF or using onePassDH / StaticUnified schemes	Crypto Officer
Key derivation	Deriving TLS keys and SSH keys	Key Derivation function in the context of TLS using SHA2-224 / SHA2-512 Key Derivation function in the context of SSH using SHA-1 / SHA2-224 / SHA2-512	Crypto Officer

Table 15: Non-Approved Services

4.5 External Software/Firmware Loaded

None

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.1.0.2zc file, with the HMAC-SHA2-256 value stored in the module file .libcrypto.so.1.0.2zc.hmac that was computed at build time. The HMAC key used for integrity verification is 256 bits in length and is stored as part of the module binary.

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

5.2 Initiate on Demand

The on-demand integrity test is performed as part of the Pre-Operational Self-Tests by reloading the module.

6 Operational Environment

6.1 Operational Environment Type and Requirements

F5OS-C consists of a Linux based operating system customized for performance that runs directly on the hardware.

Type of Operational Environment: Modifiable

How Requirements are Satisfied:

The module shall be installed as stated in Section 11.1. If properly installed, the operating system provides process isolation and memory protection mechanisms that ensure appropriate separation for memory access among the processes on the system. Each process has control over its own data, and uncontrolled access to the data of other processes is prevented.

6.2 Configuration Settings and Restrictions

The module runs on a F5OS-C 1.6.0 operating system executing on the hardware and hypervisor specified in Table OEs. The module should be installed as stated in section 11. The operator should confirm that the module is installed correctly by 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 SP 800-140F 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 Area Name	Description	Persistence Type
RAM	The memory occupied by SSPs is allocated by regular memory allocation operating system calls.	Dynamic

Table 16: Storage Areas

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
API output parameters	CM Software	App via TOEPP Path	Plaintext	Manual	Electronic	
API input parameters	App via TOEPP Path	CM Software	Plaintext	Manual	Electronic	

Table 17: SSP Input-Output Methods

The module does not support manual SSP entry or intermediate key generation output. The SSPs are provided to the module in plaintext form via input 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 in the table above.

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Free Cipher Handle	Zeroizes the SSPs contained within the cipher handle	The destruction functions overwrite the memory occupied by keys with "zeros" and deallocate the memory with the regular memory deallocation operating system call.	The application is responsible for calling the appropriate destruction functions provided in the module's API: EVP_CIPHER_CTX_cleanup, HMAC_CTX_cleanup(), FIPS_rsa_free(), EC_KEY_free(), EC_POINT_free(), OPENSSL_cleanse, OPENSSL_free, FIPS_drbg_uninstantiate
Module Reset	De-allocates the volatile memory used to store SSPs	Volatile memory used by the module is overwritten within nanoseconds when power is removed.	By unloading and reloading the module.

Table 18: SSP Zeroization Methods

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES key	AES key used for encryption, decryption, and computing MAC tags	Key length: 128 to 256-bits - 128 to 256-bits	Symmetric - CSP			AES-Key Wrapping Encryption with AES Decryption with AES Message authentication generation with AES Message authentication verification with AES Authenticated encryption with AES GCM Authenticated decryption with AES GCM
HMAC key	HMAC key for Message Authentication Generation and Verification	Key length: 112 to 192-bits - 112 to 192-bits	Symmetric - CSP			Message authentication generation with HMAC Message authentication verification with HMAC
RSA private key	RSA private key used for RSA key generation, signature generation	Modulus N: 2048, 3072 and 4096-bits - 112 to 150-bits	Asymmetric - CSP	RSA key generation		RSA signature generation
RSA public key	RSA public key used for RSA key generation, signature verification	Modulus N: 2048, 3072 and 4096-bits - 112 to 150-bits	Asymmetric - PSP	RSA key generation		RSA signature verification
ECDSA private key	ECDSA private key used for EC key generation, key verification,	Curve size: P-256, P-384 - 128 and 192-bits	Asymmetric - CSP	ECC key pair generation		ECDSA signature generation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	signature generation, shared secret computation					
ECDSA public key	ECDSA public key used for EC key generation, key verification, signature verification, shared secret computation	Curve size: P-256, P-384 - 128 and 192-bits	Asymmetric - PSP	ECC key pair generation		ECC public key verification ECDSA signature verification
EC Diffie-Hellman private key	EC Diffie-Hellman private key used for EC key generation, key verification, signature generation, shared secret computation	Curve size: P-256, P-384 - 128 and 192-bits	Asymmetric - CSP	ECC key pair generation		EC Diffie-Hellman Shared Secret Computation ECC public key verification
EC Diffie-Hellman public key	EC Diffie-Hellman public key used for EC key generation, key verification, signature generation, shared secret computation	Curve size: P-256, P-384 - 128 and 192-bits	Asymmetric - PSP	ECC key pair generation		EC Diffie-Hellman Shared Secret Computation ECC public key verification
EC Diffie-Hellman shared secret	EC Diffie-Hellman shared secret generated by KAS-ECC-SSC	Curve size: P-256, P-384 - 128 and 192-bits	Asymmetric - CSP		EC Diffie-Hellman Shared Secret Computation	EC Diffie-Hellman Shared Secret Computation
TLS pre-primary secret	TLS pre-primary secret used for deriving the TLS primary secret	ECDH Curve size:: P-256, P-384 - 128 or 192-bits	Asymmetric - CSP			Key derivation
TLS primary secret	TLS primary secret used for deriving the TLS derived key	384-bits - 128 or 192-bits	Symmetric - CSP	Key derivation		Key derivation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
TLS derived session key	TLS derived session key from TLS primary secret	Key length: 128 and 256-bits (AES); HMAC_SHA2-256, HMAC-SHA2-384 - 128 or 192-bits	Symmetric - CSP	Key derivation		Key derivation
SSH shared secret	SSH shared secret used for deriving the SSH key	Curve size: P-256, P-384 - 128 or 192-bits	Asymmetric - CSP			Key derivation
SSH derived session key	SSH derived session key	Key length: 128 and 256-bits (AES); HMAC_SHA1, HMAC-SHA2-256 - 128 or 192-bits	Symmetric - CSP	Key derivation		Key derivation
Entropy input string	Entropy input string used to seed the DRBG	256 bits - 256 bits	Random number generation - CSP			Random Number Generation
DRBG seed	DRBG seed derived from entropy input as defined in SP 800-90Ar1	256 bits - 256 bits	Random number generation - CSP	Random Number Generation		Random Number Generation
DRBG internal state (V and key values)	Internal state of CTR_DRBG	256 bits - 256 bits	Random number generation - CSP	Random Number Generation		Random Number Generation
GCM IV in TLS context	Internal IV generated for GCM to be used for TLS compliant with RFC5288	96 bits - 96 bits	IV - PSP	SP 800-38D section 8.2.1 Deterministic generation		AES-Key Wrapping Authenticated encryption with AES GCM Authenticated decryption with AES GCM
GCM IV in SSH context	Internal IV generated for GCM to be used	96 bits - 96 bits	IV - PSP	SP 800-38D section 8.2.1		AES-Key Wrapping Authenticated encryption

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	for SSH compliant with RFC5647			Deterministic generation		with AES GCM Authenticated decryption with AES GCM

Table 19: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES key	API output parameters API input parameters	RAM:Plaintext	From handle creation until freeing the cipher handle	Free Cipher Handle Module Reset	
HMAC key	API input parameters	RAM:Plaintext	From handle creation until freeing the cipher handle	Free Cipher Handle Module Reset	
RSA private key	API output parameters API input parameters	RAM:Plaintext	From handle creation until freeing the cipher handle	Free Cipher Handle Module Reset	RSA public key:Paired With
RSA public key	API output parameters API input parameters	RAM:Plaintext	From handle creation until freeing the cipher handle	Free Cipher Handle Module Reset	RSA private key:Paired With
ECDSA private key	API output parameters API input parameters	RAM:Plaintext	From handle creation until freeing the cipher handle	Free Cipher Handle Module Reset	ECDSA public key:Paired With
ECDSA public key	API output parameters API input parameters	RAM:Plaintext	From handle creation until freeing the cipher handle	Free Cipher Handle Module Reset	ECDSA private key:Paired With
EC Diffie-Hellman private key	API output parameters API input parameters	RAM:Plaintext	From handle creation until freeing the cipher handle	Free Cipher Handle Module Reset	EC Diffie-Hellman public key:Paired With
EC Diffie-Hellman public key	API output parameters API input parameters	RAM:Plaintext	From handle creation until freeing the cipher handle	Free Cipher Handle Module Reset	EC Diffie-Hellman private key:Paired With

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
EC Diffie-Hellman shared secret	API output parameters	RAM:Plaintext	From handle creation until freeing the cipher handle	Free Cipher Handle Module Reset	
TLS pre-primary secret	API input parameters	RAM:Plaintext	From handle creation until freeing the cipher handle	Free Cipher Handle Module Reset	TLS primary secret:Used With
TLS primary secret	API output parameters API input parameters	RAM:Plaintext	From handle creation until freeing the cipher handle	Free Cipher Handle Module Reset	TLS pre-primary secret :Used With
TLS derived session key	API output parameters	RAM:Plaintext	From handle creation until freeing the cipher handle	Free Cipher Handle Module Reset	TLS primary secret:Derived From
SSH shared secret	API input parameters	RAM:Plaintext	From handle creation until freeing the cipher handle	Free Cipher Handle Module Reset	SSH derived session key :Used With
SSH derived session key	API output parameters	RAM:Plaintext	From handle creation until freeing the cipher handle	Free Cipher Handle Module Reset	SSH shared secret:Derived From
Entropy input string		RAM:Plaintext	Storage duration during the usage of the CSP	Module Reset	DRBG seed :Used With
DRBG seed		RAM:Plaintext	Storage duration during the usage of the CSP	Free Cipher Handle Module Reset	DRBG internal state (V and key values) :Used With
DRBG internal state (V and key values)		RAM:Plaintext	Storage duration during the usage of the CSP	Free Cipher Handle Module Reset	DRBG seed :Used With
GCM IV in TLS context	API output parameters API input parameters	RAM:Plaintext	From handle creation until freeing the cipher handle	Free Cipher Handle	
GCM IV in SSH context	API output parameters API input parameters	RAM:Plaintext	From handle creation until freeing the cipher handle	Free Cipher Handle	

Table 20: SSP Table 2

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A4782)	HMAC key: 256-bits	Message Authentication	SW/FW 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 crypto boundary that was computed at build time

Table 21: Pre-Operational Self-Tests

Pre-operational self-tests are performed automatically when the module is loaded into memory.

While the module is executing the pre-operational self-tests, services are not available, and input and output are inhibited. The module does not return control to the calling application until the tests are completed. On successful completion of the pre-operational self-tests, the module enters operational mode and cryptographic services are available.

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
Counter DRBG (A4783)	AES-256 in CTR mode, with derivation function, prediction resistance disabled	KAT	CAST	Module becomes operational	SP800-90ARev1 section 11.3 health tests	Test run during pre-operational self-test
AES-CBC (A4783)	128-bit key	KAT	CAST	Module becomes operational	Encryption / decryption	Test run during pre-operational self-test
AES-GCM (A4783)	128-bit key	KAT	CAST	Module becomes operational	Encryption / decryption	Test run during pre-operational self-test
RSA SigGen (FIPS186-4) (A4782)	2048 bit key and SHA2-256	KAT	CAST	Module becomes operational	Signature generation	Test run during pre-operational self-test
RSA SigVer (FIPS186-4) (A4782)	2048 bit key and SHA2-256	KAT	CAST	Module becomes operational	Signature verification	Test run during pre-operational self-test
RSA KeyGen (FIPS186-4) (A4782)	4096 bit key and SHA2-256	PCT	PCT	Asymmetric algorithm is performed	Calculation and verification of a digital signature	Key generation

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
ECDSA SigGen (FIPS186-4) (A4782)	P-256 and SHA2-256	KAT	CAST	Module becomes operational	Signature generation	Test run during pre-operational self-test
ECDSA SigVer (FIPS186-4) (A4782)	P-256 and SHA2-256	KAT	CAST	Module becomes operational	Signature verification	Test run during pre-operational self-test
ECDSA KeyGen (FIPS186-4) (A4782)	P-256 and SHA2-256	PCT	PCT	Asymmetric algorithm is performed	Calculation and verification of a digital signature	Key generation
KAS-ECC-SSC Sp800-56Ar3 (A4782)	P-256	KAT	CAST	Module becomes operational	Shared secret computation	Test run during pre-operational self-test
HMAC-SHA-1 (A4782)	HMAC-SHA-1	KAT	CAST	Module becomes operational	MAC	Test run during pre-operational self-test
HMAC-SHA2-256 (A4782)	HMAC-SHA2-256	KAT	CAST	Module becomes operational	MAC	Test run during pre-operational self-test
TLS v1.2 KDF RFC7627 (A4782)	SHA-256	KAT	CAST	Module becomes operational	Key derivation used in the TLS protocol	Test run during pre-operational self-test
KDF TLS (A4782)	SHA-256	KAT	CAST	Module becomes operational	Key derivation used in the TLS protocol	Test run during pre-operational self-test
KDF SSH (A4782)	SHA-256	KAT	CAST	Module becomes operational	Key derivation used in the SSH protocol	Test run during pre-operational self-test
HMAC-SHA-1 (A4783)	HMAC-SHA-1	KAT	CAST	Module becomes operational	MAC	Test run during pre-operational self-test
HMAC-SHA2-384 (A4782)	HMAC-SHA-384	KAT	CAST	Module becomes operational	MAC	Test run during pre-operational self-test
AES-CBC (A4782)	128-bit key	KAT	CAST	Module becomes operational	Encryption / decryption	Test run during pre-operational self-test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-GCM (A4782)	128-bit key	KAT	CAST	Module becomes operational	Encryption / decryption	Test run during pre-operational self-test
Counter DRBG (A4782)	AES-256 in CTR mode, with / without derivation function, prediction resistance enabled and disabled	KAT	CAST	Module becomes operational	SP800-90ARev1 section 11.3 health tests	Test run during pre-operational self-test

Table 22: Conditional Self-Tests

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A4782)	Message Authentication	SW/FW Integrity	Determined by the operator	Module reload

Table 23: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
Counter DRBG (A4783)	KAT	CAST	On Demand	Manually
AES-CBC (A4783)	KAT	CAST	On Demand	Manually
AES-GCM (A4783)	KAT	CAST	On Demand	Manually
RSA SigGen (FIPS186-4) (A4782)	KAT	CAST	On Demand	Manually
RSA SigVer (FIPS186-4) (A4782)	KAT	CAST	On Demand	Manually
RSA KeyGen (FIPS186-4) (A4782)	PCT	PCT	On Demand	Manually
ECDSA SigGen (FIPS186-4) (A4782)	KAT	CAST	On Demand	Manually
ECDSA SigVer (FIPS186-4) (A4782)	KAT	CAST	On Demand	Manually
ECDSA KeyGen (FIPS186-4) (A4782)	PCT	PCT	On Demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
KAS-ECC-SSC Sp800-56Ar3 (A4782)	KAT	CAST	On Demand	Manually
HMAC-SHA-1 (A4782)	KAT	CAST	On Demand	Manually
HMAC-SHA2-256 (A4782)	KAT	CAST	On Demand	Manually
TLS v1.2 KDF RFC7627 (A4782)	KAT	CAST	On Demand	Manually
KDF TLS (A4782)	KAT	CAST	On Demand	Manually
KDF SSH (A4782)	KAT	CAST	On Demand	Manually
HMAC-SHA-1 (A4783)	KAT	CAST	On Demand	Manually
HMAC-SHA2-384 (A4782)	KAT	CAST	On Demand	Manually
AES-CBC (A4782)	KAT	CAST	On Demand	Manually
AES-GCM (A4782)	KAT	CAST	On Demand	Manually
Counter DRBG (A4782)	KAT	CAST	On Demand	Manually

Table 24: Conditional Periodic Information

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Halt Error	Module is no longer operational. The data output is inhibited.	HMAC-SHA2-256 KAT failure or HMAC-SHA2-256 integrity test failure Failure of any of the CASTs Failure of any of the PCTs	The module must be re-loaded	Module will not load, Error message related to the crypto function listed in Table 18 and the flag 'fips_selftest_fail' is set.Error message a PCT failure for RSA, ECDH or ECDSA pairwise consistency test and the flag 'fips_selftest_fail' is set.

Table 25: Error States

10.5 Operator Initiation of Self-Tests

The on demand self-tests can be invoked by unloading and subsequently reloading the module. This service performs the same cryptographic algorithm tests executed during pre-operational self-test and module loading. During the execution of the on demand self-tests, 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 F5OS-C software on VELOS platforms, the Crypto Officer must license the VELOS system.

For automatic VELOS system licensing, the system needs to be able to connect to the F5 licensing server either through the internet or another means of networking. You need to have the Base Registration Key (five sets of characters separated by hyphens) provided by F5, and any add-on keys (two sets of 7 characters separated by a hyphen) that you have purchased. The Base Registration Key with associated add-on keys are pre-installed on a new VELOS system. The activation of the VELOS system license is described in License the system automatically from the CLI (https://techdocs.f5.com/en-us/velos-1-6-0/velos-systems-installation-upgrade/title-install-before-install-upgrade.html#license-chassis-cli).

Installation Process: The Crypto Officer downloads the F5OS-C software image files (ie the module i.e. 1.0.2zc-fips binary and its integrity check file) and deploy it. The VELOS systems (controller or blade platforms) run F5OS-C software packages. 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 FIPS validated module activation requires installation of the license referred as 'FIPS license'. The Crypto Officer should call the show license service (with command "show system licensing"), then verify that the list of license flags includes "FIPS 140 License".

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 and OpenSSL 1.0.2zc-fips.

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/85

11.3 Non-Administrator Guidance

None

11.4 Design and Rules

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) and scenario 1d (SSHv2) in section 2.7.

11.6 End of Life

As the module does not persistently store SSPs, secure sanitization of the module consists of unloading the module. This will zeroize all SSPs in volatile memory.

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
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cipher Feedback
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter Mode
DES	Data Encryption Standard
DF	Derivation Function
DSA	Digital Signature Algorithm
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ESV	Entropy Source Validation
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standards Publication
GCM	Galois Counter Mode
HMAC	Hash Message Authentication Code
KAS	Key Agreement Schema
KAT	Known Answer Test
KW	AES Key Wrap
MAC	Message Authentication Code
NDF	No Derivation Function
NIST	National Institute of Science and Technology
OFB	Output Feedback
PAA	Processor Algorithm Acceleration
PCT	Pairwise Consistency Test

PR	Prediction Resistance
PSS	Probabilistic Signature Scheme
RNG	Random Number Generator
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SSH	Secure Shell
TDES	Triple-DES
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**
January 2024
https://csrc.nist.gov/Projects/cryptographic-module-validation-program/fips-140-3-ig-announcements

FIPS180-4 **Secure Hash Standard (SHS)**
March 2012
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

RFC5649 **Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm**
September 2009
http://www.ietf.org/rfc/rfc5649.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

SP800-56ARev3 **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

SP800-56CRev2 **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

SP800-67 **NIST Special Publication 800-67 Revision 1 - Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher**
January 2012
http://csrc.nist.gov/publications/nistpubs/800-67-Rev1/SP-800-67-Rev1.pdf

SP800-90ARev1 **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

SP800-131A **NIST Special Publication 800-131A Revision 1- Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths**
November 2015
http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf

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

SP800-133Rev2 **NIST Special Publication 800-133 - Recommendation for Cryptographic Key Generation**
June 2020
https://doi.org/10.6028/NIST.SP.800-133r2

SP800-135Rev1 **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