AFORE Solutions
CloudLink Crypto Module

Software Version 1.0

FIPS 140-2 Non-Proprietary Security Policy
Level 1 Validation

February 28, 2014
Document Version 1.4

Table of Contents

1 INTRODUCTION

1.1 Module Description/Module Specification

This document is the non-proprietary security policy for the AFORE Solutions CloudLink Crypto Module. The module is a software shared library providing a C-language application program interface (API) for use by other processes that require cryptographic functionality and is based directly on the OpenSSL FIPS Object Module.

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

Security Requirements Section	*Level*
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles and Services and Authentication	1
Finite State Machine Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	N/A
Cryptographic Module Security Policy	1
Overall Level of Certification	1

Table 1 - Security Level of Security Requirements

2 PRODUCT DESCRIPTION

The module is classified by FIPS 140-2 as a software module, multi-chip standalone module embodiment. The physical cryptographic boundary is the CloudLink virtual appliance. The logical cryptographic boundary of the Module is the ***fipscanister*** object module, a single object module file named ***fipscanister.o***. The Module performs no communications other than with the calling application (the process that invokes the Module services).

2.1 Cryptographic Boundary

2.1.1 Physical Cryptographic Boundary

The physical boundary of the module is the physical boundary of the virtual machine that contains the module. Figure 2 below shows the physical block diagram of the Dell PowerEdge R520 with Intel® Xeon® E5-2420 processor hardware.

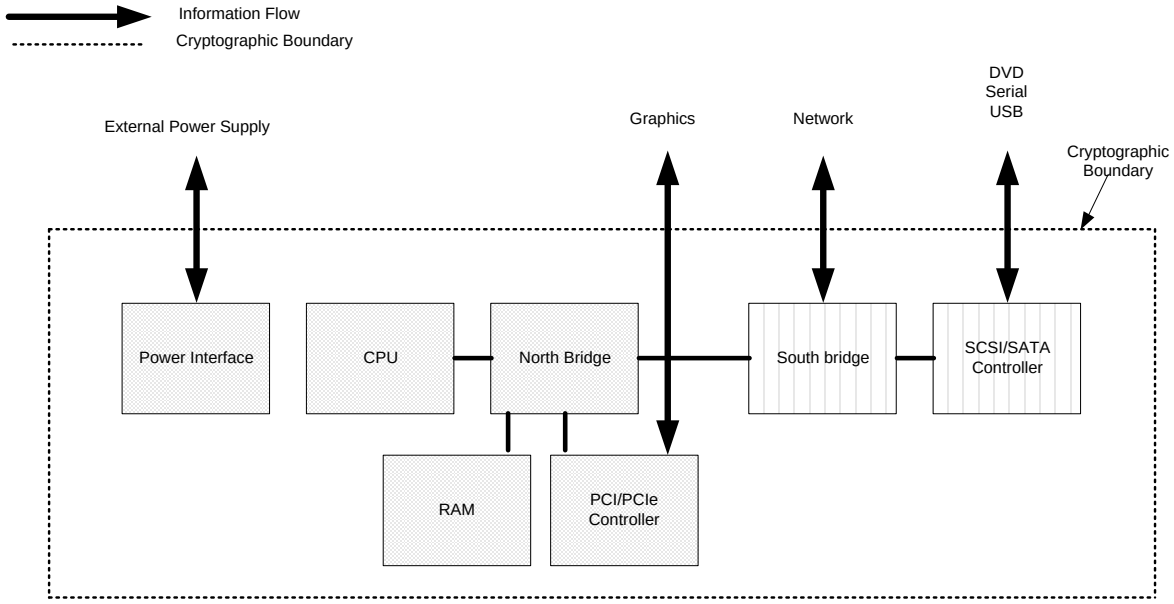

Figure 1 - Physical Cryptographic Boundary

2.1.2 Logical Cryptographic Boundary

The logical boundary of the module is the single Linux object which is not part of the Linux OS (see Figure 2).

3 MODULE PORTS AND INTERFACES

As a software module, the module has no physical characteristics. Thus, the module's manual controls, physical indicators, and physical and electrical characteristics are the same as the computer system on which it is executing. The "Application" represents other software component loaded on the appliance that employs the module's services.

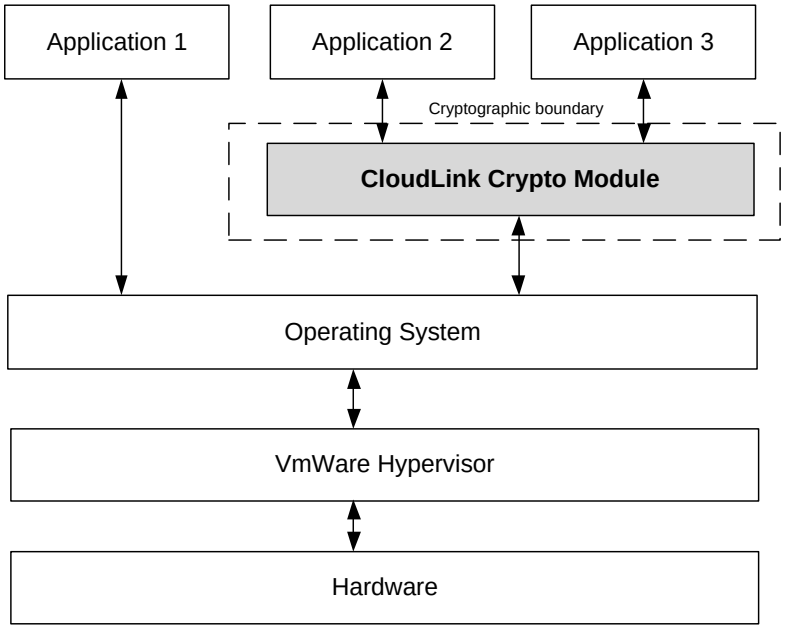

Figure 2 – Logical Cryptographic Boundary and Interface Diagram

The logical interface is a C-language application program interface (API).

Logical interface	Physical interface	Description
Control input	*Network/Serial/USB port, PCIe slot, Power button*	*API entry point and corresponding stack parameters*
Data input	*Network/Serial/USB port, DVD, PCIe slot*	*API entry point data input stack parameters*
Status output	*Network/Serial/USB port, Graphics/Video port, PCIe slot*	*API entry point return values and status stack parameters*
Data output	*Network/Serial/USB port, DVD, Graphics/Video port,*	*API entry point data output stack parameters*

	PCIe slot	

Table 2 - FIPS interface mappings

As a software module, control of the physical ports is outside 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).

4 ROLES, SERVICES AND AUTHENTICATION

The module supports a Crypto Officer and User roles which implicitly assume all services by both roles. The module doesn't support a maintenance role.

4.1 Roles and Services

Both roles have access to all of 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 3 lists Allowed roles and services.

Service	Role	Description
Initialize	User, CO	Module initialization. Does not access CSPs.
Self-test	User, CO	Perform all power up tests (FIPS_selftest). Does not access CSPs.
Show status	User, CO	Functions that provide module status information: • Version (as unsigned long or const char *) • FIPS Mode (Boolean) Does not access CSPs.
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) All other services automatically overwrite CSPs stored in allocated memory. Stack cleanup is the responsibility of the calling application. • reboot destroys all 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.
Asymmetric key generation	User, CO	Used to generate DSA and RSA keys: RSA SGK, RSA SVK; DSA SGK, DSA SVK; There is one supported entropy strength for each mechanism and algorithm type, the maximum specified in SP800-90A
Symmetric encrypt/decrypt	User, CO	Used to encrypt or decrypt data. Executes using AES EDK, Triple -DES EDK (passed in by the calling process).
Symmetric digest	User, CO	Used to generate or verify data integrity with CMAC. Executes using AES CMAC, Triple -DES CMAC (passed in by the calling process).
Message digest	User, CO	Used to generate a SHA-1 or SHA-2 message digest. Does not access CSPs.
Keyed Hash	User, CO	Used to generate or verify data integrity with HMAC.

Service	Role	Description
		Executes using HMAC Key (passed in by the calling process).
Key Agreement	User, CO	EC Diffie-Hellman (key agreement; key establishment methodology provides between 112 and 256 bits of encryption strength; non-compliant less than 112 bits of encryption strength). Used to perform 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).
Key transport[1]	User, CO	RSA (key wrapping; key establishment methodology provides between 112 and 256 bits of encryption strength; non-compliant less than 112 bits of encryption strength) Used to wrap or unwrap 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).
Digital signature	User, CO	FIPS 186-2-compliant implementations shall not be used for key generation or signature generation.Used to generate or verify RSA or DSA digital signatures. Executes using RSA SGK, RSA SVK; DSA SGK, DSA SVK;
Utility	User, CO	Miscellaneous helper functions. Does not access CSPs.

Table 3[2] - Roles and Services

Table 4 lists non-Allowed roles and services.

Service	Role	Description
Asymmetric key generation	User, CO	Used to generate asymmetric keys. RSA SGK (1024/1536 with all SHA sizes, 2048/3072/4096 with SHA-1), DSA SGK (1024/1536 with all SHA sizes, 2048/3072 with SHA-1). There is one supported entropy strength for each mechanism and algorithm type, the maximum specified in SP800-90A
Digital signature	User, CO	Used to generate or verify digital signatures using ECDSA SGK and ECDSA SVK. Used to generate digital signatures using RSA SGK (1024/1536 with all SHA sizes, 2048/3072/4096 with SHA-1), DSA SGK (1024/1536 with all SHA sizes, 2048/3072 with SHA-1)
Key Agreement	User, CO	ECC CDH (CVL) No claim is made for SP 800-56A (§5.7.1.2) compliance
Random number generation	User, CO	No claim is made for SP 800-90A Dual-EC DRBG compliance

Table 4 - Non-Approved Services

[1] "Key transport" refers to the use of keys by an external application.

[2] Some RSA and DSA algorithms are no longer permitted in asymmetric key generation. Please refer Table 4 for additional information

5 PHYSICAL SECURITY

The module does not claim physical security requirements for this section.

6 Operational Environment

The tested operating systems segregate 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 module has been tested on Ubuntu Linux Version 12.04 on VMware ESXi 5.1.0 running on Dell PowerEdge R520 with Intel® Xeon® E5-2420 processor.

7 CRYPTOGRAPHIC KEY MANAGEMENT

7.1 Cryptographic Keys and CSP's

For all CSPs 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), with the exception of RNG and DRBG state values used for the Modules' default key generation service.

Generation: The Module implements ANSI X9.31 compliant RNG and SP 800-90A compliant DRBG services for creation of symmetric keys, and for generation of DSA, elliptic curve, and RSA keys as shown in Table 5. The calling application is responsible for storage of generated keys returned by the module. The module complies with IG 7.8 Scenario 1 for symmetric key generation as well as the seed supplied to the algorithm for generating asymmetric keys.

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 re-distributed.

The following table summarizes the module's keys and CSP's:

CSP Name	Description
RSA SGK	RSA (2048 to 16384 bits) signature generation key
RSA KDK	RSA (2048 to 16384 bits) key decryption (private key transport) key
DSA SGK	[FIPS 186-3] DSA (2048/3072) signature generation key
AES EDK	AES (128/192/256) encrypt / decrypt key
AES CMAC	AES (128/192/256) CMAC generate / verify key
AES GCM	AES (128/192/256) encrypt / decrypt / generate / verify key
AES XTS	AES (256/512) XTS encrypt / decrypt key
Triple -DES EDK	Triple -DES (3-Key) encrypt / decrypt key
Triple -DES CMAC	Triple -DES (3-Key) CMAC generate / verify key
Hash_DRBG CSPs	V (440/880 bits) and C (440/880 bits), entropy input (length dependent on security strength)
RNG CSPs	Seed (128bit), AES128/192/256 seed key and associated state variables for ANSI X9.31 AES based RNG
HMAC_DRBG CSPs	V (160/224/256/384/512 bits) and Key (160/224/256/384/512 bits), entropy input (length dependent on security strength)
CTR_DRBG CSPs	V (128 bits) and Key (AES 128/192/256), entropy input (length dependent on security strength)
HMAC Key[3]	Keyed hash key (160/224/256/384/512)

Table 5 - Cryptographic Module Keys and CSP's

CAVEATS:

[3] HMAC-SHA-1 shall have a key size of at least 112 bits

Entropy:

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

Transition period:

- Digital signature generation that provides less than 112 bits of security (RSA key sizes less than 2048-bits or DSA 'p' less than 2048-bit OR 'q' less than 224 bits) is deprecated until December 31st, 2013. It is disallowed beginning January 1st, 2014.
- Digital signature generation using SHA-1 as its underlying hash function is deprecated until December 31st, 2013. It is disallowed beginning January 1st, 2014.
- Random number generation using the ANSI X9.31 Appendix A.2.4 PRNG is deprecated until December 31st, 2015. It is disallowed beginning January 1st, 2016.
- RSA key transport schemes that provide less than 112 bits of security (RSA key sizes less than 2048-bits) are deprecated until December 31st, 2013. They are disallowed beginning January 1st, 2014.

7.2 Key Destruction/Zeroization

Destruction/Zeroization of sensitive data is performed automatically by API function calls for temporarily stored CSPs. In addition, the module provides functions to explicitly destroy CSPs related to random number generation services. The calling application is responsible for parameters passed in and out of the module. The entire module is zeriozed on reboot. All ephemeral keys used by the module are zeroized upon reboot.

7.3 Key Entry/Output

For all CSPs and Public Keys:
Entry: All CSPs enter the Module's logical boundary in plaintext 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.

The module does not output intermediate key generation values.

CSP Name	Description
RSA SVK	RSA (1024 to 16384 bits) signature verification public key
RSA KEK	RSA (2048 to 16384 bits without SHA-1) key wrapping/unwrapping
DSA SVK	[FIPS 186-3] DSA (1024/2048/3072) signature verification key or [FIPS 186-2] DSA (1024) signature verification key
EC DH Public	EC DH (All NIST Recommended B, K and P curves except sizes 163 and 192) public key agreement key.

Table 6 - Public Keys

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. All API functions are executed by the invoking calling application in a non-overlapping sequence such that no two API functions will execute concurrently. An authorized application user (Crypto-Officer and User) has access to all key data generated during the operation of the Module.

7.4 Approved or Allowed Security Functions

Tables 7 and 8 list the Approved and Non-approved but Allowed algorithms, respectively

Function	Algorithm	Options	Cert #
Random Number Generation; Symmetric key generation	[ANS X9.31] RNG	AES 128/192/256	Cert# 1220
	[SP 800-90] DRBG[4] Prediction resistance supported for all variations	Hash DRBG HMAC DRBG, no reseed CTR DRBG (AES), no derivation function	#378
Encryption, Decryption and CMAC	[SP 800-67]	3-Key Triple-DES TCFB; CMAC generate and verify	#1540
	[FIPS 197] AES	128/ 192/256 ECB, CBC, OFB, CFB 1, CFB 8, CFB 128, CTR, XTS; CCM; GCM; CMAC generate and verify	#2545
	[SP 800-38B] CMAC [SP 800-38C] CCM [SP 800-38D] GCM [SP 800-38E] XTS		
Message Digests	[FIPS 180-3]	SHA-1, SHA-2 (224, 256, 384, 512)	#2146
Keyed Hash	[FIPS 198] HMAC	SHA-1, SHA-2 (224, 256, 384, 512)	#1566
Digital Signature and Asymmetric Key Generation	[FIPS 186-2] RSA	SigVer9.31, SigVerPKCS1.5, SigVerPSS (1024/1536/2048/3072/4096 with all SHA sizes) GenKey9.31, SigGen9.31, SigGenPKCS1.5,SigGenPSS (2048/3072/4096 with all SHA-2 sizes)	#1300
	[FIPS 186-2] DSA	PQG Ver, Sig Ver (1024 with SHA-1 only)	#778
	[FIPS 186-3] DSA	PQG Ver, Sig Ver (1024/2048/3072 with all SHA sizes) PQG Gen, Key Pair Gen, Sig Gen (2048/3072 with all SHA-2 sizes)	#778
CVL (TLS KDF)	[SP 800-135rev1] TLS KDF	TLS 1.0/1.1/1.2 versions[5]	#104

Table 7 - FIPS Approved Cryptographic Functions

[4]For all DRBGs the "supported security strengths" is just the highest supported security strength per [SP800-90A] and [SP800-57].

[5] CAVP and CMVP have not tested the TLS protocol.

Function	Algorithm	Options	Cert #
Digital Signature and Asymmetric Key Generation	[FIPS 186-2] RSA	GenKey9.31, SigGen9.31, SigGenPKCS1.5, SigGenPSS	#1300
	[FIPS 186-2] DSA	PQG Gen, Key Pair Gen, Sig Gen(1024 with SHA-1 only)	#778
	[FIPS 186-3] DSA	PQG Gen, Key Pair Gen, Sig Gen (1024 with all SHA sizes, 2048/3072 with SHA-1)	#778

Table 7a[6] – Historical FIPS Approved Cryptographic Functions

Category	Algorithm	Description
Key Wrapping, Key Unwrapping	RSA (key wrapping; key establishment methodology provides between 112 and 256 bits of encryption strength; non-compliant less than 112 bits of encryption strength)	The RSA algorithm may be used by the calling application for wrapping or unwrapping 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.
Key Agreement	EC Diffie-Hellman (key agreement; key establishment methodology provides between 112 and 256 bits of encryption strength; non-compliant less than 112 bits of encryption strength)	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.

Table 8 - Non-Approved but Allowed Functions

7.5 Non-Approved and non-Allowed Security Functions

Table 9 lists the Non-Approved and Non-approved but Allowed algorithms.

Category	Algorithm	Description
Digital Signature and Asymmetric Key Generation	[FIPS 186-2] ECDSA	No claim is made for [FIPS 186-2] ECDSA and [FIPS 186-3] ECDSA compliance
	[FIPS 186-3] ECDSA	
	[FIPS 186-2] RSA	GenKey9.31, SigGen9.31, SigGenPKCS1.5, SigGenPSS (1024/1536 with all SHA sizes, 2048/3072/4096 with SHA-1)
	[FIPS 186-2] DSA	PQG Gen, Key Pair Gen, Sig Gen (1024 with all SHA sizes, 2048/3072 with SHA-1)
	[FIPS 186-3] DSA	PQG Gen, Key Pair Gen, Sig Gen (1024 with all SHA sizes, 2048/3072 with SHA-1)
ECC CDH (KAS)	[SP 800-56A] (§5.7.1.2)	No claim is made for SP 800-56A (§5.7.1.2) compliance

[6] The algorithms listed in Table 7a are not to be used in FIPS Approved mode

Random number generation	[SP 800-90A] Dual-EC DRBG	No claim is made for SP 800-90A Dual-EC DRBG compliance

Table 9 - Non-Approved and non-Allowed Functions

8 SELF-TEST

The module performs power-up self-tests and conditional self-tests.

8.1 Power-up Self-Tests

Algorithm	Type	Test Attributes
Software integrity	KAT	HMAC-SHA1
HMAC	KAT	One KAT per SHA1, SHA224, SHA256, SHA384 and SHA512 Per IG 9.3, this testing covers SHA POST requirements.
AES	KAT	Separate encrypt and decrypt, ECB mode, 128 bit key length
AES CCM	KAT	Separate encrypt and decrypt, 192 key length
AES GCM	KAT	Separate encrypt and decrypt, 256 key length
XTS-AES	KAT	128, 256 bit key sizes to support either the 256-bit key size (for XTS-AES-128) or the 512-bit key size (for XTS-AES-256)
AES CMAC	KAT	Sign and verify CBC mode, 128, 192, 256 key lengths
Triple -DES	KAT	Separate encrypt and decrypt, ECB mode, 3-Key
Triple -DES 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: SHA256 HMAC_DRBG: SHA256
X9.31 RNG	KAT	128, 192, 256 bit AES keys

Table 10 - Power On Self Tests (KAT = Known answer test; PCT = Pairwise consistency test)

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.

8.2 Conditional Self-Tests

The module performs the following conditional self-tests:
Conditional RNG Test: A conditional RNG is performed for each approved and RNG implemented within the module
Pairwise Consistency Test: Pairwise consistency tests are run on demand when the module generates RSA key pairs. The module performs a sign operation with the private key and verifies it with the public key.

Algorithm	Test
DRBG	Tested as required by [SP800-90] Section 11
DRBG	FIPS 140-2 continuous test for stuck fault
DSA	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

Table 11 - Conditional Tests

The module implements both the SP 800-90 DRBGs (with the exception of Dual_EC_DRBG) and X9.31 PRNG as its random number generator. This employs four critical functions which must also be tested on a regular basis to ensure the security of the SP 800-90 DRBG: Instantiate test, Reseed test, Generate test and Un-Instantiate test.
In the event of a DRBG self-test failure the calling application must uninstantiate and reinstantiate the DRBG per the requirements of SP 800-90A; this is not something the Module can do itself. The uninstantiation of the DRBG by the calling application zeroizes the internal state of the DRBG to ensure it is not accessible prior to the reinstatiation of the DRBG."

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

When the module is in a self-test state or error state, all data output is inhibited

9 Crypto-Officer and User Guidance

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 until the module is properly reinitialized by the calling application.

The application can test to see if FIPS mode has been successfully performed

There are no configuration instructions required for the Module.

10 Mitigation of Other Attacks

This section is not applicable. The module is not designed to mitigate against other attacks.