Qumulo, Inc.

Qumulo Secure

Version: 1.0

FIPS 140-2 Non-Proprietary Security Policy

FIPS Security Level: 1
Document Version: 0.5

Prepared for:

Qumulo, Inc.
1501 4th Avenue, Suite 1600
Seattle, WA 98101
United States of America

Phone: +1 206 260 3588
www.qumulo.com

Prepared by:

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 Qumulo Secure 1.0 from Qumulo, Inc. (Qumulo). This Security Policy describes how Qumulo Secure meets the security requirements of Federal Information Processing Standards (FIPS) Publication 140-2.

FIPS 140-2 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 Cryptographic Module Validation Program (CMVP) website, which is maintained by National Institute of Standards and Technology (NIST) and Canadian Centre for Cyber Security (CCCS).

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. Qumulo Secure is also referred to in this document as 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 Qumulo website (www.qumulo.com) contains information on the full line of products from Qumulo.
- The search page on the CMVP website (https://csrc.nist.gov/Projects/cryptographic-module-validation-program/Validated-Modules/Search) can be used to locate and obtain vendor contact information for technical or sales-related questions about the module.

1.3 Document Organization

The Security Policy document is organized into two primary sections. Section 2 provides an overview of the validated module. 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 area of the FIPS standard. It also provides high-level descriptions of how the module meets 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. Qumulo Secure

2.1 Overview

Qumulo, Inc. is the leader in hybrid cloud file storage solutions, enabling enterprises to manage and store billions of digital assets, and providing real-time visibility, scale, and control of data across on-prem and cloud-based platforms. Their Qumulo Core file system solution is a modern scale-out storage system designed from the ground up to handle multiple petabytes of data both on premises and in the cloud. Qumulo Core uses advanced software-based techniques that give storage administrators up-to-the-minute file system analytics, intelligence, and control. With a block-based data protection mechanism, Qumulo Core uses raw capacity far more efficiently than other systems, especially as the number of files becomes large.

Figure 1 shows a diagram of the Qumulo Core software solution's architecture.

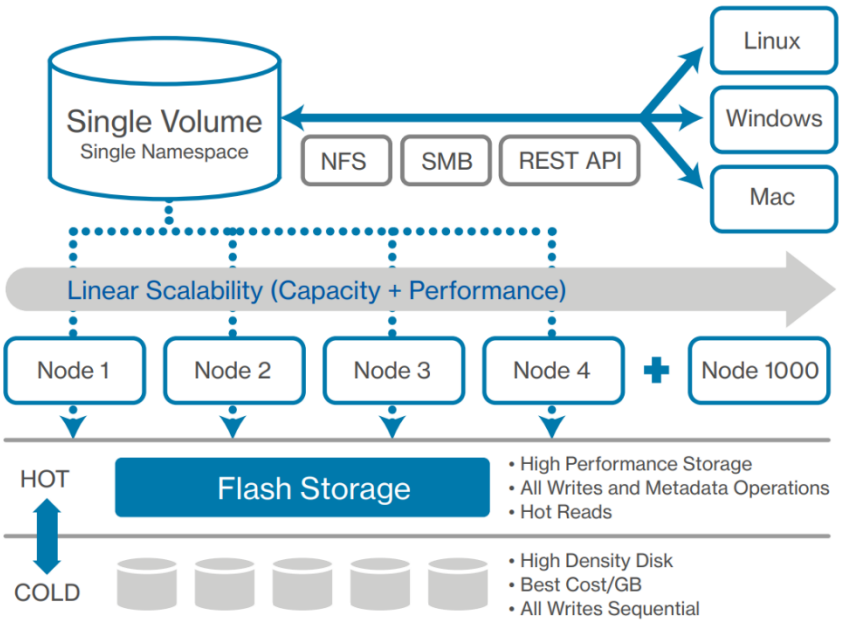

Figure 1 – Qumulo Core Solution Architecture

Qumulo Core has a "flash-first" hybrid design that uses both solid-state drives (SSD) and high-density hard disk drives (HDD). This balanced design optimizes capacity, performance, and cost for a wide range of workloads.

For scalability, Qumulo's file system solution has a distributed architecture where many individual computing nodes work together to form a cluster with scalable performance and a single, unified file system. Qumulo clusters, in turn, work together to form a globally distributed but highly connected storage fabric tied together with continuous replication relationships. Figure 2 below shows the connections between clients, Qumulo clusters comprised of nodes running Qumulo Core, and multiple Qumulo clusters, comprising the fabric, running in multiple environments and geographic locations.

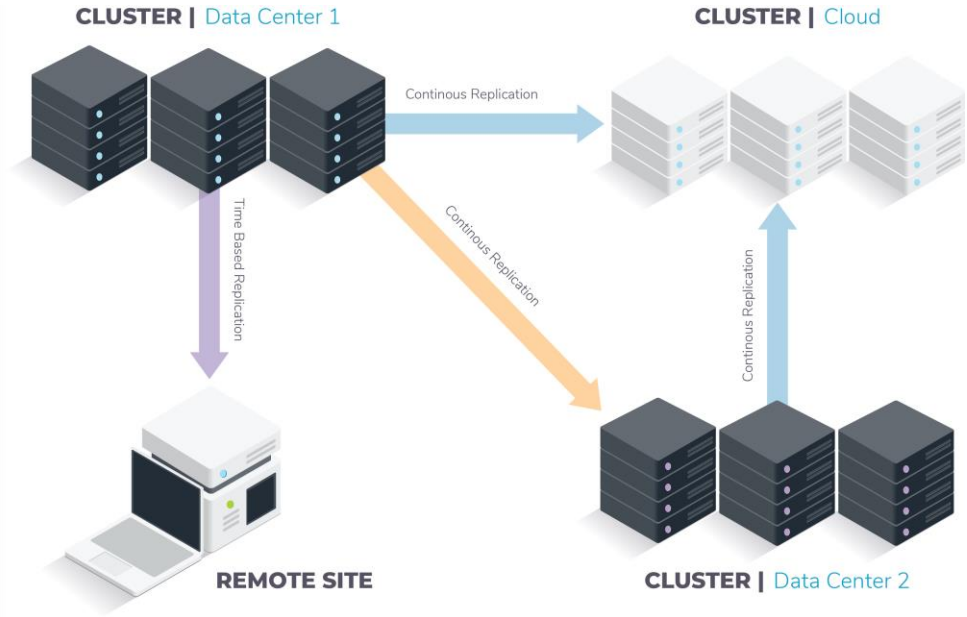

Figure 2 – Qumulo Core Fabric

The file system itself has been designed for massive scale. There is no preset limit to the number of files or their sizes. Unlike other systems, Qumulo Core can accommodate, with real-time management, billions of files and directories. Qumulo Core uses techniques similar to those used for large, distributed databases to manage file metadata. Using a proprietary distributed algorithm, it also maintains totals, updated dynamically, of file sizes and counts at each directory level. The aggregated metadata enables many of the advanced sampling techniques used in the Qumulo Core dashboard displays.

The Qumulo Secure v1.0 cryptographic module is a component of the Qumulo Core software solution (versions 3.x.x and later). Qumulo Secure offers symmetric encryption/decryption, digital signature generation/verification, hashing, cryptographic key generation, random number generation, message authentication, and key establishment functions to secure data-at-rest/data-in-flight and to support secure communications protocols (including TLS[1] 1.2).

The Qumulo Secure 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	1

[1] TLS – Transport Layer Security

Section	Section Title	Level
6	Operational Environment	1
7	Cryptographic Key Management	1
8	EMI/EMC[2]	1
9	Self-tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A[3]

The module has an overall security level of 1.

2.2 Module Specification

The Qumulo Secure is a software-hybrid cryptographic module with an overall security level of 1. As a software-hybrid, the module consists of disjoint software and hardware components. All hardware and software components are contained within the host platform's physical enclosure.

The module's hardware component is the host platform's processor. In the tested configuration, the host platform employs a six-core, 2.2 GHz Intel® Xeon® D-1531 CPU[4] (see Figure 3 below).

Figure 3 – Intel Xeon D-1531

The module's software component comprises a proprietary variant of the OpenSSL 1.1.1 library files (libssl.so and libcrypto.so), a wrapper (libqumulo_secure.so) that controls access to the module's functionality, and a digest file (qumulo_secure.hmac) for testing integrity. Collectively, the software component is version 1.0.

The module is designed to exclusively utilize the AES-NI[5] and AVX[6] extended instruction sets provided by the host platform's CPU for processor algorithm acceleration (PAA) of all of its AES and SHA implementations. For this validation, the module was tested and found to be compliant with FIPS 140-2 requirements on the platform(s) listed in Table 2.

[2] EMI/EMC – Electromagnetic Interference / Electromagnetic Compatibility
[3] N/A – Not applicable
[4] CPU – Central Processing Unit
[5] AES-NI – Advanced Encryption Algorithm New Instructions
[6] AVX – Advanced Vector Extensions

Table 2 – Tested Platforms

Platform	Operating System	Processor
Qumulo C72T	Ubuntu 18.04 LTS	Intel® Xeon® D-1531

The vendor affirms the module's continued validation compliance when operating on the platforms listed in Table 3.

Table 3 – Vendor-Affirmed Platforms

Platform	Operating System	Processor
Qumulo QC24	Ubuntu 18.04 LTS	Intel® Xeon®
Qumulo QC40	Ubuntu 18.04 LTS	Intel® Xeon®
Qumulo QC104	Ubuntu 18.04 LTS	Intel® Xeon®
Qumulo QC208	Ubuntu 18.04 LTS	Intel® Xeon®
Qumulo QC260	Ubuntu 18.04 LTS	Intel® Xeon®
Qumulo QC360	Ubuntu 18.04 LTS	Intel® Xeon®
Qumulo K144T	Ubuntu 18.04 LTS	Intel® Xeon®
Qumulo K168T	Ubuntu 18.04 LTS	Intel® Xeon®
Qumulo C168T	Ubuntu 18.04 LTS	Intel® Xeon®
Qumulo C192T	Ubuntu 18.04 LTS	AMD EPYC
Qumulo C432T	Ubuntu 18.04 LTS	AMD EPYC
Qumulo P23T	Ubuntu 18.04 LTS	Intel® Xeon®
Qumulo P92T	Ubuntu 18.04 LTS	Intel® Xeon®
Qumulo P184T	Ubuntu 18.04 LTS	Intel® Xeon®
Qumulo P368T	Ubuntu 18.04 LTS	Intel® Xeon®
HPE Gen9 90T	Ubuntu 18.04 LTS	Intel® Xeon®
HPE Gen9 180T	Ubuntu 18.04 LTS	Intel® Xeon®
HPE Gen9 288T	Ubuntu 18.04 LTS	Intel® Xeon®
HPE Gen10 36T	Ubuntu 18.04 LTS	Intel® Xeon®
HPE Gen10 90T	Ubuntu 18.04 LTS	Intel® Xeon®
HPE Gen10 192T	Ubuntu 18.04 LTS	Intel® Xeon®
HPE Gen10 336T	Ubuntu 18.04 LTS	Intel® Xeon®

Further, per section G.5 of *the Implementation Guidance for FIPS PUB 140-2 and the CMVP*, the cryptographic module maintains validation compliance when operating on any compatible GPC provided that the GPC includes a processor with the AES-NI and AVX extended instruction sets and uses a single-user operating system/mode

specified on the validation certificate, or another compatible single-user operating system. Note that such a GPC may be deployed on-prem or in a supported public cloud environment (Amazon Web Services, Google Cloud Platform, or Microsoft Azure).

The CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when ported to an operational environment not listed on the validation certificate.

2.2.1 Physical Cryptographic Boundary

As a software-hybrid cryptographic module, the module takes on the physical characteristics of the host platform. The module's physical cryptographic boundary is defined as the metal enclosure of the host platform; the host platform's enclosure fully encompasses the module's disjoint software and hardware components. The host platform's processor is the hardware component of the software-hybrid module, and it provides acceleration for the module's AES and SHA software implementations via its available AES-NI and AVX extended instruction sets.

The processor interfaces with the host platform's other hardware components, including the motherboard, RAM, read-only memory (ROM), hard disk(s), power supply, and fans. Other devices may be attached to the platform, such as a monitor, keyboard, mouse, DVD drive, fixed disk drive, printer, video adapter, audio adapter, or network adaptor.

Please see Figure 4 below for a block diagram of a typical GPC, as well as a depiction of the hardware component of the software-hybrid module and the physical cryptographic boundary.

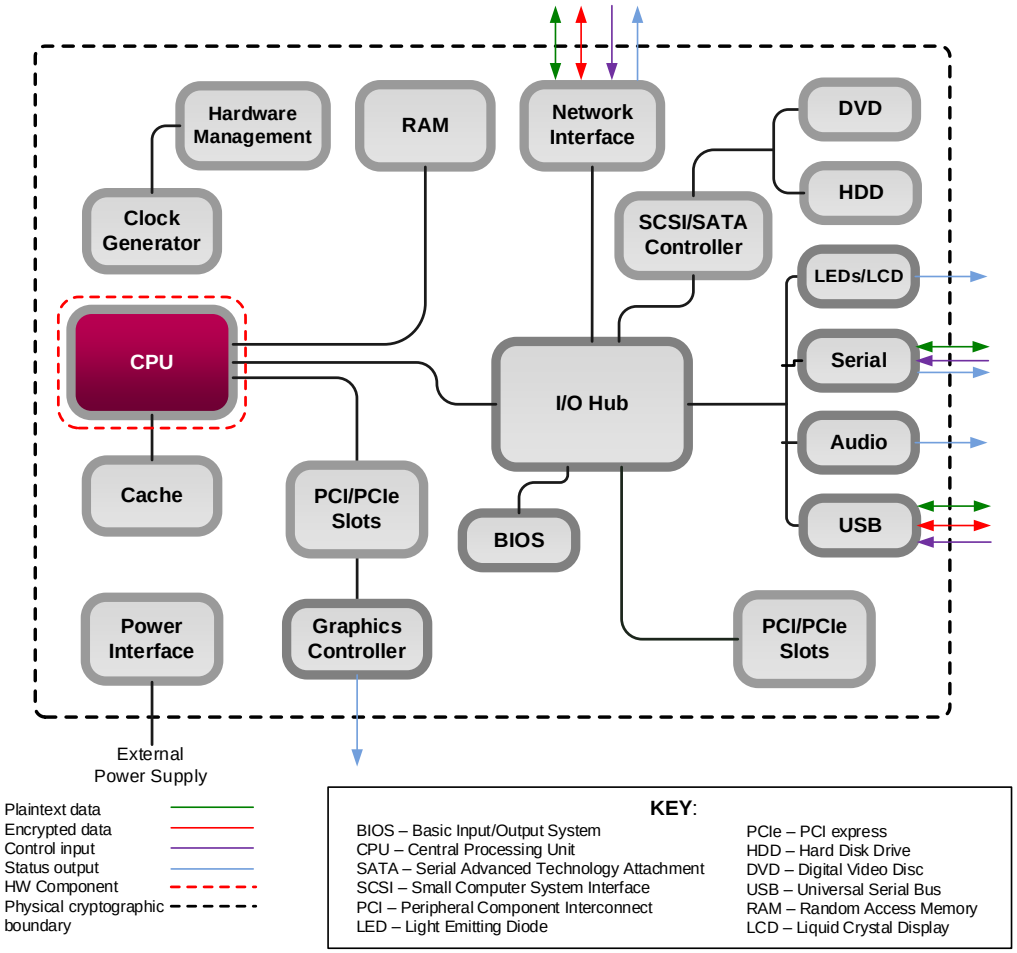

Figure 4 – GPC Block Diagram (with Module's Hardware Component)

2.2.2 Logical Cryptographic Boundary

The module's logical cryptographic boundary encompasses the module's software components. Qumulo Secure's software component provides encryption/decryption, hashing, digital signature functions, MAC[7] functions, and random bit generation; key agreement/transport schemes; and support for the TLS 1.2 protocol. The module utilizes the AES-NI and AVX instruction sets provided by the host platform's CPU for processor algorithm acceleration (PAA) of all of its AES and SHA implementations.

Figure 5 below is a logical block diagram depicting the module's software components executing in memory and their interactions with surrounding software components, as well as the module's physical cryptographic boundary.

[7] MAC – Message Authentication Code

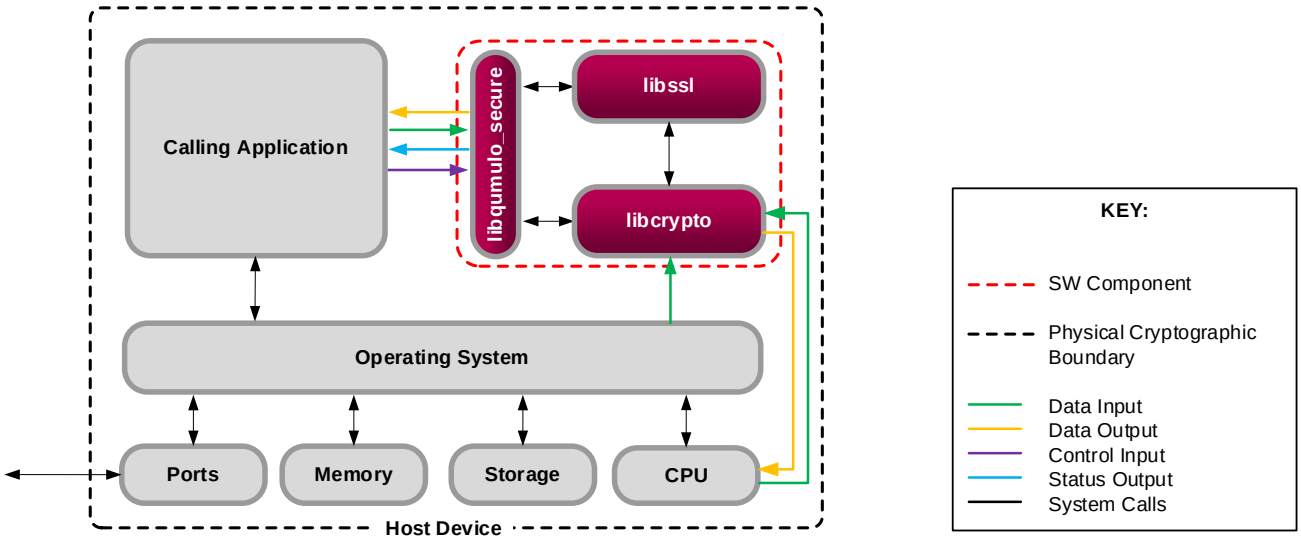

Figure 5 – Module Block Diagram (with Module's Software Component)

2.2.3 Modes of Operation

The module supports two modes of operation: Approved and non-Approved.

As allowed per section 1.19 of the *Implementation Guidance for FIPS PUB 140-2 and the CMVP*, the module alternates on a service-by-service basis between Approved and non-Approved modes of operation. Once the module is operational and the self-tests have completed successfully, the module executes in the Approved mode of operation by default. The module switches to its non-Approved mode whenever it executes a non-Approved/non-compliant security function. The module switches back to the Approved mode when the non-Approved/non-compliant security function is complete.

Critical security parameters are not shared between modes.

Section 2.4.2 below lists the services that constitute the Approved mode and the non-Approved mode.

2.2.4 Algorithm Implementations

The module includes an AES-XTS implementation in `libqmulo.secure.so` and a TLS 1.2 key derivation function (KDF) in `libssl.so`. The module's remaining cryptographic primitives are implemented in Qumulo's proprietary variant of `libcrypto.so`. The algorithm implementation providers are listed in Table 4 below.

Table 4 – Algorithm Implementation Providers

Certificate Number	Implementation Name	Version	Use
A1392	Qumulo Secure Cryptographic Library	1.0	Cryptographic primitives
A1393	Qumulo Secure XTS	1.0	XTS-AES implementation

Certificate Number	Implementation Name	Version	Use
A1394	Qumulo Secure TLS KDF	1.0	TLS v1.2 KDF implementation

All FIPS-Approved algorithms implemented by the module are listed in Table 5 below.

Table 5 – FIPS-Approved Cryptographic Algorithms

Certificate Number	Algorithm	Standard	Modes / Methods	Key Lengths / Curves / Moduli	Use
A1392	AES[8]	FIPS PUB[9] 197 NIST SP 800-38A	CBC[10], CFB1[11], CFB8, CFB128, CTR[12], ECB[13], OFB[14]	128, 192, 256	Encryption/decryption
		NIST SP 800-38B	CMAC[15]	128, 192, 256	MAC Generation/verification
		NIST SP 800-38C	CCM[16]	128, 192, 256	Encryption/decryption
		NIST SP 800-38D	GCM	128, 192, 256	Encryption/decryption
		NIST SP 800-38D	GMAC[17]	128, 192, 256	MAC generation/verification
		NIST SP 800-38F	KW[18], KWP[19]	128, 192, 256	Key wrapping/unwrapping
A1393	AES	NIST SP 800-38E	XTS[20,21,22]	256	Encryption/decryption
Vendor Affirmed	CKG[23]	NIST SP 800-133	-	-	Cryptographic key generation
A1394	CVL[24]	NIST SP 800-135rev1	TLS 1.2	-	Key derivation
A1392	DRBG[25]	NIST SP 800-90Arev1	Counter-based (derivation function - yes; prediction resistance - no)	128, 192, 256-bit AES-CTR	Deterministic random bit generation
A1392	DSA[26]	FIPS PUB 186-4	-	2048, 3072	Key pair generation

[8] AES – Advanced Encryption Standard
[9] PUB – Publication
[10] CBC – Cipher Block Chaining
[11] CFB – Cipher Feedback
[12] CTR – Counter
[13] ECB – Electronic Code Book
[14] OFB – Output Feedback
[15] CMAC – Cipher-Based Message Authentication Code
[16] CCM – Counter with CBC-MAC
[17] GMAC – Galois Message Authentication Code
[18] KW – Key Wrap
[19] KWP – Key Wrap with Padding
[20] XOR – Exclusive OR
[21] XEX – XOR Encrypt XOR
[22] XTS – XEX-Based Tweaked-Codebook Mode with Ciphertext Stealing
[23] CKG – Cryptographic Key Generation
[24] CVL – Component Validation List
[25] DRBG – Deterministic Random Bit Generator
[26] DSA – Digital Signature Algorithm

Certificate Number	Algorithm	Standard	Modes / Methods	Key Lengths / Curves / Moduli	Use
			SHA2-224, SHA2-256, SHA2-384, SHA2,512	2048, 3072	Domain parameter generation
			SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2,512	1024, 2048, 3072	Domain parameter verification
			SHA2-224, SHA2-256, SHA2-384, SHA2,512	2048, 3072	Digital signature generation
			SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2,512	1024, 2048, 3072	Digital signature verification
A1392	ECDSA[27]	FIPS PUB 186-4	-	B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521	Key pair generation
			SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2,512	B-163, B-233, B-283, B-409, B-571, K-163, K-233, K-283, K-409, K-571, P-192, P-224, P-256, P-384, P-521	Public key validation
			SHA2-224, SHA2-256, SHA2-384, SHA2,512	B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521	Digital signature generation
			SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2,512	B-163, B-233, B-283, B-409, B-571, K-163, K-233, K-283, K-409, K-571, P-192, P-224, P-256, P-384, P-521	Digital signature verification
A1392	HMAC[28]	FIPS PUB 198-1	SHA-1-96, SHA-1, SHA-224, SHA2-256, SHA2-384, SHA2-512	112 (minimum)	Message authentication
A1392	KAS-SSC[29]	NIST SP 800-56Arev3	FFC[30] DH	2048/224, 2048/256	Shared secret computation
			ECC CDH[31]	B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521	Shared secret computation
A1392	KTS[32]	NIST SP 800-38D	AES-GCM[33]	128, 192, 256	Key transport (authenticated encryption)
		FIPS PUB 197 NIST SP 800-38B	AES with CMAC[34]	128, 192, 256	Key transport (encryption with message authentication)
		FIPS PUB 197 FIPS PUB 198-1	AES with HMAC[35]	128, 192, 256	Key transport (encryption with message authentication)

[27] ECDSA – Elliptic Curve Digital Signature Algorithm
[28] HMAC – Keyed-Hash Message Authentication Code
[29] KAS-SSC – Key Agreement Scheme – Shared Secret Computation
[30] FFC – Finite Field Cryptography
[31] ECC CDH – Elliptic Curve Cryptography Cofactor Diffie-Hellman
[32] KTS – Key Transport Scheme
[33] Per FIPS 140-2 Implementation Guidance D.9, AES-GCM is an Approved key transport technique.
[34] Per FIPS 140-2 Implementation Guidance D.9, AES with CMAC is an Approved key transport technique.
[35] Per FIPS 140-2 Implementation Guidance D.9, AES with HMAC is an Approved key transport technique.

Certificate Number	Algorithm	Standard	Modes / Methods	Key Lengths / Curves / Moduli	Use
		NIST SP 800-38F	AES key wrap	128, 192, 256	Key transport
A1392	RSA[36]	FIPS PUB 186-4	-	2048, 3072	Key pair generation
			ANSI X9.31	2048, 3072 (SHA2-256, SHA2-384, SHA2-512)	Digital signature generation
				1024, 2048, 3072 (SHA-1, SHA2-256, SHA2-384, SHA2-512)	Digital signature verification
			PKCS[37]#1 v1.5, PSS[38]	2048, 3072, 4096 (SHA2-224, SHA2-256, SHA2-384, SHA2-512)	Digital signature generation
				1024, 2048, 3072 (SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512)	Digital signature verification
A1392	SHA	FIPS PUB 202	SHA3-256	-	Message digest
A1392	SHS[39]	FIPS PUB 180-4	SHA-1-96, SHA-1, SHA-224, SHA2-256, SHA2-384, SHA2-512	-	Message digest

The vendor affirms the following cryptographic security methods:

- Cryptographic key generation – As per *NIST SP 800-133*, the module uses its FIPS-Approved DRBG to generate cryptographic keys. The resulting symmetric key or generated seed is an unmodified output from the DRBG. The module requests 256 bits of entropy from the calling application per request (the minimum number of bits of entropy loaded per request is 112 bits). The calling application and its entropy sources are located within the operational environment inside the module's physical boundary but outside the logical boundary. There is no assurance of the minimum strength of the generated keys.

The module implements the non-Approved but allowed algorithms shown in Table 6 below.

Table 6 – Allowed Algorithms

Algorithm	Caveat	Use
RSA	Key establishment methodology provides between 112 and 256 bits of encryption strength	Key transport[40]

[36] RSA – Rivest Shamir Adleman
[37] PKCS – Public Key Cryptography Standard
[38] PSS – Probabilistic Signature Scheme
[39] SHS – Secure Hash Standard
[40] Per FIPS 140-2 Implementation Guidance D.9, RSA key transport is an allowed key transport technique when using the PKCS #1 v1.5 padding scheme and a modulus of at least 2048 bits.

Algorithm	Caveat	Use
SHA-1 (Cert. A1392)	-	Digital signature generation in TLS[41]

The module employs the following non-Approved/non-compliant algorithms (use of these algorithms is restricted to the module's non-Approved mode of operation):

- ARIA
- Blake2
- Blowfish
- Camellia
- CAST, CAST5
- ChaCha20
- DES[42]
- DH (non-compliant with key sizes below 2048 bits)
- DSA[43] (non-compliant with key sizes below the minimums Approved for FIPS mode)
- ECDH (non-compliant with curves P-192, K-163, B-163, and non-NIST curves)
- EdDSA[44]
- ECDSA[45] (non-compliant with curves P-192, K-163, B-163, and non-NIST curves)
- MD2[46], MD4, MD5
- Poly1305
- RC2, RC4
- RMD160
- RSA (non-compliant with key sizes below the minimums Approved for FIPS mode)
- TLS 1.3 KDF
- Triple DES[47] (non-compliant)
- SEED
- SM2, SM3, SM3

2.3 Module Interfaces

The module supports the following four logical interfaces:

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

[41] Per NIST SP 800-52, SHA-1 is an allowed hashing technique for generating digital signatures within the TLS protocol.
[42] DES – Data Encryption Scheme
[43] DSA – Digital Signature Algorithm
[44] EdDSA – Edwards-curve Digital Signature Algorithm
[45] ECDSA – Elliptic Curve Digital Signature Algorithm
[46] MD – Message Digest
[47] DES – Data Encryption Standard

Physically, the module features the physical ports of the host platform. However, as a software-hybrid cryptographic module, the module's interfaces consist solely of the APIs offered by the wrapper. Further, in accordance with section 1.9 of the *FIPS 140-2 Implementation Guidance*, the module's control input and status output interfaces are directed only through the module's software component.

A mapping of the FIPS-defined interfaces, the host platform's physical ports/interfaces, and the module's logical interfaces can be found in Table 7.

Table 7 – FIPS 140-2 Logical Interface Mappings

FIPS 140-2 Interface	Physical Port/Interface	Logical Interface
Data Input	Physical ports of the tested platforms	API input arguments that provide input data for processing
Data Output	Physical ports of the tested platforms	API output arguments that return generated or processed data back to the caller
Control Input	Physical ports of the tested platforms	API input arguments that are used to initialize and control the operation of the module
Status Output	Physical ports of the tested platforms	API call return values
Power	Physical ports of the tested platforms	N/A

2.4 Roles, Services, and Authentication

The sections below describe the module's authorized roles, services, and operator authentication methods.

2.4.1 Authorized Roles

There are two authorized roles that module operators may assume: Crypto Officer (CO) role and a User role. 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 its only operator.

The module does not support multiple concurrent operators.

2.4.2 Operator Services

Descriptions of the services available are provided in Table 8 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 8 – Mapping of Operator Services to Inputs, Outputs, CSPs, and Type of Access

Service	Operator		Description	Input	Output	CSP and Type of Access
	CO	User				
Show status	✓	✓	Return FIPS mode status	API call parameters	Status	None
Perform on-demand self-tests	✓	✓	Perform power-up self-tests	API call parameters	Status	None
Zeroize	✓	✓	Zeroize and de-allocates memory containing sensitive data	Restart calling application; reboot or power-cycle host platform	None	All CSPs – W
Perform symmetric encryption	✓	✓	Encrypt plaintext data	API call parameters, key, plaintext	Status, ciphertext	AES key – RX AES XTS key – RX
Perform symmetric decryption	✓	✓	Decrypt ciphertext data	API call parameters, key, ciphertext	Status, plaintext	AES key – RX AES XTS key – RX
Generate symmetric digest	✓	✓	Generate symmetric digest	API call parameters, key, plaintext	Status, digest	AES CMAC key – RX AES GMAC key – RX
Verify symmetric digest	✓	✓	Verify symmetric digest	API call parameters, digest	Status	AES CMAC key – RX AES GMAC key – RX
Perform authenticated symmetric encryption	✓	✓	Encrypt plaintext using supplied AES GCM key and IV	API call parameters, key, plaintext	Status, ciphertext	AES GCM key – RX AES GCM IV – RX
Perform authenticated symmetric decryption	✓	✓	Decrypt ciphertext using supplied AES GCM key and IV	API call parameters, key, ciphertext	Status, plaintext	AES GCM key – RX AES GCM IV – RX
Generate random number	✓	✓	Return random bits to the calling application	API call parameters	Status, random bits	DRBG seed – WRX DRBG entropy input – RX DRBG 'V' value – WRX DRBG 'Key' value – WRX
Perform keyed hash operations	✓	✓	Compute a message authentication code	API call parameters, key, message	Status, MAC	HMAC key – RX
Perform hash operation	✓	✓	Compute a SHA message digest	API call parameters, message	Status, hash	None
Generate DSA domain parameters	✓	✓	Generate DSA domain parameters	API call parameters	Status, domain parameters	None
Verify DSA domain parameters	✓	✓	Verify DSA domain parameters	API call parameters	Status, domain parameters	None
Generate asymmetric key pair	✓	✓	Generate a public/private key pair	API call parameters	Status, key pair	DSA public key – W DSA private key – W ECDSA public key – W ECDSA private key – W RSA public key – W RSA private key – W
Verify ECDSA public key	✓	✓	Verify an ECDSA public key	API call parameters, key	Status	ECDSA public key – R

Service	Operator		Description	Input	Output	CSP and Type of Access
	CO	User				
Generate digital signature	✓	✓	Generate a digital signature	API call parameters, key, message	Status, signature	DSA private key – RX ECDSA private key – RX RSA private key – RX
Verify digital signature	✓	✓	Verify a digital signature	API call parameters, key, signature, message	Status	DSA public key – RX ECDSA public key – RX RSA public key – RX
Perform key wrap	✓	✓	Perform AES key wrap	API call parameters, encryption key, key	Status, encrypted key	AES GCM key – RX AES GCM IV – RX
Perform key unwrap	✓	✓	Perform AES key unwrap	API call parameters, decryption key, encrypted key	Status, decrypted key	AES GCM key – RX AES GCM IV – RX
Perform key encapsulation	✓	✓	Perform RSA key encapsulation	API call parameters, encryption key, key	Status, encrypted key	RSA public key – RX
Perform key un-encapsulation	✓	✓	Perform key RSA un-encapsulation	API call parameters, decryption key, encrypted key	Status, decrypted key	RSA private key – RX
Compute shared secret	✓	✓	Compute DH/ECDH shared secret	API call parameters	Status, shared secret	DH public component – RX DH private component – RX ECDH public component – RX ECDH private component – RX TLS pre-master secret – W
Derive TLS keys	✓	✓	Derive TLS 1.2 session and integrity keys	API call parameters, pre-master secret	Status, TLS keys	TLS pre-master secret – RX TLS master secret – WX AES key – W AES GCM key – W HMAC key – W

Table 9 below lists the services available exclusively in the non-Approved mode of operation.

Table 9 – Non-Approved Services

Service	Operator		Security Function(s)
	CO	User	
Perform symmetric data encryption (non-compliant)	✓	✓	ARIA, Blowfish, Camellia, CAST, CAST5, ChaCha20, DES, RC2, RC4, SEED, SM4, Triple DES (non-compliant)
Perform symmetric data decryption (non-compliant)	✓	✓	ARIA, Blowfish, Camellia, CAST, CAST5, ChaCha20, DES, RC2, RC4, SEED, SM4, Triple DES (non-compliant)
Perform MAC operations (non-compliant)	✓	✓	CMAC with Triple DES, Poly1305
Perform keyed hash operations (non-compliant)	✓	✓	HMAC MD5
Perform hash operation (non-compliant)	✓	✓	Blake2, MD2, MD4, MD5, RMD160, SM3

Service	Operator		Security Function(s)
	CO	User	
Perform digital signature functions (non-compliant)	✓	✓	DSA (non-compliant), ECDSA (non-compliant), EdDSA, RSA (non-compliant), SM2
Perform key agreement functions (non-compliant)	✓	✓	DH (non-compliant), ECDH (non-compliant)
Derive TLS keys	✓	✓	TLS 1.3 KDF (non-compliant)

2.4.3 Authentication

The module does not support authentication mechanisms; all operator roles are assumed implicitly.

2.5 Physical Security

The Qumulo Secure cryptographic module is a software-hybrid module that operates on a multi-chip standalone platform which conforms to the level 1 requirements for physical security. All disjoint components of the module are entirely contained within the production-grade enclosure of the host platform, which blocks physical access to the module.

2.6 Operational Environment

All cryptographic keys and CSPs are under the control of the host platform's operating system, which protects the module's CSPs against unauthorized disclosure, modification, and substitution. The process and memory management functionality of host platform's operating system prevents unauthorized access to plaintext private and secret keys, CSPs, and intermediate key generation values by external processes during module execution. The module only allows access to CSPs through its well-defined API. Processes that are forked or spawned by the module are owned by the module and are not owned by external processes.

2.7 Cryptographic Key Management

The module supports the secret/private cryptographic keys, key components, CSPs, and public keys listed below in Table 10 and Table 11.

Table 10 – Secret/Private Keys, Key Components, and CSPs

CSP	CSP Type	Generation / Input	Output	Storage	Zeroization	Use
AES key	128/192/256-bit AES key (CBC, CCM, CFB, CTR, ECB, OFB, KW, KWP modes)	Input in plaintext via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Unload module; Remove power	Symmetric encryption, decryption
AES GCM key	128/192/256-bit AES key	Input in plaintext via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Unload module; Remove power	Symmetric encryption, decryption
AES GCM IV	96-bit value	Generated internally at its entirety randomly[48]	Never output from the module	Keys are not persistently stored by the module	Unload module; Remove power	Initialization vector for AES GCM
AES XTS key	256-bit AES-XTS key	Input in plaintext via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Unload module; Remove power	Symmetric encryption, decryption
AES CMAC key	128/192/256-bit AES key	Input in plaintext via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Unload module; Remove power	MAC generation, verification
AES GMAC key	128/192/256-bit AES key	Input in plaintext via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Unload module; Remove power	MAC generation, verification
HMAC key	160-bit (minimum) HMAC key	Input in plaintext via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Unload module; Remove power	Keyed hash
DSA private key	2048/3072-bit DSA private key	Generated internally via Approved DRBG OR Input in plaintext via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Unload module; Remove power	Signature generation
ECDSA private key	ECDSA private key (all FIPS-Approved curves)	Generated internally via Approved DRBG OR Input in plaintext via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Unload module; Remove power	Signature generation

[48] The IV generation method complies with technique #2 in FIPS 140-2 IG A.5. This method follows the IV construction method described in section 8.2.2 of *NIST SP 800-38D*.

CSP	CSP Type	Generation / Input	Output	Storage	Zeroization	Use
RSA private key	2048/3072/4096-bit RSA private key	Generated internally via Approved DRBG OR Input in plaintext via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Unload module; Remove power	Signature generation; decryption
DH private component	2048-bit	Generated internally via Approved DRBG OR Input in plaintext via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Unload module; Remove power	DH shared secret generation
ECDH private component	All FIPS-Approved P/B/K-curves	Generated internally via Approved DRBG OR Input in plaintext via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Unload module; Remove power	ECDH shared secret generation
TLS pre-master secret	384-bit random value	Input in plaintext via API call parameter	Never output from the module	Keys are not persistently stored by the module	Unload module; Remove power	Derivation of the TLS master secret
TLS master secret	384-bit shared secret	Derived internally using the TLS pre-master secret via TLS KDF	Never output from the module	Keys are not persistently stored by the module	Unload module; Remove power	Derivation of the TLS session key and TLS authentication key
DRBG entropy input	256-bit value	Generated externally[49] and input in plaintext via API call parameter	Never output from the module	Keys are not persistently stored by the module	Unload module; Remove power	Entropy material for DRBG
DRBG seed	Random data – 440 or 880 bits	Generated internally using nonce along with DRBG entropy input	Never output from the module	Keys are not persistently stored by the module	Unload module; Remove power	Seeding material for DRBG
DRBG 'V' value	Internal DRBG state value	Generated internally	Never output from the module	Keys are not persistently stored by the module	Unload module; Remove power	Used for DRBG

[49] The module employs a non-deterministic random number generator which is outside of the logical cryptographic boundary.

CSP	CSP Type	Generation / Input	Output	Storage	Zeroization	Use
DRBG 'Key' value	Internal DRBG state value	Generated internally	Never output from the module	Keys are not persistently stored by the module	Unload module; Remove power	Used for DRBG

Table 11 – Public Keys

CSP	CSP Type	Generation / Input	Output	Storage	Zeroization	Use
DSA public key	2048/3072-bit DSA public key	Generated internally via Approved DRBG OR Input in plaintext via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Unload module; Remove power	Signature verification
ECDSA public key	ECDSA public key (all FIPS-Approved curves)	Generated internally via Approved DRBG OR Input in plaintext via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Unload module; Remove power	Signature verification
RSA public key	1024/2048/3072/4096-bit RSA public key	Generated internally via Approved DRBG OR Input in plaintext via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Unload module; Remove power	Signature verification; encryption *1024-bit keys for signature verification only*
DH public component	2048-bit DH public component	Generated internally via Approved DRBG OR Input in plaintext via API call parameter	Output in plaintext	Keys are not persistently stored by the module	Unload module; Remove power	DH shared secret generation
ECDH public component	All FIPS-Approved P/B/K-curves	Generated internally via Approved DRBG OR	Output in plaintext	Keys are not persistently stored by the module	Unload module; Remove power	ECDH shared secret generation

CSP	CSP Type	Generation / Input	Output	Storage	Zeroization	Use
		Input in plaintext via API call parameter				

The module is compatible with TLS 1.2 and supports the AES-GCM cipher suites from section 3.3.1 of *NIST SP 800-52rev2*. The AES-GCM IV is generated as specified in *RFC 5288*. This generation method follows section A.5 (technique #1) of *the Implementation Guidance for FIPS PUB 140-2 and the CMVP*. This IV shall only be used in the context of the AES-GCM mode encryption within the TLS 1.2 protocol.

2.8 EMI / EMC

The Qumulo Secure was tested on the platform listed in Section 2.2 above. This platform was 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

Self-tests are performed by the module when the module is first powered up and initialized, as well as during module operation when certain conditions exist. The following sections list the 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 test (using an HMAC SHA-256 digest)
- Cryptographic algorithm implementation tests
    - libqumulo_secure.so
        - AES-XTS encrypt and decrypt KATs (256 bits)
    - libcrypto.so
        - AES-CCM encrypt and decrypt KATs[50] (128-bit)
        - AES-KW encrypt and decrypt KATs (256-bit)
        - DRBG KAT
        - DSA PCT[51] (2048-bit)
        - ECDSA PCT (P-256 curve)
        - RSA sign and verify KATs (2048-bit, X9.31 scheme)
        - SHA KATs (SHA-1, SHA2-512, SHA3-256)
        - FFC Primitive “Z” Computation KAT (2048-bit)
        - ECC Primitive “Z” Computation KAT (P-256 curve)

Per FIPS 140-2 IG 9.3, since the module employs an HMAC SHA2-256 for the integrity check, explicit KATs for the HMAC and SHA2-256 implementations are not required.

Per FIPS 140-2 IG 9.4, since the testing of the AES-KW and AES-ECB implementations in libcrypto.so tests both authenticated encryption functions as well as forward and inverse cipher functions, explicit KATs for the other supported modes of AES are not required. Further, since the implementations for SHA-1, SHA2-256, and SHA2-512 are covered by existing power-up KATs, explicit KATs for SHA2-224 and SHA2-384 are not required.

[50] KAT – Known Answer Test
[51] PCT – Pairwise Consistency Test

2.9.2 Conditional Self-Tests

The module performs the following conditional self-tests in libcrypto.so:

- DSA PCT
- ECDSA PCT
- RSA PCT

2.9.3 Critical Functions Self-Tests

The module performs health checks for the DRBG's Generate, Instantiate, and Reseed functions as specified in section 11.3 of *NIST SP 800-90Arev1*. These tests are performed as power-up tests. The module also performs all applicable key assurances for its DH and ECDH implementations as specified in section 9 of *NIST SP 800-56Arev3*. These tests are performed as conditional tests.

Additionally, the module performs the following conditional critical functions test(s) in libqumulo_secure.so:

- XTS AES duplicate key test

2.9.4 Self-Test Failure Handling

Upon failure of a power-up or conditional self-test, the module will enter a critical error state. In this state, the module will immediately terminate the calling application. Once terminated, the module's data output interfaces and cryptographic services are no longer available for use.

To recover, the module must be reinitialized by restarting the calling application or via reboot/power-cycle of the host platform. If the power-up self-tests complete successfully, then the module can resume normal operations. If the module continues to experience self-test failures after reinitializing, then the module will not be able to resume normal operations, and the CO should contact Qumulo, Inc. for assistance.

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.

3. Secure Operation

The sections below describe how to ensure the module is operating in its validated configuration. **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**.

3.1 Module Setup

The following paragraphs describe the steps necessary to ensure that the Qumulo Secure is running in its validated configuration.

Please note that the Qumulo Secure cryptographic module is not delivered to end-users as a standalone offering. Rather, it is an integrated component of the Qumulo Core solution. Qumulo does not provide end-users with any mechanisms to directly access the module, its APIs, or any information sent to/from Qumulo software.

3.1.1 Installation

As the module is an integrated component of the Qumulo Core software, module operators have no ability to independently load the module onto the target platform. The module and its calling application are to be installed on a platform specified in section 2.2 or one where portability is maintained. Qumulo does not provide any mechanisms to directly access the module, its APIs, or any information sent to/from the Qumulo Core software.

3.1.2 Initialization

This module is designed to support Qumulo applications, and these applications are the sole consumers of the cryptographic services provided by the module. No end-user action is required to initialize the module for operation; the calling applications perform all actions required to initialization the module.

The calling application handles the initialization tasks as described below:

- The cryptographic module invokes the `getentropy`() function to obtain entropy for random number generation, and then passively receives entropy from the calling application while exercising no control over the amount or the quality of the obtained entropy.

  To provide support for FIPS operation, Qumulo developers design their calling applications to be invoked using the LD_PRELOAD environment variable with a specified library. The LD_PRELOAD environment variable can be used to change symbol dependencies from a system default location to another specified location. Using this method, the calling application can override the default `getentropy()` call and instead reference a library with a `getentropy()` function of Qumulo's own choosing and/or design.

  For operation in FIPS mode, Qumulo developers shall use the LD_PRELOAD environment variable to specify their own shared library that includes a function called `getentropy()` that does not leverage the Linux system entropy; here, when the module requests entropy, it must rely on the source(s) selected by

the calling application. **Failure to use an appropriate entropy source will result in non-compliant behavior and is outside the scope of this Security Policy.**

- The calling application loads the module into memory for execution. Upon loading, the module's power-up integrity test and cryptographic algorithm self-tests are performed automatically via a default entry point (DEP) implemented in the module. These tests are performed prior to the OS passing process control to the calling application and without any specific action from the calling application or the end-user. End-users have no means to short-circuit or bypass these actions. Failure of any of these tests will result in a failure of the module to load for execution.

3.1.3 Configuration

No configuration steps are required to be performed by end-users.

3.2 Operator Guidance

The following sections provide guidance to module operators for ensuring that the module is operating in its FIPS-Approved configuration.

3.2.1 Crypto Officer Guidance

There are no specific management activities required of the CO role to ensure that the module runs securely. However, if any irregular activity is noticed or the module is consistently reporting errors, then Qumulo Customer Support should be contacted.

3.2.2 User Guidance

Although the User role provides no ability to modify the configuration of the module, end-users should notify the CO if any irregular activity is observed.

3.2.3 General Operator Policies and Guidance

The following list provides additional policies for the general operation of the module:

- The `qumulo_secure_in_fips_approved_mode()` API can be used to determine the module's operational status. A non-zero return value indicates that the module has passed all power-up self-tests and is currently in its FIPS-Approved mode.

- To verify the module version, module operators can examine the hex value of the module's HMAC SHA digest. Version 1.0 of the module has the following hex value:

  ```
  2a9e 878b 8192 8f50 d08d dda1 e359 ad61fb3c 3b95 362c 3f83 12ff f850 f830 0351
  ```

  Please refer to the appropriate article under the "Qumulo Administration" category of the Qumulo Core Knowledge Base for information on displaying the module digest.

- The module's power-up self-tests can be run on-demand by restarting the calling application or via reboot/power-cycle of the host platform.

- As a software-hybrid cryptographic module, there is no mechanism within the module boundary for the persistent storage of keys and CSPs. Key management, including protection and zeroization, of any keys and CSPs that exist outside the module's logical boundary is the responsibility of the end-user. For the zeroization of keys in volatile memory, module operators can unload the module from memory or reboot/power-cycle the host platform.

3.3 Additional Guidance and Usage Policies

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

- The cryptographic module's services are designed 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 is responsible for the storage and zeroization of keys and CSPs passed into and out of the module.

- The replacement `getentropy()` function implemented in Qumulo's calling applications shall reference entropy sources that meet the minimum security strength of 112 bits required for the CTR_DRBG as shown in *NIST SP 800-90Arev1*, Table 3.

- 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.

4. Acronyms and Abbreviations

Table 12 provides definitions for the acronyms and abbreviations used in this document.

Table 12 – Acronyms and Abbreviations

Acronym	Definition
AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
API	Application Programming Interface
AVX	Advanced Vector Extensions
CBC	Cipher Block Chaining
CCCS	Canadian Centre for Cyber Security
CCM	Counter with CBC-MAC
CFB	Cipher Feedback
CKG	Cryptographic Key Generation
CMAC	Cipher-Based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CO	Cryptographic Officer
CPU	Central Processing Unit
CS	Ciphertext Stealing
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
DSA	Digital Signature Algorithm
ECB	Electronic Code Book
ECC CDH	Elliptic Curve Cryptography Cofactor Diffie-Hellman
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EdDSA	Edwards-curve Digital Signature Algorithm
EMI/EMC	Electromagnetic Interference / Electromagnetic Compatibility
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standard

Acronym	Definition
GB	Gigabyte
GCM	Galois/Counter Mode
GPC	General-Purpose Computer
HDD	Hard Disk Drive
HMAC	Keyed-Hash Message Authentication Code
KAS-SSC	Key Agreement Scheme – Shared Secret Computation
KAT	Known Answer Test
KDF	Key Derivation Function
KTS	Key Transport Scheme
KW	Key Wrap
MD5	Message Digest 5
NFS	Network File System
NIST	National Institute of Standards and Technology
OFB	Output Feedback
OS	Operating System
PAA	Processor Algorithm Acceleration
PCT	Pairwise Consistency Test
PKCS	Public Key Cryptography Standard
PSS	Probabilistic Signature Scheme
PUB	Publication
REST	Representational State Transfer
RSA	Rivest, Shamir, and Adleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SMB	Server Message Block
SP	Special Publication
SSD	Solid-State Drive
TLS	Transport Layer Security
XEX	XOR Encrypt XOR
XOR	Exclusive OR
XTS	XEX-Based Tweaked-Codebook Mode with Ciphertext Stealing

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