IBM Cloud Object Storage System's™

FIPS Cryptographic Module v1.1

Version 20170317D, 17 March 2017

FIPS 140-2 Security Policy

Contents

Document Information

Copyright Notice

Intended Purpose and Audience

This security guide is designed to instruct developers on how to create a version of the IBM Cloud Object Store uses an encryption module that is compliant to FIPS 140-2 Level 1 for a software module.

Acknowledgments

IBM serves as the "vendor" for this validation. Project management coordination and technical work for this effort was provided by:

Mark Seaborn
222 S Riverside Plz Ste 1700
Chicago, IL 60606 USA
+1 312 423 6640 x2354
mseaborn@us.ibm.com

with additional technical work by:

Jason Resch
222 S Riverside Plz Ste 1700
Chicago, IL 60606 USA
jresch@us.ibm.com

Trent Johnson
222 S Riverside Plz Ste 1700
Chicago, IL 60606 USA
trejo@us.ibm.com

Dusty Hendrickson
222 S Riverside Plz Ste 1700
Chicago, IL 60606 USA
dhendrickson@us.ibm.com

CygnaCom Laboratories performed the validation testing. For information on validation or revalidations of software, contact:

Nithya Rachamadugu Director
Security Evaluation Laboratory (SEL)
Cryptographic Equipment Assessment Laboratory (CEAL)
Cygnacom Solutions
7925 Jones Branch Drive, Suite 5400
McLean, VA 22102
703-270-3551 tel
nithya@cygnacom.com

IBM wishes to acknowledge the OpenSSL Software Foundation for the exemplary work done in developing OpenSSL, the FIPS 140-2 cryptographic module and the Security Policy upon which this document was modeled. All referenced resources from the OpenSSL Foundation can be found at https://www.openssl.org.

Revision History

Date	Modification
2015-12-11	Created for v1.1 of Cleversafe FIPS Cryptographic Module
2017-03-17	Added new platforms to the certification and rebranded to IBM as a part of the acquisition.

References

Reference	Full Specification Name
ANSI X9.31	*Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)*
FIPS 140-2	*Security Requirements for Cryptographic modules, May 25, 2001*
FIPS 180-4	http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf]
FIPS 186-4	*Digital Signature Standard*
FIPS 197	*Advanced Encryption Standard*
FIPS 198-1	*The Keyed-Hash Message Authentication Code (HMAC)*
SP 800-38B	*Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*
SP 800-38C	*Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*
SP 800-38D	*Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*
SP 800-56A	*Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*
SP 800-67R1	*Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*
SP 800-89	*Recommendation for Obtaining Assurances for Digital Signature Applications*
SP 800-90	*Recommendation for Random Number Generation Using Deterministic Random Bit Generators*
SP 800-90Ar1	http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf]
SP 800-131A	*Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths*

Chapter 1 Introduction

This document is the non-proprietary security policy for the IBM Object Store FIPS Cryptographic Module, hereafter referred to as the Module.

The Module is a software library providing a C-language API for use by other processes that require cryptographic functionality. The Module is classified by FIPS 140-2 as a software module, multi-chip standalone module embodiment. The physical cryptographic boundary is the physical perimeter of the general-purpose computer on which the Module is installed. The logical cryptographic boundary of the Module is the fipscanister object module, a single object module file named fipscanister.o. The Module only communicates with the calling application (the process that invokes the Module services).

The FIPS 140-2 security levels for the Module are as follows:

Table 1. Security Level of Security Requirements

Security Requirement	Security Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services, and Authentication	2
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	3
Mitigation of Other Attacks	N/A

The Module's software version for this validation is 1.1. The Module's block diagram can be found in **Figure 1: Module Block Diagram**. The diagram depicts the Module's cryptographic boundary as a heavy red boarder labeled *Logical Boundary (Object Module)

The module software is a direct reuse of the FIPS Object Module v2.0.10 from the OpenSSL Software Foundation. The Module contains an implementation of Dual_EC_DRBG. The Dual_EC_DRBG algorithm shall not be used in the FIPS Approved mode of operation v2.0.10.

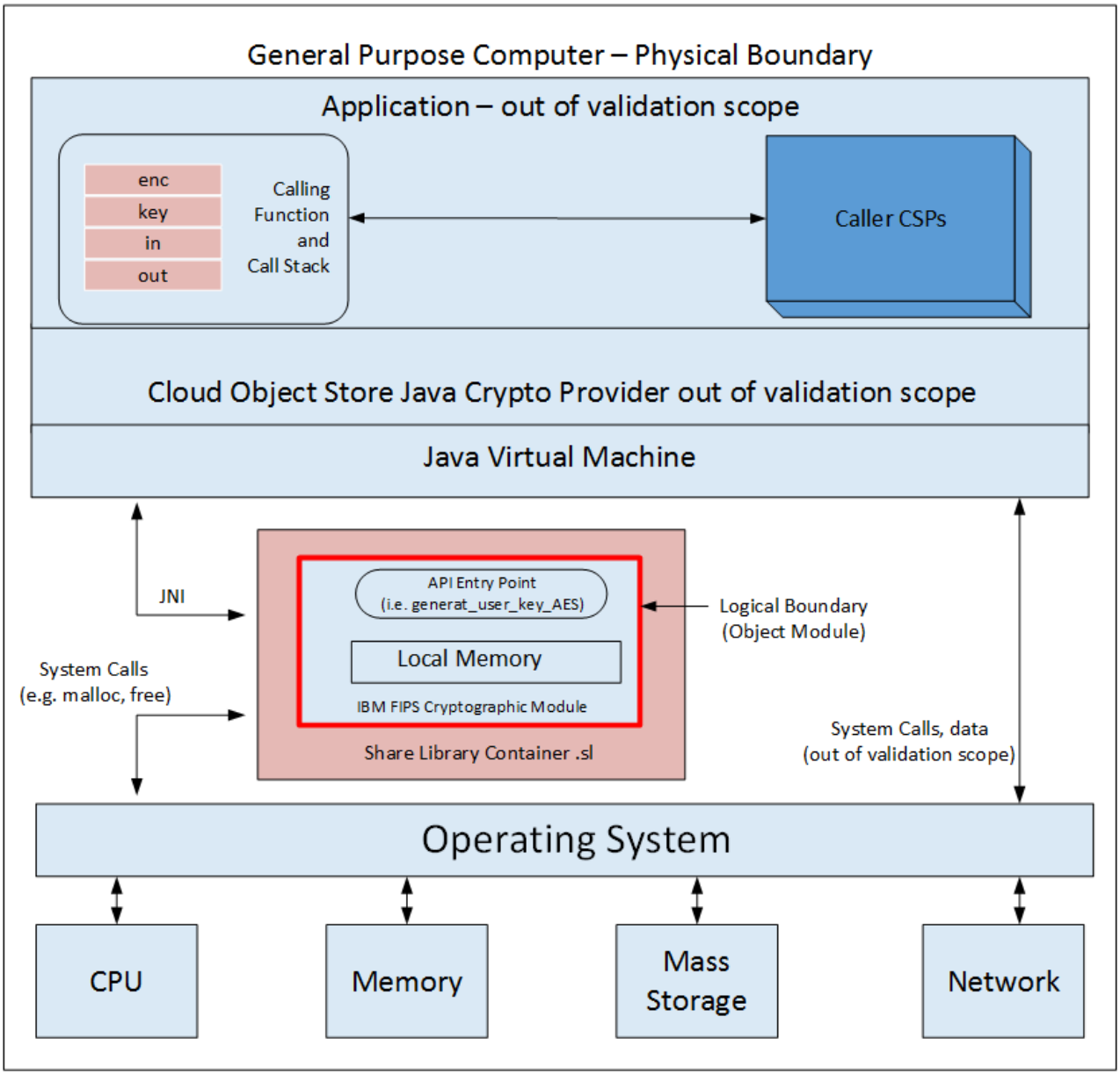

Figure 1: Module Block Diagram

Chapter 2 . Tested Configurations

The Elliptic Curve Support column indicates support for all NIST-defined B, K, and P curves (BKP).

Table 2. Tested Configurations

#	Operational Environment	Processor	Optimizations (Target	Elliptic Curve Support
1	ClevOS 3.8-FIPS-Edition	Intel Xeon	AES	BKP
2	ClevOS 3.8-FIPS-Edition	Intel Xeon	AES-NI	BKP
3	ClevOS 3.8.2.19-FIPS-Edition	Intel Xeon	AES	BKP
4	ClevOS 3.8.2.19-FIPS-Edition	Intel Xeon	AES-NI	BKP

ClevOS3.8-FIPS-Edition and ClevOS 3.8.2.19-FIPS-Edition will be referred to as ClevOS FIPS-Edition for the rest of this document.

Chapter 3 Ports and Interfaces

The physical ports of the Module are the same as the computer system on which it executes. The logical interface is a C API.

Table 3. Logical interfaces

Logical interface type	Description
Control input	API entry point and corresponding stack parameters
Data input	API entry point data input stack parameters
Status output	API entry point return values and status stack parameters
Data output	API entry point data output stack parameters

As a software module, the control of the physical ports is outside the Module scope. However, when the Module is performing self-tests, or is in an error state, all output on the logical data output interface is inhibited. The Module is single-threaded and in error scenarios returns only an error value (no data output is returned).

Chapter 4 Modes of Operation and Cryptographic Functionality

The Module supports only a FIPS 140-2 Approved mode. ***Tables 4*** and ***5*** list the *Approved* and *Non-approved but Allowed* algorithms, respectively.

Table 4. FIPS Approved Cryptographic Functions

Function	Algorithm	Options	Cert #
Random Number Generation; Symmetric Key Generation	[SP 800-90] DRBG Prediction resistance supported for all variations (For all DRBGs the "supported security strengths" is just the highest supported security strength per SP800-90Arev1 and *SP800-57)	Hash DRBG HMAC DRBG, no reseed CTR DRBG (AES), no derivation function	941, 942, 1428
Encryption, Decryption and CMAC	[SP 800-67]	3-Key TDES TECB, TCBC, TCFB, TOFB; CMAC generate and verify	2011, 2012, 2380
	[FIPS 197] AES [SP 800-38B] CMAC [SP 800-38C] CCM [SP 800-38D] GCM [SP 800-38E] XTS	128/192/256 ECB, CBC, OFB, CFB 1, CFB 8, CFB 128, CTR, XTS; CCM; GCM; CMAC generate and verify	3611, 3612, 4422
Message Digests	[FIPS 180-4]	SHA-1, SHA-2 (224, 256, 384, 512)	2984, 2985, 3640
Keyed Hash	[FIPS 198] HMAC	SHA-1, SHA-2 (224, 256, 384, 512)	2318, 2319, 2935
Digital Signature and Asymmetric Key Generation	[FIPS 186-2] RSA	GenKey9.31, SigGen9.31, SigGenPKCS1.5, SigGenPSS, SigVer9.31, SigVerPKCS1.5, SigVerPSS (2048/3072/4096 with all SHA-2 sizes)	1858, 1859, 2409
	[FIPS 186-4] DSA	PQG Gen, PQG Ver, Key Pair Gen, Sig Gen, Sig Ver (1024-/2048- /3072-with all SHA-2 sizes)	1006, 1007, 1186
	[FIPS 186-4] ECDSA	PKG: CURVES (P-224, P-256, P-384, P-521, K-224, K-256, K-384, K- 521, B-224, B-256, B-384, B-521, ExtraRandomBits,Testing Candidates) PKV: CURVES (ALL-P, ALL-K, ALL-B) SigGen: CURVES (P-224: (SHA-224, 256, 384, 512) P-256: (SHA-224, 256, 384, 512) P-384: (SHA-224, 256, 384, 512) P-521: (SHA-224, 256, 384, 512) K-233: (SHA-224, 256, 384, 512) K-283: (SHA-224, 256, 384, 512) K-409: (SHA-224, 256, 384, 512) K-571: (SHA-224, 256, 384, 512) B-233: (SHA-224, 256, 384, 512) B-283: (SHA-224, 256, 384, 512) B-409: (SHA-224, 256, 384, 512) B-571: (SHA-224, 256, 384, 512)) SigVer: CURVES (P-192: (SHA-1, 224, 256, 384, 512) P-224: (SHA-1, 224, 256, 384, 512) P-256: (SHA-1, 224, 256, 384, 512) P-384: (SHA-1, 224, 256, 384, 512) P-521: (SHA-1, 224, 256, 384, 512) K-163: (SHA-1, 224, 256, 384, 512)	743, 744, 1071

Function	Algorithm	Options	Cert #
		K-233: (SHA-1, 224, 256, 384, 512) K-283: (SHA-1, 224, 256, 384, 512) K-409: (SHA-1, 224, 256, 384, 512) K-571: (SHA-1, 224, 256, 384, 512) B-163: (SHA-1, 224, 256, 384, 512) B-233: (SHA-1, 224, 256, 384, 512) B-283: (SHA-1, 224, 256, 384, 512) B-409: (SHA-1, 224, 256, 384, 512) B-571: (SHA-1, 224, 256, 384, 512))	
ECC CDH (KAS)	[SP 800-56A] (§5.7.1.2)	All NIST-defined B, K and P curves except sizes 163 and 192	630, 631, 1137

The Module supports only NIST-defined curves for use with ECDSA and ECC CDH. IBM's tested configuration uses all NIST-defined curves and hence all columns are marked with **BKP** in ***Table 2***.

Table 5. Non-FIPS Approved but Allowed Cryptographic Functions

Category	Algorithm	Description
Key Agreement	EC DH	Non-compliant (untested) DH scheme using elliptic curve, supporting all NIST-defined B, K and P curves. Key agreement is a service provided for calling process use, but is not used to establish keys into the Module.
Key Encryption, Decryption	RSA	The RSA algorithm may be used by the calling application for encryption or decryption of keys. No claim is made for SP 800-56B compliance, and no CSPs are established into or exported out of the Module using these services.

The Module implements the following services which are Non-Approved per the SP 800-131A transition:

Table 6. FIPS Non-Approved Cryptographic Functions

Function	Algorithm	Options
Random Number Generation; Symmetric Key Generation	[ANSI X9.31] RNG	AES 128/192/256
	[SP 800-90] DRBG	Dual_EC_DRBG
Digital Signature and Asymmetric Key Generation	[FIPS 186-2] RSA	GenKey9.31, SigGen9.31, SigGenPKCS1.5, SigGenPSS (1024/1536 with all SHA sizes,
	[FIPS 186-2] DSA	PQG Gen, Key Pair Gen, Sig Gen (1024 with all SHA sizes, 2048/3072 with SHA-1)
	[FIPS 186-4] DSA	PQG Gen, Key Pair Gen, Sig Gen (1024 with all SHA sizes, 2048/3072 with SHA-1)
	[FIPS 186-2] ECDSA	PKG: CURVES (P-192 K-163 B-163) SIG(gen): CURVES (P-192, P-224, P-256, P-384, P-521, K-163, K-233,
	[FIPS 186-4] ECDSA	PKG: CURVES (P-192 K-163 B-163) SigGen: CURVES (P-192: (SHA-1, 224, 256, 384, 512); P-224: (SHA- 1); P-256: (SHA-1); P-384: (SHA-1) P-521: (SHA-1); K-163: (SHA-1, 224, 256, 384, 512); K-233: (SHA-1); K-283: (SHA-1); K-409: (SHA-1); K-571: (SHA-1); B-163: (SHA-1, 224, 256, 384, 512); B-233: (SHA-1); B-283: (SHA-1); B-409: (SHA-1); B-571: (SHA-1))

ECC CDH (CVL)	[SP 800-56A] (§5.7.1.2)	All NIST-Recommended B, K and P curves sizes 163 and 192

These algorithms shall not be used when operating in the FIPS Approved mode of operation.

EC DH Key Agreement provides a maximum of 256 bits of security strength. RSA Key Wrapping provides a maximum of 256 bits of security strength.

The Module requires an initialization sequence (see IG 9.5): the calling application invokes **FIPS_mode_set()**, which returns a **1** for success and **0** for failure. If **FIPS_mode_set()** fails, then all cryptographic services fail from then on. The application can test to see if FIPS mode has been successfully performed by validating the return value of the **FIPS_mode_set()** API call.

Note: The function call in the Module is `FIPS_module_mode_set()` which is used by *{clevOsFips}* via the `FIPS_mode_set()` wrapper function.

The Module is a cryptographic engine library, which can be used only in conjunction with IBM's operating system, *ClevOS FIPS-Edition*. Aside from the use of the NIST-defined elliptic curves as trusted third-party domain parameters, all other FIPS 186-4 assurances are outside the scope of the Module, and are the responsibility of the calling process.

Critical Security Parameters and Public Keys

All CSPs used by the Module are described in this section. All access to these CSPs by Module services is described in *Authentication and Services*. The CSP names are generic, corresponding to API parameter data structures.

Table 7. Critical Security Parameters

CSP Name	Description
RSA SGK	RSA (1024- to 16384-bit) signature generation key
RSA KDK	RSA (1024- to 16384-bit) key decryption (private key transport) key
DSA SGK	[FIPS 186-4] DSA (1024-/2048-/3072-bit) signature generation key or [FIPS 186-2] DSA (1024) signature generation key
ECDSA SGK	ECDSA (All NIST-defined B, K, and P curves) signature generation key
EC DH Private	EC DH (All NIST-defined B, K, and P curves) private key agreement key.
AES EDK	AES (128/192/256 bit) encrypt / decrypt key
AES CMAC	AES (128/192/256 bit) CMAC generate / verify key
AES GCM	AES (128/192/256 bit) encrypt / decrypt / generate / verify key
AES XTS	AES (256/512 bit) XTS encrypt / decrypt key
TDES EDK	TDES (3-Key) encrypt / decrypt key
TDES CMAC	TDES (3-Key) CMAC generate / verify key
HMAC Key	Keyed hash key (160-/224-/256-/384-/512-bit)
RNG CSPs	Seed (128 bit), AES 128-/192-/256-bit seed key and associated state variables for ANSI X9.31 AES- based RNG. There is an explicit test for equality of the seed and
Hash_DRBG CSPs	V (440/888 bit) and C (440/888 bit), entropy input (length dependent on security
HMAC_DRBG CSPs	V (160/224/256/384/512 bit) and Key (160/224/256/384/512 bit), entropy input (length dependent on security strength)
CTR_DRBG CSPs	V (128 bit) and Key (AES 128/192/256 bit), entropy input (length dependent on

CSP Name	Description
Dual_EC_DRBG CSPs	S (P-256, P-384, P-521), entropy input (length dependent on security strength)
CO-AD-Digest	Pre-calculated HMAC-SHA-1 digest used for Crypto Officer role authentication
User-AD-Digest	Pre-calculated HMAC-SHA-1 digest used for User role authentication

The Crypto Officer loads authentication data into the Module during the Module build process. Authentication data cannot be accessed otherwise.

The Module does not output intermediate key generation values.

Table 8. Public Keys

CSP Name	Description
RSA SVK	RSA (1024- to 16384-bit) signature verification public key
RSA KEK	RSA (1024- to 16384-bit) key encryption (public key transport) key
DSA SVK	FIPS 186-4 DSA (1024-/2048-/3072-bit) signature verification key or [FIPS 186-2] DSA (1024-bit) signature verification key
ECDSA SVK	ECDSA (All NIST-defined B, K and P curves) signature verification key
EC DH Public	EC DH (All NIST-defined B, K and P curves) public key agreement key.

For all CSP and Public Keys

Storage

RAM, associated to entities by memory location. The Module stores RNG and DRBG state values for the lifetime of the RNG or DRBG instance. The Module uses CSPs passed in by the calling application on the stack. The Module does not store any CSP persistently (beyond the lifetime of an API call), apart from RNG and DRBG state values used for the Module's default key generation service.

Generation

The Module implements an ANSI X9.31 compliant RNG and SP 800-90Arev1 compliant DRBG services for creation of symmetric keys, and for generation of DSA, EC and RSA keys as shown in *Table 4*. The calling application is responsible for storage of generated keys returned by the Module.

Note: As the certification for the Module will remain under "Alternative Scenario 1A revalidation," IBM cannot remove ANSI X9.31 functionalities from the implementation. IBM is aware that ANSI X9.31 is scheduled to be disallowed as of 31 December 2015 and that the functionality exists in the Module. IBM product implementations that rely on the Module use DRBG for random number generation and not ANSI X9.31.

Entry

All CSPs enter the Module's logical boundary in plain text as API parameters, associated by memory location. However, none cross the physical boundary.

Output

The Module does not output CSPs, other than as explicit results of key generation services. However, none cross the physical boundary.

Destruction

API function calls for temporarily stored CSPs automatically zero out sensitive data. The Module also provides functions to explicitly destroy CSPs related to RNG services. The calling application is responsible for parameters passed in and out of the Module.

Private and secret keys — as well as seeds and entropy input — are provided to the Module by the calling application and are destroyed when released by the appropriate API function calls. Keys residing in internally allocated data structures (during the lifetime of an API call) can only be accessed using the Module defined API. The operating system protects memory and process space from unauthorized access. Only the calling application that creates or imports keys can use or export such keys. The invoking/calling application executes all API functions in a non-overlapping sequence such that no two API functions execute concurrently. An authorized application acting as a user (Crypto Officer and User) has access to all key data generated during the operation of the Module.

In the event Module power is lost and restored, the calling application must ensure that any AES-GCM keys used for encryption or decryption are redistributed.

The calling applications shall use entropy sources that meet the security strength required for the RNG mechanism: 128 bits for the [ANSI X9.31] RNG mechanism, and as shown in [SP 800-90Arev1] ***Table 2*** (Hash_DRBG, HMAC_DRBG), ***Table 3*** (CTR_DRBG) and ***Table 4*** (Dual_EC_DRBG).

Callback functions supply entropy. Those functions must return an error if the minimum entropy strength cannot be met.

Chapter 5 Roles, Authentication and Services

The Module implements the required User and Crypto Officer roles and requires authentication for those roles. Only one role may be active at a time and the Module does not allow concurrent operators.

The User or Crypto Officer role is assumed by passing the appropriate password to the `FIPS_module_mode_set()` function. The password values may be specified at build time and must have a minimum length of 16 characters.

Any attempt to authenticate with an invalid password will result in an immediate and permanent failure condition rendering the Module unable to enter the FIPS mode of operation, even with subsequent use of a correct password.

The Crypto Officer loads authentication data into the Module during the Module build process. Authentication data cannot be accessed otherwise.

Since minimum password length is 16 characters, the probability of a random successful authentication attempt in one try is a maximum of $1/256^{16}$, or less than $1/10^{38}$. The Module permanently disables further authentication attempts after a single failure, so this probability is independent of time.

Both following roles have access to all the services provided by the Module.

User Role (User)

Loading the Module and calling any of the API functions.

Crypto Officer Role (CO)

Installation of the Module on the host computer system and calling of any API functions.

All services implemented by the Module are listed below, along with a description of service CSP access.

Table 9. Services and CSP Access

Service	Role	Description	Accesses CSPs
Initialize	User, CO	Initializes Module	–
Self-test	User, CO	Perform self-tests (FIPS_selftest)	–
Show Status	User, CO	Functions that provide module status information: Version (as unsigned `long` or `const char *`) FIPS Mode (Boolean)	–
Zeroize	User, CO	Functions that destroy CSPs: *`fips_rand_prng_reset`:: destroys RNG CSPs. *`fips_drbg_uninstantiate`:: for a given DRBG context, overwrites DRBG CSPs (Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRBG CSPs, Dual_EC_DRBG CSPs.) All other services automatically overwrite CSPs stored in allocated memory. Stack cleanup is the responsibility of the calling application.	X

Service	Role	Description	Accesses CSPs
Random Number Generation	User, CO	Used for random number and symmetric key generation. *Seed or reseed an RNG or DRBG instance *Determine security strength of an RNG or DRBG instance *Obtain random data Uses and updates RNG CSPs, Hash_DRBG CSPs, HMAC_DRBG CSPs, CTR_DRBG CSPs, Dual_EC_DRBG CSPs.	X
Asymmetric Key Generation	User, CO	Generates DSA, ECDSA and RSA keys: RSA SGK, RSA SVK DSA SGK, DSA SVK; ECDSA SGK, ECDSA SVK There is one supported entropy strength for each mechanism and algorithm type, the maximum specified in SP800-90Arev1.	X
Symmetric Encrypt/Decrypt	User, CO	Encrypts or decrypts data. Executes using AES EDK, TDES EDK (passed in by the calling process).	X
Symmetric Digest	User, CO	Generates or verifies data integrity with CMAC. Executes using AES CMAC, TDES, CMAC (passed in by the calling process).	X
Message Digest	User, CO	Generates a SHA-1 or SHA-2 message digest.	—
Keyed Hash	User, CO	Generates or verifies data integrity with HMAC. Executes using HMAC Key (passed in by the calling process).	X
Key transport	User, CO	Encrypts or decrypts a key value on behalf of the calling process (does not establish keys into the Module). Executes using RSA KDK, RSA KEK (passed in by the calling process).	X
Key Agreement	User, CO	Performs key agreement primitives on behalf of the calling process (does not establish keys into the Module). Executes using EC DH Private, EC DH Public (passed in by the calling process).	X
Digital Signature	User, CO	Generates or verifies RSA, DSA or ECDSA digital signatures. Executes using RSA SGK, RSA SVK; DSA SGK, DSA SVK; ECDSA SGK, ECDSA SVK (passed in by the calling process).	X
Utility	User, CO	Provides miscellaneous helper functions.	—

Note:

"Key transport" in the table above can refer to:

a. moving keys in and out of the Module or

b. the use of keys by an external application.

The latter definition is the one that applies to the IMB Cloud Object Store's FIPS Object Module.

Chapter 6 Self-Test

The Module performs the self-tests listed below on invocation of Initialize or Self-Test.

Table 10. Power On Self Tests

Algorithm	Type	Test Attributes
Software integrity	KAT	HMAC-SHA1
HMAC	KAT	One KAT per SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512. Per IG 9.3, this
AES	KAT	Separate encrypt and decrypt, ECB mode, 128-bit key length
AES CCM	KAT	Separate encrypt and decrypt, 192-bit key length
AES GCM	KAT	Separate encrypt and decrypt, 256-bit key length
XTS-AES	KAT	128-, 256-bit key sizes to support either the 256-bit key length (for XTS-AES-128) or the 512-bit key length (for XTS-AES-256)
AES CMAC	KAT	Sign and verify CBC mode, 128-, 192-, 256-key lengths
TDES	KAT	Separate encrypt and decrypt, ECB mode, 3-Key
TDES CMAC	KAT	CMAC generate and verify, CBC mode, 3-Key
RSA	KAT	Sign and verify using 2048-bit key, SHA-256, PKCS#1
DSA	PCT	Sign and verify using 2048-bit key, SHA-384
DRBG	KAT	CTR_DRBG: AES, 256-bit with and without derivation function; HASH_DRBG: SHA- 256; HMAC_DRBG: SHA-256;
ECDSA	PCT	Keygen, sign, verify using P-224, K-233 and SHA-512. The K-233 self-test is not performed for operational environments that support prime curve only (see *Table 2*).
ECC CDH	KAT	Shared secret calculation per SP 800-56A §5.7.1.2, IG 9.6
X9.31 RNG	KAT	128, 192, 256-bit AES keys

(KAT = Known answer test; PCT = Pairwise consistency test)

The Module is installed as a component of the IBM Cloud Object Storage operating system, *ClevOS FIPS-Edition*. The Module restricts key generation, digital signatures and encryption to FIPS approved algorithms only, when in FIPS mode. Attempts to use non-FIPS approved algorithms in FIPS mode will fail when attempted. *ClevOS FIPS-Edition* is the underlying operating system for the IBM Cloud Object Storage System. *ClevOS FIPS-Edition* uses the Module for all key generation, digital signature and encryption functions for data stored in the system by default.

The FIPS_mode_set() function performs all power-up self-tests listed above with no operator intervention required, returning a “1” if all power-up self-tests succeed, and a “0” otherwise. If any component of the power- up self-test fails an internal flag is set to prevent subsequent invocation of any cryptographic function calls. The module will only enter the FIPS Approved mode if the module is reloaded and the call to FIPS_mode_set() succeeds.

The power-up self-tests may also be performed on-demand by calling FIPS_selftest(), which returns a “1” for success and “0” for failure. Interpretation of this return code is the responsibility of the calling application.

The Module also implements the following conditional tests:

Table 11. Conditional Tests

Algorithm	Test
DRBG	Tested as required by [SP800-90Arev1] Section 11
DRBG	FIPS 140-2 continuous test for stuck fault
DSA	Pairwise consistency test on each generation of a key pair
ECDSA	Pairwise consistency test on each generation of a key pair
RSA	Pairwise consistency test on each generation of a key pair
ANSI X9.31 RNG	Continuous test for stuck fault

In the event of a DRBG self-test failure, ClevOS FIPS-Edition will uninstantiate and re-instantiate the DRBG per the requirements of [SP 800-90Arev1]. This is not something the Module can do itself. However, ClevOS can do this automatically.

Pairwise consistency tests are performed for both possible modes of use, e.g., Sign/Verify and Encrypt/Decrypt.

The Module supports one operational environment configuration for elliptic curve:

1. All NIST-defined curves (listed in Table 2 with the EC column marked BKP).

Chapter 7 Operational Environment

The tested operating system segregates user processes into separate process spaces. Each process space is logically separated from all other processes by the operating system software and hardware. The Module functions entirely within the process space of the calling application, and implicitly satisfies the FIPS 140-2 requirement for a single user mode of operation.

The tested operating system is a "limited modifiable" environment, only allowing system configuration information such as IP Addresses to be specified by the end user. All packages used by the operating system are under strict control of IBM. The IBM Cloud Object Storage System runs all processes under a predefined set of privileged users, and each process runs in a separate process space. Only one process on any ClevOS makes calls to a given instance of the Module.

At system boot, ClevOS FIPS-Edition uses system configuration settings to force all processes to call the Module's FIPS_mode_set(). FIPS_mode_set() then calls functions that perform all POSTs listed above without additional operator intervention.

The IBM Cloud Object Storage System Manager indicates that the IBM appliance is in FIPS mode after it has received verification that all IBM Cloud Object Storage System components have entered FIPS mode. If any component of IBM Cloud Object Storage System fails the POST, the IBM Cloud Object Storage System Manager indicates that the appliance failed to go into FIPS mode. The IBM Cloud Object Storage System Manager displays the FIPS mode of each component in the IBM Cloud Object Storage System to allow the operator to diagnose which devices cannot enter FIPS mode.

Components that cannot enter FIPS mode are marked as non-compliant and prevented from participating in the IBM Cloud Object Storage System.

The FIPS operation state can be tested at the CLI with a call to get_fips_mode. get_fips_mode returns a 1 if the device is in FIPS mode and a 0 if it is not in FIPS mode.

Chapter 8 Mitigation of Other Attacks

The Module is not designed to mitigate attacks which are outside of the scope of FIPS 140-2.

Appendix A: Installation and Usage Guide

The tested platform uses Debian packaging to build and install the module. Following standard conventions, the *dpkg-buildpackage* command is used by the build system to create a Debian package for the Module. Then the standard *dpkg* utility is used to make and install the module onto the platform. A file named *rules* is created for the Debian packaging process to direct the compilation of the Module's source. The *rules* files and other Debian specific packaging files are placed in a Debian directory located at the same level of the directory structure the as the directory containing the uncompressed Module source code from openssl-fips-2.0.10.tar.gz. The commands in the *rules* file are executed relative to the top of the directory containing that uncompressed and expanded content.

The Debian package rule file contains the following two commands that compile the module from the source. These commands are issued in a shell at the top level of the Module's source code directory structure.

```
rules:
        ./config
      make
```

Source Assurance and Installation

To assure the integrity of the source code for the module, a copy of opensslfips2.0.10.tar.gz distributed on CD was requested for delivery by UPS. The HMACSHA1 digest of the distribution file on the CD was validated against the digest published in Appendix B of the OpenSSL Security Policy. The installation process is listed below:

1. Obtain a copy of the Module's distribution file openssl-fips-2.0.10.tar.gz from the OpenSSL Software Foundation (OSF).
2. Copy the distribution tar file openssl-fips-2.0.10.targ.gz to the build environment.
3. Validate that the HMAC-SHA-1 signature matches the result stated in Appendix B of this Security Policy.
4. Unpack the source from the distribution file using the following command
    a. tar –xvf openssl-fips-2.0.10.tar.gz
5. Build the fipscanister package using the instructions in front of this section.
6. Place the package in the build pipeline so that the *dpkg* utility installs the fipscanister into ClevOS during the standard build process.

Linking the Runtime Executable Application

Note that applications interfacing with the Module are outside of the cryptographic boundary. When linking the application with the FIPS Object Module two steps are necessary:

1. The HMAC-SHA-1 digest of the FIPS Object Module file must be calculated and verified against the installed digest to ensure the integrity of the Module.
2. A HMAC-SHA-1 digest of the Module must be generated and embedded in the Module for use by the FIPS_mode_set()[10] function at runtime initialization.

The ClevOS build system uses *make* facilities (via Debian packaging) to embed the digest in OpenSSL at build time. Failure to embed the digest in the executable object will prevent initialization of FIPS mode.

At runtime, the FIPS_mode_set() function compares the embedded HMAC-SHA-1 digest generated from the FIPS Object Module object code. This digest is the final link in the chain of validation from the original source to the runtime executable application file. The ClevOS operating system enables FIPS mode by calling FIPS_module_set()

before allowing any customer data to be encrypted by the system.

Optimization

The platform was tested against both AES-NI acceleration on and off. ClevOS if AES-NI acceleration is available on the platform and AES-NI will always be used by the platform.

Appendix B: Controlled Distribution File Fingerprint

The IBM Object Store FIPS Cryptographic Module v1.1 consists of the FIPS Object Module (the *fipscanister.o* or *fipscanister.lib* contiguous unit of binary object code) generated from the source files contained in the distribution file openssl-fips-2.0.10.tar.gz.

The SHA-1 digest of package containing the source used to create the canister is

af8bda4bb9739e35b4ef00a9bc40d21a6a97a780

The *openssl* command below may be run from the CLI of a ClevOS FIPS Edition to calculate the digest:

- openssl sha1 -hmac etaonrishdlcupfm openssl-fips-2.0.10.tar.gz

The set of files specified in this tar file constitutes the complete set of source files of this module. There shall be no additions, deletions, or alterations of this set as used during the building of the module. The OpenSSL distribution tar file (and patch file if used) shall be verified using the above HMAC-SHA-1 digest(s).

For traceability, one can use the 16-byte key (specified by OSF) of:

65 74 61 6f 6e 72 69 73 68 64 6c 63 75 70 66 6d

(equivalent to the ASCII string "etaonrishdlcupfm") to generate the HMAC-SHA-1 value for the FIPS Object Module integrity check.

Appendix C: Compilers

This appendix lists the specific compilers used to generate the Module for the respective Operational Environments. Note this list does not imply that use of the Module is restricted to only the listed compiler versions, only that the use of other versions has not been confirmed to produce a correct result.

#	Operational Environment	Compiler
1	ClevOS 3.8.x FIPS Edition	gcc 4.7.2

Appendix D: Glossary

Abbreviation	Term
AES	Advanced Encryption Standard
ANSI	American National Standards Institute
API	application program interface
B	B Elliptic Curve
CBC-MAC	Cipher Block Chaining Message Authentication Code
CBC	Cipher Block Chaining
CCM	Counter with CBC-MAC
CMAC	Cipher-based Message Authentication Code
CSP	Critical Security Parameter
CTR_DRBG	Counter mode Deterministic Random Byte Generator
ECB	Electronic Code Book
DH	Diffie-Hellman scheme
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Algorithm
Dual_EC_DRBG	Dual Elliptic Curve Deterministic Random Bit Generator
IBM Cloud Object Storage System	The IBM Cloud Object Storage System Appliances are the general purpose computing devices where the Module is installed.
EC	Elliptic Curve
ECC CDH	Elliptic Curve Cryptography Cofactor Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EDK	Encrypt/Decrypt Key
FIPS	Federal Information Processing Standards
GCM	Galois/Counter Mode
HASH_DRBG	Hash - Deterministic Random Bit Generator
HMAC	Keyed-Hash Message Authentication Code
HMAC_DRBG	Keyed-Hash Message Authentication Code Deterministic Random Bit Generator
K	K Elliptic Curve
KAS	Key Agreement Scheme
KAT	Known Answer Test Power on Self-Test
KDK	Key Decryption Key
KEK	Key Encryption Key
NIST	National Institute of Standards and Technology
P	Prime Elliptic Curve
PCT	Pairwise Consistency Test Power on Self-Test
PKCS	Public-Key Cryptography Standards
PKG	Public Key Generation
PKV	Public Key Validation
POST	Power-up Self-Test
RNG	Random Number Generator
RSA	Rivest-Shamir-Adleman Cryptosystem
SGK	Signature Generation Key