Hewlett Packard Enterprise Development LP

HPE SimpliVity OmniStack Crypto Library

Software Version: 2.1

FIPS 140-2 Non-Proprietary Security Policy

FIPS Security Level: 1
Document Version: 0.7

Prepared for:

Hewlett Packard Enterprise

Hewlett Packard Enterprise Development LP
8 Technology Drive
Westborough, MA 01581
United States of America

Phone: +1 855 788 4636
www.hpe.com

Prepared by:

Corsec

Corsec Security, Inc.

13921 Park Center Road, Suite 460
Herndon, VA 20171
United States of America

Phone: +1 703 267 6050
www.corsec.com

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1 Purpose

This is a non-proprietary Cryptographic Module Security Policy for the HPE SimpliVity OmniStack Crypto Library (software version: 2.1) from Hewlett Packard Enterprise Development LP (HPE). This Security Policy describes how the HPE SimpliVity OmniStack Crypto Library meets the security requirements of Federal Information Processing Standards (FIPS) Publication 140-2, which details the U.S. and Canadian government requirements for cryptographic modules. More information about the FIPS 140-2 standard and validation program is available on the National Institute of Standards and Technology (NIST) and the Communications Security Establishment (CSE) Cryptographic Module Validation Program (CMVP) website at http://csrc.nist.gov/groups/STM/cmvp.

This document also describes how to run the module in a secure FIPS-Approved mode of operation. This policy was prepared as part of the Level 1 FIPS 140-2 validation of the module. The HPE SimpliVity OmniStack Crypto Library is also referred to in this document as the cryptographic module or the module.

1.2 References

This document deals only with operations and capabilities of the module in the technical terms of a FIPS 140-2 cryptographic module security policy. More information is available on the module from the following sources:

- The HPE website (www.hpe.com) contains information on the full line of products from HPE.
- The CMVP website (http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140val-all.htm) contains contact information for individuals responsible for answer technical or sales-related questions for the module.

1.3 Document Organization

The Security Policy document is organized into two (2) primary sections. Section 2 provides an overview of the validated modules. This includes a general description of the capabilities and the use of cryptography, as well as a presentation of the validation level achieved in each applicable functional areas of the FIPS standard. It also provides high-level descriptions of how the modules meet FIPS requirements in each functional area. Section 3 documents the guidance needed for the secure use of the module, including initial setup instructions and management methods and policies.

2. HPE SimpliVity OmniStack Crypto Library

2.1 Overview

To meet growing virtual infrastructure requirements, HPE offers hyperconverged infrastructure solutions designed to meet the needs of IT environments from branch offices to enterprises. Hyperconverged infrastructure consists of a virtual computing platform that combines all IT components and services typically found below the hypervisor into an x86-based "building block". Designed to create efficiencies in virtual computing environments, hyperconverged infrastructure eliminates the need to manage discrete devices or have specialized training in component-level technology (such as SANs[1]).

HPE markets and sells its hyperconverged solutions under the "SimpliVity" brand name. The core technology powering these solutions includes the OmniStack Virtual Controller, or OVC (a software-based file system with data optimization, abstracting data from its underlying hardware), and the OmniStack Accelerator Card, or OAC (a purpose-built PCIe[2] that offloads global inline deduplication, compression, and optimization processes).

HPE packages their hyperconverged infrastructure solution on enterprise-grade x86 platform called the OmniCube (see Figure 1 below). The OmniCube is sold with the HPE software components and the OAC on a 2U server with an Intel Xeon processor and VMware ESXi v5.5 or 6.0 running Ubuntu 14.04.5 LTS[3]. The OmniCube is available in several hardware models that scale for the appropriate workload by offering varying amounts of storage capacity, number of cores, and virtual machine support.

Figure 1 – HPE SimpliVity OmniCube

Another key component of the overall solution is Arbiter. Arbiter is a software agent that is used to break quorum ties between federated OmniCube servers. Arbiter is installed on a host server or virtual machine (VM) running Windows Server 2012 R2.

The HPE SimpliVity OmniStack Crypto Library supplies the cryptographic functionality necessary to support TLS[4]-secured communications between OVC and Arbiter. In this document, the OVC and Arbiter software components (both of which link to the HPE SimpliVity OmniStack Crypto Library for cryptographic service support) will be referred to collectively as the "calling application".

[1] SAN – Storage Area Network
[2] PCIe – Peripheral Computer Interconnect Express
[3] LTS – Long Term Support
[4] TLS – Transport Layer Security

The HPE SimpliVity OmniStack Crypto Library is validated at the FIPS 140-2 Section levels shown in Table 1.

Table 1 – Security Level per FIPS 140-2 Section

Section	Section Title	Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1
3	Roles, Services, and Authentication	1
4	Finite State Model	1
5	Physical Security	N/A
6	Operational Environment	1
7	Cryptographic Key Management	1
8	Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC)	1
9	Self-tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A

2.2 Module Specification

The HPE SimpliVity OmniStack Crypto Library is a software module with a multiple-chip standalone embodiment. The overall security level of the module is 1.

The module consists of a cryptographic library based on the OpenSSL FIPS Canister (OpenSSL FIPS Object Module version 2.0.13). It is compiled into object form (fipscanister.o for Linux-based systems, fipscanister.lib on Windows-based systems) and then statically linked to an instance of the OpenSSL version 1.0.2k libcrypto library at build-time. The OpenSSL libcrypto library is then linked to the calling application (dynamically linked on Linux, statically linked on Windows), which is loaded into memory for execution by the operating system loader.

For FIPS 140-2 conformance testing, the module was tested and found compliant when running on the following operational environments:

- OmniCube CN-3400-12bcc appliance with an Intel Xeon E5-2600 V3 Series processor with VMware ESXi 6.0.0 hosting a virtual machine running Ubuntu 14.04.5 LTS OS[5] (OVC)

- OmniCube CN-3400-12bcc appliance with an Intel Xeon E5-2600 V3 Series processor with VMware ESXi 6.0.0 hosting a virtual machine running Windows Server 2012 R2 OS (Arbiter)

The module implements the FIPS-Approved algorithms listed in Table 2 below.

Table 2 – FIPS-Approved Cryptographic Algorithms

Certificate Number		Algorithm	Standard	Mode / Method	Key Lengths / Curves / Moduli	Use
Ubuntu	Windows					
#5059	#5045	AES[6]	FIPS PUB 197	ECB[7], CBC[8], CTR[9]	128, 192, 256	encryption/decryption
			NIST SP[10] 800-38C	CCM[11]	128, 192, 256	encryption/decryption
			NIST SP 800-38D	GCM[12]	128, 192, 256	encryption/decryption
Vendor Affirmed	Vendor Affirmed	CKG[13]	NIST SP 800-133	-	-	symmetric key generation
#1616	#1599	CVL[14]	NIST SP 800-135rev1	SSH[15], TLS v1.2	-	key derivation
#1617	#1600	CVL	NIST SP 800-56A	ECC CDH[16] Primitive	NIST-defined P curves (P-256 and P-384)	shared secret computation
#1880	#1866	DRBG[17]	NIST SP 800-90A	CTR-based	128, 192, 256	deterministic random bit generation
				Hash-based	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	deterministic random bit generation
				HMAC-based	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	deterministic random bit generation
#1311	#1300	ECDSA[18]	FIPS PUB 186-4	KPG[19]	NIST-defined P curves (P-256 and P-384)	key pair generation
				PKV[20]	NIST-defined P curves (P-256 and P-384)	public key verification
				SigGen, SigVer	NIST-defined P curves (P-256 and P-384)	digital signature generation and verification

[5] OS – Operating System
[6] AES – Advance Encryption Standard
[7] ECB – Electronic Code Book
[8] CBC – Cipher Block Chaining
[9] CTR – Counter
[10] SP – Special Publication
[11] CCM – Counter with CBC-MAC
[12] GCM – Galois Counter Mode
[13] CKG – Cryptographic Key Generation
[14] CVL – Component Validation List
[15] SSH – Secure Shell
[16] ECCCDH – Elliptic Curve Cryptography Cofactor Diffie-Hellman
[17] DBRG – Deterministic Random Bit Generator
[18] ECDSA – Elliptic Curve Digital Signature Algorithm
[19] KPG – Key Pair Generation
[20] PKV – Public Key Verification

#3379	#3365	HMAC[21]	FIPS PUB 198-1	SHA-1, SHA-224, SHA-256. SHA-384, SHA-512	160, 224, 256, 384, 512	message authentication
#2744	#2731	RSA[22]	FIPS PUB 186-4	SigGen9.31, SigGenPKCS[23]1.5, SigGenPSS[24],	2048, 3072	digital signature generation
				SigVer9.31, SigVerPKCS1.5, SigVerPSS	1024, 2048, 3072	digital signature verification
				KeyGen9.31	2048, 3072	key pair generation
#4124	#4110	SHS[25]	FIPS PUB 180-4	SHA[26]-1, SHA-224, SHA-256, SHA-384, SHA-512	-	message digest
#2617	#2605	Triple-DES[27]	NIST SP 800-67rev2	TECB, TCBC	Keying option 1	encryption/decryption

***Note1**: No parts of the SSH and TLS protocols, other than the KDFs, have been tested by the CAVP[28] and CMVP.*
***Note2**: The module explicitly disallows the use of SHA-1 when generating digital signatures.*
***Note3**: The module employs a CKG method whose resulting symmetric key or generated seed is an unmodified output of the DRBG.*

The module also implements the following non-Approved algorithm:

- EC Diffie-Hellman (key agreement: key establishment methodology provides 128 or 192 bits of encryption strength)
    - supports curves P-256 and P-384
    - allowed per FIPS IG D.8 scenario #3

As a software module, the HPE SimpliVity OmniStack Crypto Library has both a logical cryptographic boundary and a physical cryptographic boundary. The physical and logical boundaries are described in Sections 2.2.1 and 2.2.2, respectively.

2.2.1 Physical Cryptographic Boundary

As a software cryptographic module, the module has no physical components. Therefore, the physical boundary of the cryptographic module is defined by the hard enclosure around the host server on which it runs.

The host server hardware, the OmniCube CN-3400-12bcc, consists of a motherboard, a Central Processing Unit (CPU), random access memory (RAM), read-only memory (ROM), hard disk(s), hardware case, power supply, and fans. Other devices may be attached to the hardware appliance such as a monitor, keyboard, mouse, DVD[29] drive, printer, video adapter, audio adapter, or network adapter. In the validated configuration, the processor is an Intel Xeon processor. Please see Figure 2 for a block diagram the host server and a depiction of the physical cryptographic boundary.

[21] HMAC – (keyed-) Hashed Message Authentication Code
[22] RSA – Rivest Shamir Adleman
[23] PKCS – Public Key Cryptography Standard
[24] PSS – Probabilistic Signature Scheme
[25] SHS – Secure Hash Standard
[26] SHA – Secure Hash Algorithm
[27] DES – Data Encryption Standard
[28] CAVP – Cryptographic Algorithm Validation Program
[29] DVD – Digital Versatile Disc

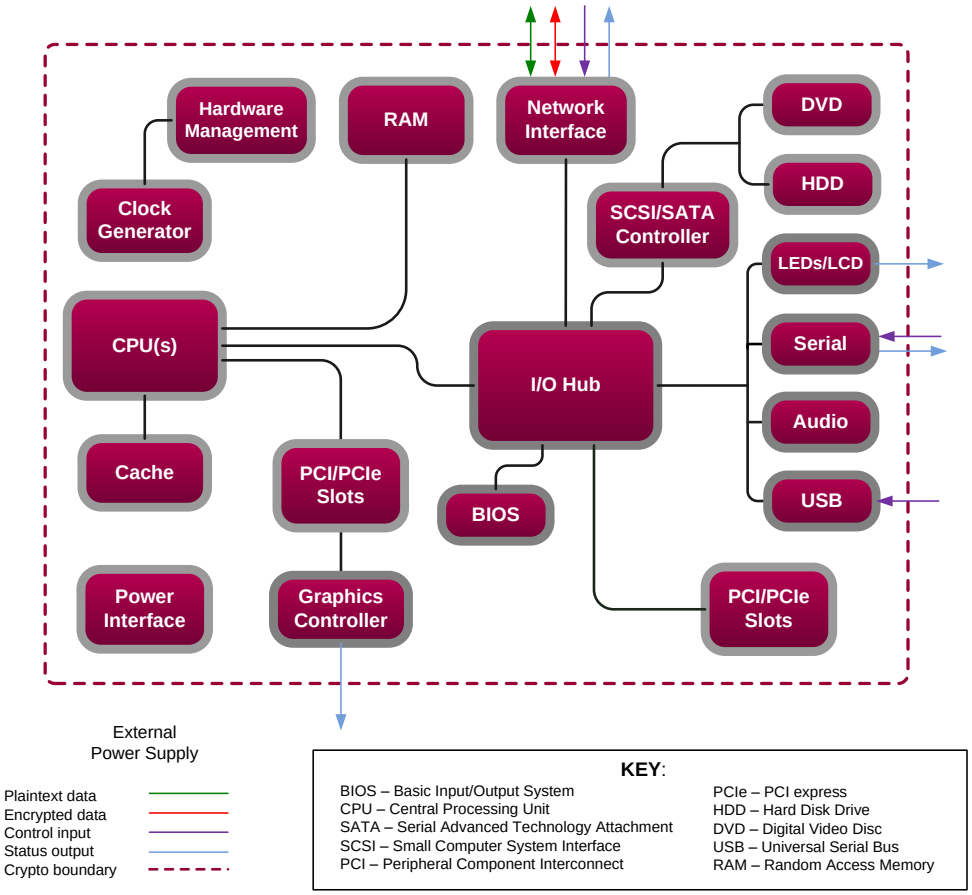

Figure 2 – Host Server Physical Block Diagram

2.2.2 Logical Cryptographic Boundary

The module is used by the calling application to provide symmetric cipher operation, digital signature generation and verification, hashing, cryptographic key generation, random number generation, message authentication functions, and secure key agreement/key exchange protocol support. The module is entirely contained within the physical cryptographic boundary described in Section 2.2.1.

FIGURE 3 shows the logical block diagram of the module executing in memory and its interactions with surrounding software components, as well as the module's logical cryptographic boundary.

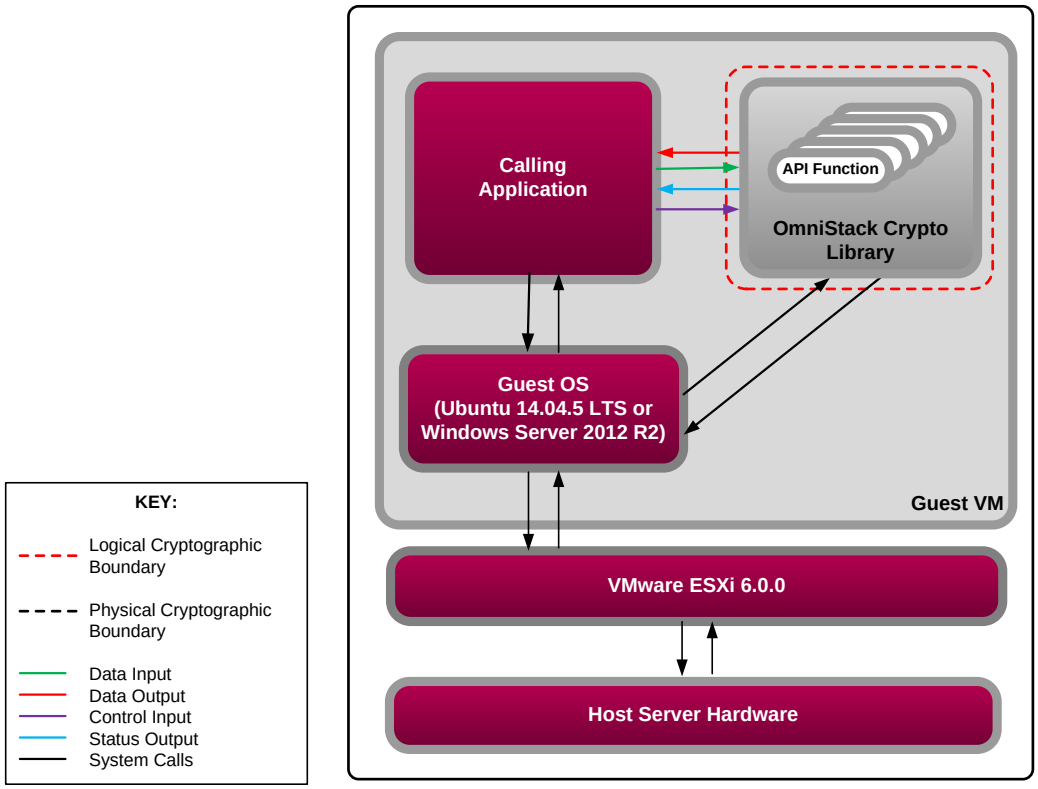

Figure 3 – Logical Block Diagram

2.3 Module Interfaces

The module isolates communications to logical interfaces that are defined in the software as an API[30]. The API interface is mapped to the following four logical interfaces:

- Data Input
- Data Output
- Control Input
- Status Output

The module's physical boundary features the physical ports of a host server. The module's manual controls, physical indicators, and physical, logical, and electrical characteristics are those of the host server. The module's logical interfaces are at a lower level in the software. The physical data and control input through physical mechanisms is translated into the logical data and control inputs for the software module. A mapping of the FIPS 140-2 logical interfaces, the physical interfaces, and the module interfaces can be found in Table 3.

Table 3 – FIPS 140-2 Logical Interface Mappings

FIPS Interface	Physical Interface	Module Interface (API)
Data Input	USB[31] ports (keyboard, mouse, data), network interface, serial ports	The API calls that accept input data for processing through their arguments.

[30] API – Application Programming Interface
[31] USB – Universal Serial Bus

FIPS Interface	Physical Interface	Module Interface (API)
Data Output	Graphics controller, USB ports, network interface, serial ports	The API calls that return, by means of their return codes or arguments, generated or processed data back to the caller.
Control Input	USB ports (keyboard, mouse), network interface, serial ports	The API calls that are used to initialize and control the operation of the module.
Status Output	Graphic controller, network interface, serial ports, Audio ports, LCD[32]/LEDs[33]	Return values for API calls.
Power Input	AC[34] power socket	-

***Note**: As a software module, control of the physical ports is outside the scope of the module.*

2.4 Roles and Services

There are two authorized roles that module operators may assume: Crypto Officer (CO) role and a User role. Since no authentication mechanisms are implemented, roles are assumed implicitly. When invoking a module service, a module operator implicitly assumes the CO and User roles simultaneously; thus, both roles have access to all module services. The module does not allow multiple concurrent operators in the FIPS-Approved mode of operation. As per section 6.1 of the *Implementation Guidance for FIPS PUB 140-2 and the CMVP*, the calling application that loaded the module is the only operator.

Descriptions of the services available are provided in Table 4 below. Please note that the keys and Critical Security Parameters (CSPs) listed in the table indicate the type of access required using the following notation:

- R – Read: The CSP is read.
- W – Write: The CSP is established, generated, modified, or zeroized.
- X – Execute: The CSP is used within an Approved or Allowed security function or authentication mechanism.

Table 4 – Operator Services

Service	Operator		Description	CSP and Type of Access
	CO	User		
Run power-up self-test on demand	✓	✓	Performs power-up self-tests	None
Show status	✓	✓	Returns the current operational mode of the module	None

[32] LCD – Liquid Crystal Display
[33] LED – Light Emitting Diode
[34] AC – Alternating Current

Service	Operator		Description	CSP and Type of Access
	CO	User		
Zeroize	✓	✓	Zeroizes and de-allocates memory containing sensitive data	AES key – W Triple-DES key – W HMAC key – W RSA private/public key – W ECDSA private/public key – W ECDH[35] public/private components – W DRBG Seed – W DRBG Entropy – W DRBG 'C' value – W DRBG 'V' value – W
Zeroize DRBG CSPs	✓	✓	Zeroizes and de-allocates memory containing DRBG CSPs	DRBG Seed – W DRBG Entropy – W DRBG 'C' value – W DRBG 'V' value – W
Generate random number	✓	✓	Returns the specified number of random bits to the calling application	DRBG Seed – WRX DRBG Entropy – RX DRBG 'C' value – WRX DRBG 'V' value – WRX
Generate message digest	✓	✓	Compute and return a message digest using SHS algorithms	None
Generate keyed hash (HMAC)	✓	✓	Compute and return a message authentication code	HMAC key – RX
Generate symmetric key	✓	✓	Generate and return the specified type of symmetric key (Triple-DES or AES)	AES key – W Triple-DES Key – W
Perform symmetric encryption	✓	✓	Encrypt plaintext using supplied key and algorithm specification (Triple-DES or AES)	AES key – RX Triple-DES key – RX
Perform symmetric decryption	✓	✓	Decrypt ciphertext using supplied key and algorithm specification (Triple-DES or AES)	AES key – RX Triple-DES key – RX
Perform authenticated symmetric encryption	✓	✓	Encrypt plaintext using supplied AES GCM key and IV	AES GCM key – RX AES GCM IV – RX
Perform authenticated symmetric decryption	✓	✓	Decrypt ciphertext using supplied AES GCM key and IV	AES GCM key – RX AES GCM IV – RX
Generate asymmetric key pair	✓	✓	Generate and return the specified type of asymmetric key pair (RSA or ECDSA)	RSA private/public key – W ECDSA private/public key – W
Calculate key agreement primitive	✓	✓	Calculate ECDH key agreement primitive	ECDH Public/Private components – WRX
Generate signature	✓	✓	Generate a signature for the supplied message using the specified key and algorithm (RSA or ECDSA)	RSA private key – RX ECDSA private key – RX
Verify signature	✓	✓	Verify the signature on the supplied message using the specified key and algorithm (RSA or ECDSA)	RSA public key – RX ECDSA public key – RX

[35] ECDH – Elliptic Curve Diffie-Hellman

Service	Operator		Description	CSP and Type of Access
	CO	User		
Perform SSH key derivation	✓	✓	Establish SSH protocol keys via SSH KDF	SSH shared secret – R, X AES key – W AES GCM key – W Triple-DES key – W HMAC key – W
Perform TLS key derivation	✓	✓	Establish TLS protocol keys via TLS KDF	TLS pre-master secret – R, X TLS master secret – W AES key – W AES GCM key – W Triple-DES key – W HMAC key – W

2.5 Physical Security

The cryptographic module is a software module and does not include physical security mechanisms. Therefore, as per section G.3 of the *Implementation Guidance for FIPS PUB 140-2 and the CMVP*, requirements for physical security are not applicable.

2.6 Operational Environment

The module was tested and found to be compliant with FIPS 140-2 requirements on the following platforms and environments:

- HPE OmniCube CN-3400-12bcc appliance with an Intel Xeon E5-2600 V3 Series processor with VMware ESXi 6.0.0 hosting a virtual machine running Ubuntu 14.04.5 LTS OS (OVC)

- HPE OmniCube CN-3400-12bcc appliance with an Intel Xeon E5-2600 V3 Series processor with VMware ESXi 6.0.0 hosting a virtual machine running Windows Server 2012 R2 OS (Arbiter)

As per section 6.1 of the *Implementation Guidance for FIPS PUB 140-2 and the CMVP*, the application that makes calls to the cryptographic module is the single user of the cryptographic module, even when the application is serving multiple clients.

All cryptographic keys and CSPs are under the control of the OS, which protects its CSPs against unauthorized disclosure, modification, and substitution. Additionally, the OS provides dedicated process space to each executing process, and the module operates entirely within the calling application's process space. The module only allows access to CSPs through its well-defined API.

2.7 Cryptographic Key Management

The module supports the CSPs listed below in Table 5.

Table 5 – Cryptographic Keys, Cryptographic Key Components, and CSPs

CSP	CSP Type	Generation/Input	Output	Storage	Zeroization	Use
AES key	AES 128, 192, 256-bit key	Internally generated via Approved DRBG OR Internally derived via TLS/SSH KDF OR Input via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Power cycle/reboot; remove power; unload module	Encryption, decryption
AES GCM key	AES GCM 128, 192, 256-bit key	Internally generated via Approved DRBG OR Internally derived via TLS/SSH KDF OR Input via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Power cycle/reboot; remove power; unload module	Encryption, decryption
AES GCM IV[36]	96-bit value	Internally generated via Approved DRBG (per SP 800-38D section 8.2.1)	Never output from the module	Plaintext in volatile memory	Power cycle/reboot; remove power; unload module	Initialization vector for AES GCM

[36] IV – Initialization Vector

CSP	CSP Type	Generation/Input	Output	Storage	Zeroization	Use
Triple-DES key	Triple-DES 168-bit key (Keying option 1)	Internally generated via Approved DRBG OR Internally derived via TLS/SSH KDF OR Input via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Power cycle/reboot; remove power; unload module	Encryption, decryption
HMAC key	HMAC 160, 224, 256, 384, or 512-bit key	Internally generated via Approved DRBG OR Internally derived via TLS/SSH KDF OR Input via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Power cycle/reboot; remove power; unload module	Message authentication with SHS
TLS pre-master secret	Pre-master secret value	Input via API call parameter	Never output from the module	Keys are not persistently stored by the module	Power cycle/reboot; remove power; unload module	Derivation of the TLS master secret
TLS master secret	Master secret value	Internally derived via TLS KDF	Never output from the module	Keys are not persistently stored by the module	Power cycle/reboot; remove power; unload module	Derivation of AES and Triple-DES session keys and HMAC authentication keys used for TLS
SSH shared secret	Shared secret value	Input via API call parameter	Never output from the module	Keys are not persistently stored by the module	Power cycle/reboot; remove power; unload module	Derivation of AES and Triple-DES session keys and HMAC authentication keys used for TLS

CSP	CSP Type	Generation/Input	Output	Storage	Zeroization	Use
RSA private key	RSA 2048 or 3072-bit key	Internally generated per FIPS PUB 186-4 OR Input via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Power cycle/reboot; remove power; unload module	Signature generation
RSA public key	RSA 1024, 2048, 3072-bit key	Internally generated per FIPS PUB 186-4 OR Input via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Power cycle/reboot; remove power; unload module	Signature verification
ECDSA private key	NIST-defined P curves (P-256 and P-384)	Internally generated per FIPS PUB 186-4 OR Input via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Power cycle/reboot; remove power; unload module	Signature generation
ECDSA public key	NIST-defined P curves (P-256 and P-384)	Internally generated per FIPS PUB 186-4 OR Input via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Power cycle/reboot; remove power; unload module	Signature verification
ECDH private component	NIST-defined P curves (P-256 and P-384)	Internally generated via Approved DRBG OR Input via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Power cycle/reboot; remove power; unload module	Used by host application
ECDH public component	NIST-defined P curves (P-256 and P-384)	Internally generated via Approved DRBG OR Input via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Power cycle/reboot; remove power; unload module	Used by host application

CSP	CSP Type	Generation/Input	Output	Storage	Zeroization	Use
DRBG Seed	Random data – 256-bits	Generated internally using nonce along with DRBG entropy input	Never output from the module	Keys are not persistently stored by the module	Power cycle/reboot; remove power; unload module; API call	Seeding material for SP 800-90A DRBGs
DRBG Entropy	256-bit value	Externally generated and input via API call parameter	Never output from the module	Keys are not persistently stored by the module	Power cycle/reboot; remove power; unload module; API call	Entropy material for SP 800-90A DRBGs
DRBG 'C' Value	Internal state value	Internally generated	Never output from the module	Keys are not persistently stored by the module	Power cycle/reboot; remove power; unload module; API call	Used for Hash_DRBG
DRBG 'V' Value	Internal state value	Internally generated	Never output from the module	Keys are not persistently stored by the module	Power cycle/reboot; remove power; unload module; API call	Used for Hash_DRBG, HMAC_DRBG, and CTR_DRBG
DRBG 'Key' Value	Internal state value	Internally generated	Never output from the module	Keys are not persistently stored by the module	Power cycle/reboot; remove power; unload module; API call	Used for HMAC_DRBG and CTR_DRBG

Caveat: *The module generates cryptographic keys whose strengths are modified by the available entropy. No assurance of the minimum strength of generated keys.*

2.8 EMI / EMC

The HPE SimpliVity OmniStack Crypto Library was tested on the servers listed in Section 2.6 above. These servers were tested and found conformant to the EMI/EMC requirements specified by 47 Code of Federal Regulations, Part 15, Subpart B, Unintentional Radiators, Digital Devices, Class A (business use).

2.9 Self-Tests

Cryptographic self-tests are performed by the module when the module is first loaded into memory as well as when a random number or asymmetric key pair is generated. The calling application can also invoke the power-up self-tests on demand by performing the "Run self-test on demand" service via the `FIPS_selftest()` API.

The following sections list self-tests performed by the module, their expected error status, and the error resolutions.

2.9.1 Power-Up Self-Tests

The module performs the following self-tests at power-up:

- Software integrity check (using HMAC-SHA1)
- Known Answer Tests (KATs)
  - AES-ECB encrypt KAT
  - AES-ECB decrypt KAT
  - AES-CCM encrypt KAT
  - AES-CCM decrypt KAT
  - AES-GCM encrypt KAT
  - AES-GCM decrypt KAT
  - Triple-DES encrypt KAT
  - Triple-DES decrypt KAT
  - RSA signature generation/verification test
  - HMAC KATs (with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512)
  - CTR_DRBG KAT
  - Hash DRBG KAT
  - HMAC DRBG KAT
  - ECDH KAT
- Pairwise Consistency Tests
  - ECDSA pairwise consistency test

The module implements an HMAC KAT for each associated SHA implementation (SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512). These KATs utilize (and thus test) the full functionality of the SHA-1 and SHA-2 implementations; thus, no independent KATs for SHA-1 and SHA-2 implementations are required.

2.9.2 Conditional Self-Tests

The module performs the following self-tests on power-up and conditionally:

- Continuous RNG[37] Test for the DRBG
- Continuous RNG Test for the NDRNG
- RSA pairwise consistency test for key pair generation
- ECDSA pairwise consistency test for key pair generation

2.9.3 Critical Functions Self-Tests

The module performs the following critical functions tests on power-up and conditionally:

- SP 800-90 DRBG (Hash, HMAC, CTR) Instantiate Test
- SP 800-90 DRBG (Hash, HMAC, CTR) Generate Test
- SP 800-90 DRBG (Hash, HMAC, CTR) Reseed Test

2.9.4 Self-Test Failure Handling

If the module passes all the self-tests, it will return a value of "1" (indicating success) to the calling application, which indicates that the module is ready to be placed in its FIPS-Approved mode of operation.

If any of the above self-test fails, the self-test function returns a value of "0" (indicating failure). The module then ceases all cryptographic functionality and enters the critical error state. While in the critical error state, the module only returns "failure" status if any subsequent cryptographic services are requested. The module can only recover from the critical error state by rebooting the host server (thus restarting the module) and passing all power-on self-tests.

If rebooting the host server does not result in the successful execution of the self-tests, then the module will not be able to resume normal operations, and the CO should contact Hewlett Packard Enterprise Development LP.

2.10 Mitigation of Other Attacks

This section is not applicable. The module does not claim to mitigate any attacks beyond the FIPS 140-2 Level 1 requirements for this validation.

[37] RNG – Random Number Generator

3. Secure Operation

The sections below describe how to place and keep the HPE SimpliVity OmniStack Crypto Library in the FIPS-Approved mode of operation. **Operating the module without following the guidance herein (including the use of undocumented services) will result in non-compliant behavior and is outside the scope of this Security Policy**.

Please note that the HPE SimpliVity OmniStack Crypto Library is not delivered to end-users as a standalone offering. Rather, it is an integrated component of the OmniCube and OmniStack product software running on the OVC and Arbiter. This product software is pre-installed at the factory on the host server hardware prior to delivery to end-users. HPE does not provide end-users with any mechanisms to directly access the module, its APIs, or any information sent to/from the calling application.

3.1 Installation

The module is part of a product software package that is factory-installed. Thus, the module has no independent installation steps that end-users must follow.

3.2 Initialization

The module's power-up self-tests are automatically executed when the module is invoked by the calling application.

The module has a defined default entry point (DEP) containing code that the OS loader executes automatically when the library is loaded into memory for execution (but prior to the calling application assuming process control over the library). The DEP invokes self-test code via the `FIPS_module_mode_set()` API with a non-zero parameter. This action sets an internal FIPS mode flag to 'TRUE', placing the module in its Approved mode. No additional initialization or configuration steps are required to be performed by end-users.

3.3 Operator Guidance

The following sections provide guidance to module operators for the correct and secure operation of the module. As it relates to this module, the calling application (i.e. the product software) takes on the role of both Crypto Officer and User.

3.3.1 Crypto Officer Guidance

Subsequent to the successful invocation of the `FIPS_module_mode_set()` API to place the module in its Approved mode of operation, no further management activities are required to ensure that the module runs securely; once initialized, the module only executes in a FIPS-Approved mode of operation. However, if any irregular activity is noticed or the module is consistently reporting errors, then HPE Customer Support should be contacted.

3.3.2 User Guidance

The User does not have any ability to modify the configuration of the module. However, if any irregular activity is noticed or the module is consistently reporting errors, then HPE Customer Support should be contacted.

3.3.3 General Operator Guidance

The following provide further guidance for the general operation of the module:

- The module does not store any CSP persistently (beyond the lifetime of an API call), with the exception of DRBG state values used for the module's default key generation service. Zeroization of sensitive data is performed automatically by API function calls for temporarily stored CSPs.

  The module stores DRBG state values for the lifetime of the DRBG instance. The `FIPS_drbg_uninstantiate()` API can be used to explicitly destroy CSPs related to random number generation services.

- To determine the module's operational status, the `FIPS_mode()` API can be used. A non-zero return value indicates that the module is running in its FIPS-Approved mode; a "0" indicates non-FIPS mode.

- To execute the module's power-up self-tests on-demand, the module's host server can be rebooted/power-cycled. Additionally, the `FIPS_selftest()` API can be used. A return value of "1" indicates that all self-tests completed successfully; a "0" indicates a test failure.

3.4 Additional Guidance and Usage Policies

The notes below provide additional guidance and policies that must be followed by module operators:

- As a software cryptographic library, the module's services are intended to be provided to a calling application. Excluding the use of the NIST-defined elliptic curves as trusted third-party domain parameters, all other assurances from FIPS 186-4 (including those required of the intended signatory and the signature verifier) are outside the scope of the module, and are the responsibility of the calling application.

- The calling application uses entropy sources that meets the security strength required for the DRBG as shown in NIST SP 800-90A, Table 2 (Hash DRBG and HMAC DRBG) and Table 3 (CTR DRBG). This entropy is supplied by means of a callback function. The callback function must return an error if the minimum entropy strength cannot be met.

- Module operators are responsible for ensuring that the module performs no more than 2^{16} 64-bit data block encryptions under the same three-key Triple-DES key.

- As the module does not persistently store keys, the calling application is responsible for the storage and zeroization of keys and CSPs passed into and out of the module.

- In the event that power to the module is lost and subsequently restored, the calling application must ensure that any AES-GCM keys used for encryption or decryption are re-distributed. Further, the calling

application must ensure that the AES-GCM key/IV uniqueness requirements described in section A.5 of the *Implementation Guidance for FIPS PUB 140-2 and the CMVP* are met.

- In the event that the module encounters a DRBG self-test failure, the calling application must uninstantiate and reinstantiate the DRBG per the requirements found in NIST SP 800-90A.

3.5 Non-FIPS-Approved Mode

When built and distributed as described in this Security Policy, the module does not support a non-Approved mode of operation.

4. Acronyms

Table 6 provides definitions for the acronyms used in this document.

Table 6 – Acronyms

Acronym	Definition
AC	Alternating Current
AES	Advanced Encryption Standard
API	Application Programming Interface
CBC	Cipher Block Chaining
CCM	Counter with CBC-MAC
CKG	Cryptographic Key Generation
CMVP	Cryptographic Module Validation Program
CO	Cryptographic Officer
CPU	Central Processing Unit
CSE	Communications Security Establishment
CSP	Critical Security Parameter
CTR	Counter
CVL	Component Validation List
DEP	Default Entry Point
DES	Data Encryption Standard
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
DVD	Digital Versatile Disc
ECB	Electronic Code Book
ECDH	Elliptic Curve Diffie-Hellman
ECCCDH	Elliptic Curve Cryptography Cofactor Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EMI/EMC	Electromagnetic Interference /Electromagnetic Compatibility
FIPS	Federal Information Processing Standard
GCM	Galois/Counter Mode
HMAC	Hash Message Authentication Code
KAS	Key Agreement Scheme
KAT	Known Answer Test
KPG	Key Pair Generation
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LTS	Long Term Support

Acronym	Definition
NDRNG	Non-Deterministic Random Number Generator
NIST	National Institute of Standards and Technology
OAC	OmniStack Accelerator Card
OS	Operating System
OVC	OmniStack Virtual Controller
PCIe	PCI express
PKCS	Public Key Cryptography Standard
PKV	Public Key Verification
PSS	Probabilistic Signature Scheme
RAM	Random Access Memory
RNG	Random Number Generator
RSA	Rivest, Shamir, and Adleman
SAN	Storage Area Network
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SP	Special Publication
SSH	Secure Shell
TDES	Triple Data Encryption Standard
TLS	Transport Layer Security
USB	Universal Serial Bus

Prepared by:

Corsec Security, Inc.

13921 Park Center Road, Suite 460
Herndon, VA 20171
United States of America

Phone: +1 703 267 6050
Email: info@corsec.com
http://www.corsec.com