Ionic Security Inc.

FIPS Crypto Module

Software Version: 1.1

FIPS 140-2 Non-Proprietary Security Policy

FIPS Security Level: 1
Document Version: 1.3

Prepared for:

Ionic Security Inc.
1170 Peachtree Street NE
Suite 400
Atlanta, GA 30309
United States of America

Phone: +1 404 736 6000
www.ionicsecurity.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 Figure

1. Introduction

1.1 Purpose

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

This document also describes how to run the module in a secure FIPS-Approved mode of operation. This Security Policy was prepared as part of the Level 1 FIPS 140-2 validation of the module. The FIPS Crypto Module is referred to in this document as FCM, crypto module, or the module.

1.2 References

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

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

1.3 Document Organization

The Security Policy document is one document in a FIPS 140-2 Submission Package. In addition to this document, the Submission Package contains:

- Vendor Evidence document
- Finite State Model document
- Submission Summary document
- Other supporting documentation as additional references

This Security Policy and the other validation submission documentation were produced by Corsec Security, Inc. under contract to Ionic Security. With the exception of this non-proprietary Security Policy, the FIPS 140-2 Submission Package is proprietary to Ionic Security. For access to these documents, please contact Ionic Security.

2. FIPS Crypto Module

2.1 Overview

The Ionic Security FIPS Crypto Module (FCM) is a cryptographic module within Ionic Security's data protection platform, "Ionic.com".

Ionic.com is a unified data and mobility security platform for the enterprise, providing access control, intellectual property monitoring, data encryption, and policy management, and is implemented as a hybrid PaaS[1] system combined with device endpoint instrumentation and enforcement software.

It is a next-generation solution offering increased data protection, particularly when confronted with the security challenges presented by cloud services and BYOD[2] mobile computing. Ionic.com provides an administrative GUI[3] called the Dashboard, which is the management point for adding and modifying applications, policies, users and groups, devices, and other elements of the system.

The Dashboard is the user's interface to Ionic Security's Key Management, Authentication, Audit, and Policy Management functions hosted in the Ionic.com set of services. These services communicate with the endpoint software (the "Ionic plugins"), as they work together to execute data protection and to control plugin operations and updates.

Ionic.com implements the FCM for all cryptographic functions such as key pair generation, digital signature generation and verification, symmetric encryption and decryption, hashing, message authentication, and random number generation.

The FCM is an independent and modular component of the system. The module provides access to all cryptographic functions through an abstraction layer. The abstraction layer provides C language API[4]s, which can be called to easily access the cryptographic engine.

The FCM is validated at the FIPS 140-2 Section levels shown in Table 1.

[1] PaaS – Platform as a Service
[2] BYOD – Bring Your Own Device
[3] GUI – Graphical User Interface
[4] API – Application Programming Interface

Table 1 – Security Level per FIPS 140-2 Section

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

2.2 Module Specification

The FIPS Crypto Module (FCM) is a software module with a multiple-chip standalone embodiment. The overall security level of the module is 1. The module was tested and found to be FIPS 140-2 compliant on the following platforms:

- Hewlett-Packard (HP) Z230 desktop running an Intel Core i7 processor with a Windows 7 OS[6]
- Intel Server System R1304GZ4GC running an Intel Xeon E5 processor with CentOS[7] 7.1.

As a software cryptographic module, there are no physical protection mechanisms implemented. Therefore, the module must rely on the physical characteristics of the host system. The physical boundary of the cryptographic module is defined by the hard enclosure around the host system on which it runs. The module supports the physical interfaces of the HP Z230 desktop and the Intel Server System. These interfaces include the network interface, LED[8]s, serial ports, and USB[9] ports. See Figure 1

[5] EMI/EMC – Electromagnetic Interference / Electromagnetic Compatibility
[6] OS – Operating System
[7] CentOS – Community Enterprise Operating System
[8] LED – Light-Emitting Diode
[9] USB – Universal Serial Bus

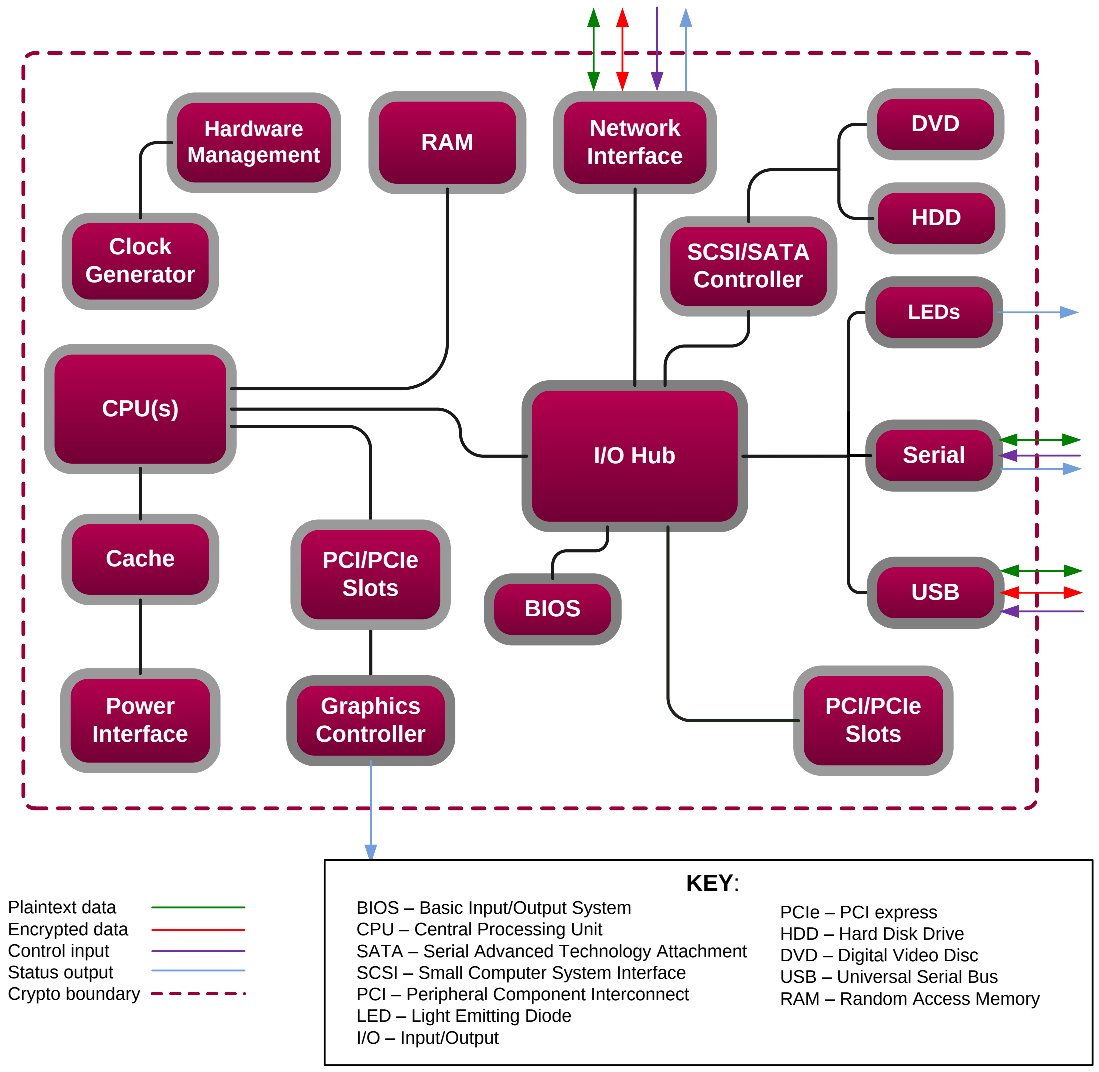

Figure 1 – FCM Physical Cryptographic Boundary

Figure 2 below shows a logical block diagram of the module executing in memory and its interactions with surrounding software components, as well as the module's logical cryptographic boundary[10]. The files and binaries that make up the cryptographic module are shown as "FIPS Crypto Module" in the diagram below. The module's services are designed to be called by the Ionic Security Fusion Platform. The module's logical boundary is a contiguous perimeter that surrounds all memory-mapped functionality provided by the module when loaded and stored in the host platform's memory.

[10] The logical boundary of the module consists of a single file, "CryptoAbstract.dll" for Windows and "CryptoAbstract.so" for CentOS.

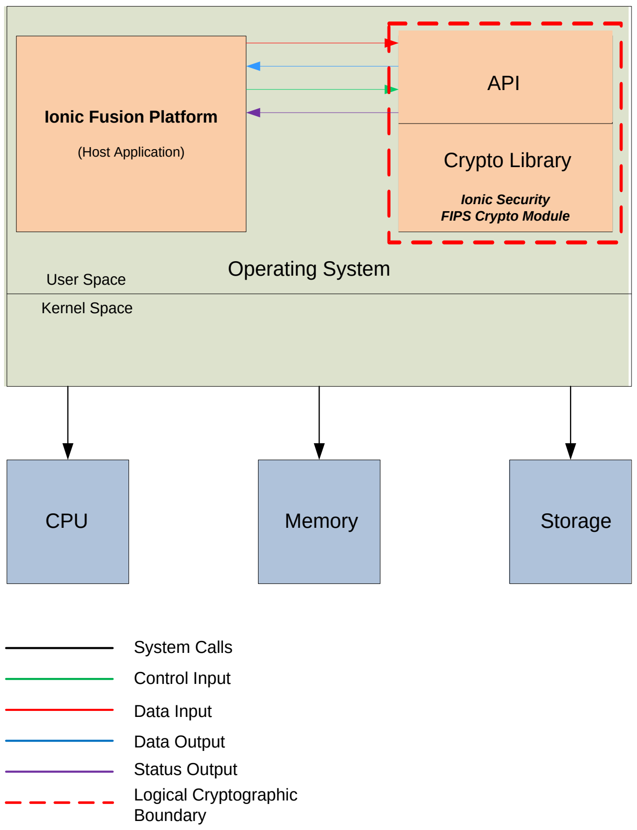

Figure 2 – FCM Logical Cryptographic Boundary

2.3 Module Interfaces

The module's logical interfaces exist at a low level in the software as an API. Both the API and physical interfaces can be categorized into the following interfaces defined by FIPS 140-2:

- Data input
- Data output
- Control input
- Status output

As a software module, the module has no physical characteristics. The module's manual controls, physical indicators, and physical and electrical characteristics are those of the host platform. A mapping of the FIPS 140-2 logical interfaces, the physical interfaces, and the module interfaces can be found in Table 2 below.

Table 2 – FIPS 140-2 Logical Interface Mappings

FIPS 140-2 Interface	Physical Interface	Module Interface (API)
Data Input	USB ports (keyboard, mouse), network interface, serial ports	Arguments for API calls that contain data to be used or processed by the module
Data Output	USB ports, network interface, serial ports	Arguments for API calls that contain or point to where the result of the function is stored
Control Input	USB ports (keyboard, mouse), network interface, serial ports	API Function calls and parameters that initiate and control the operation of the module
Status Output	Graphic controller, network interface, serial ports, LEDs	Return values from API function calls and error messages

2.4 Roles and Services

The module does not support an authentication mechanism. There are two roles in the module (as required by FIPS 140-2) that operators may assume: a Crypto Officer (CO) role and a User role. Roles are assumed implicitly through the execution of either a CO or User service. Each role and their corresponding services are detailed in the sections below. Please note that the keys and Critical Security Parameters (CSPs) listed in Table 3 and Table 4 below indicates the types of access required using the following notation:

- R – Read: The CSP is read in from memory.
- 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.

2.4.1 Crypto Officer Role

To assume the CO role, an operator of the module will perform one of the services listed in Table 3. The CO has the ability to initialize and shutdown the module, show status, and run self-tests on demand.

Table 3 – Crypto Officer Services

Service	Description	CSP and Type of Access
Initialize module	Place the module into the Approved-mode, Perform integrity check and power-up self-tests	None
Show status	Returns the current mode/status of the module	None
Shutdown	Zeroizes and de-allocates memory containing sensitive data	All keys – W
Run self-tests on demand	Power cycle the host system and perform integrity check and power-up self-tests	None

2.4.2 User Role

To assume the User role, an operator of the module will perform one of the services listed in Table 4. The User has the ability to generate symmetric and asymmetric keys, perform symmetric encryption and decryption, generate random numbers, and generate a hash and message authentication code.

Table 4 – User Services

Service	Description	CSP and Type of Access
Symmetric encryption (AES[11] CTR[12])	Encrypt plaintext using supplied key and algorithm specification	AES key – RX AES CTR IV - WX
Symmetric decryption (AES CTR)	Decrypt ciphertext using supplied key and algorithm specification	AES key – RX AES CTR_IV - RX
Symmetric encryption (AES ECB[13])	Encrypt plaintext using supplied key and algorithm specification	AES key – RX
Symmetric decryption (AES ECB)	Decrypt ciphertext using supplied key and algorithm specification	AES key – RX
Symmetric encryption (AES GCM[14])	Encrypt plaintext using supplied key and algorithm specification	AES GCM Key - RX AES GCM IV – WX
Symmetric decryption (AES GCM)	Decrypt ciphertext using supplied key and algorithm specification	AES GCM Key - RX AES GCM IV – RX
Retrieve AES initialization vector length	Determine the length of the AES ciphertext initialization vector (Non-AES GCM)	None
Retrieve AES key length	Determine AES ciphertext key length	None
Retrieve AES GCM tag length	Determine AES-GCM tag length	None
Generate hash (256 and 512)	Compute and return a hash	None
Generate HMAC hash (256 and 512)	Compute and return a message digest authorization code	HMAC key – RX
Generate private key	Generate an RSA private key	RSA[15] private key - W
Generate public key	Generates an RSA public key	RSA public key - W
Save private key	Save private key to specified memory location	RSA private key - R
Save public key	Save public key to specified memory location	RSA public key - R
Load private key	Load private key for use by the module	RSA private key - W
Load public key	Load public key for use by the module	RSA public key - W

[11] AES – Advance Encryption Service
[12] CTR – Counter
[13] ECB – Electronic Code Book
[14] GCM – Galois Counter Mode
[15] RSA – Rivest, Shamir, Adleman

Service	Description	CSP and Type of Access
RSA encrypt	Encrypt plaintext keys (wrap) using supplied key and algorithm specification	RSA private key – RX
RSA decrypt	Decrypt ciphertext keys (unwrap) using supplied key and algorithm specification	RSA public key – RX
Release private key	Release private key object. Key object is no longer available.	RSA private key – W
Release public key	Release public key object. Key object is no longer available.	RSA public key – W
Signature generation	Generate a signature for the supplied message using the specified key and algorithm	RSA private key – RX
Signature verification	Verify the signature on the supplied message using the specified key and algorithm	RSA public key – RX
Generate random number	Returns the specified number of random bits to calling application	DRBG seed – X DRBG entropy – RX
Seed random bit generator (user supplied value)	Seed the random bit generator with an optional additionally supplied user value[16]	DRBG Seed – RX
Automatically seed DRBG	Automatically seed the DRBG from the entropy source	DRBG entropy – RX
PBKDF2[17] Hash	Computes PBKDF2 hash	PBKDF2 Password – RX

2.5 Physical Security

The FCM is a software module, which FIPS defines as a multiple-chip standalone cryptographic module. As such, it does not include physical security mechanisms. Thus, the FIPS 140-2 requirements for physical security are not applicable.

2.6 Operational Environment

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

- Hewlett-Packard (HP) Z230 desktop running an Intel Core i7 processor with a Windows 7 SP1 OS
- Intel Server System R1304GZ4GC running an Intel Xeon E5 processor with CentOS 7.1.

[16] Note that this service is for an "optional additional input" that can be supplied by a user which contains high entropy data from external sources. This input does not bypass the in-built entropy pool but is feed directly into the DRBG as an additional input.
[17] PBKDF2 – Password-Based Key Derivation Function 2 - PBKDF2 is published in Internet Engineering Task Force Request for Comments (RFC) 2898 and maps to PBKDF defined in NIST SP 800-132.

The FCM is a software module so it relies on the CPU[18] of the host system. All cryptographic keys and CSPs are under the control of the OS, which protects its CSPs against unauthorized disclosure, modification, and substitution. The module only allows access to CSPs through its well-defined API.

2.7 Cryptographic Key Management

When loaded into memory, the module operates in the FIPS-Approved mode. In this mode, the module implements the FIPS-Approved algorithms listed in Table 5 below.

Table 5 – FIPS-Approved Algorithm Implementations

Algorithm	Certificate Number
AES ECB, CTR encryption/decryption with 256-bit keys	3772
AES GCM encryption/decryption and message authentication with 256-bit keys[19]	3772
RSA key-pair Generation with 2048- and 3072-bit keys	1942
RSA signature generation (PSS[20]) with 2048- and 3072-bit keys	1942
RSA signature verification (PSS) with 2048- and 3072- bit keys	1942
SHA-256 and SHA-512	3142
HMAC[21] with SHA-256 and SHA-512	2472
SP[22] 800-90A HMAC DRBG[23],[24]	1042
SHA-256 (DRBG implementation)	3200
HMAC with SHA-256 (DRBG implementation)	2520
PBKDF2	Vendor Affirmed

The module performs PBKDF2 as defined in IETF[25] RFC[26] #2898. The vendor affirms compliance with SP 800-132, using option 1(a) in Section 5.4 to derive the Data Protection Key (DPK). The PBKDF2 is used for storage applications only.

The module employs the following non-FIPS-Approved algorithm implementations allowed for use in FIPS mode:

- RSA (key wrapping; key establishment methodology provides 112 or 128 bits of encryption strength.)
- NDRNG[27] – for seeding the FIPS-approved DRBG

[18] CPU – Central Processing Unit
[19] 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 as required by IG A.5.
[20] PSS – Probabilistic Signature Scheme
[21] HMAC – (keyed-) Hash Message Authentication Code
[22] SP – Special Publication
[23] DRBG – Deterministic Random Bit Generator
[24] The Ionic FCM implements FIPS 140-2 Implementation Guidance 7.8 scenario 1, no post processing of the random number before key generation
[25] IETF – Internal Engineering Task Force
[26] RFC – Request for Comment
[27] NDRNG – Non-Deterministic Random Number Generator

The module employs the following non-FIPS-Approved algorithm implementations that are not allowed for use in FIPS mode:

- RSA (key wrapping; key establishment methodology less than 112 bits of encryption strength.)

Caveat: The module generates cryptographic keys whose strengths are modified by the available entropy.

The module supports the CSPs listed below in Table 6.

Table 6 – List of Cryptographic Keys, Cryptographic Key Components, and CSPs

CSP	CSP Type	Generation / Input	Output	Storage	Zeroization	Use
AES key	AES-CTR 256-bit key	Internally Generated via approved DRBG; Input via module API in plaintext through GPC INT Path[28]	Output in plaintext via GPC INT Path	Not persistently stored by the module	Restart host application, Cycle host system power, and API call	Encryption, decryption
AES key	AES-ECB 256-bit key	Internally Generated via approved DRBG; Input via module API in plaintext through GPC INT Path[29]	Output in plaintext via GPC INT Path	Not persistently stored by the module	Restart host application, Cycle host system power, and API call	Encryption, decryption
AES GCM key	AES 256-bit key	Internally Generated via approved DRBG; Input via module API in plaintext through GPC INT Path	Output in plaintext via GPC INT Path	Not persistently stored by the module	Restart host application, Cycle host system power, and API call	Encryption, decryption, and message authentication

[28] GPC INT Path is defined in Implementation Guidance Section 7.7
[29] GPC INT Path is defined in Implementation Guidance Section 7.7

CSP	CSP Type	Generation / Input	Output	Storage	Zeroization	Use
AES GCM IV	128-bit IV	Internally Generated via approved DRBG in compliance with Section 8.2.1 of NIST SP 800-38D[30]	Output in plaintext via GPC INT Path	Not persistently stored by the module	Restart host application, Cycle host system power, and API call	IV input for AES-GCM key
NIST SP 800-90A HMAC DRBG seed	DRBG seed (minimum 256-bits)	Internally generated using entropy, nonce and personalization string. An additional 32-bit entropy sting can be added to the initial seed via the module API.	Does not exit the module	Not persistently stored by the module	Restart host application, Cycle host system power, and API call	Seeding material for SP 800-90A HMAC DRBG
NIST SP 800-90A HMAC DRBG entropy input string	256-bit random value	Generated by the hardware NDRNG[31]	Does not exit the module	Not persistently stored by the module	Restart host application, Cycle host system power, and API call	Entropy material for SP 800-90A HMAC DRBG
NIST SP 800-90A HMAC DRBG 'V' Value	Internal state value	Internally Generated	Does not exit the module	Not persistently stored in the module	Restart host application, Cycle host system power, and API call	Internal state value for SP 800-90A HMAC DRBG
NIST SP 800-90A HMAC DRBG 'Key' Value	Internal state value	Internally Generated	Does not exit the module	Not persistently stored in the module	Restart host application, Cycle host system power, and API call	Internal value for SP 800-90A HMAC DRBG

[30] AES GCM IV generation is in compliance with IG A.5 Scenario 2. The AES GCM IV is generated internally by an approved DRBG. The DRBG is generated internally from a cryptographically strong source. The IV length is 128-bits long, which exceeds the minimum required by NIST SP 800 38D.
[31] Per IG 7.14 scenario 1(b), the FCM issues a GET command to obtain a 256-bit random value entropy input string from outside the module's logical boundary.

CSP	CSP Type	Generation / Input	Output	Storage	Zeroization	Use
RSA private key	RSA 2048 and 3072 -bit key	Internally Generated via approved DRBG; Input via module API in plaintext through GPC INT Path	Output in plaintext via GPC INT Path	Not persistently stored by the module	Restart host application, Cycle host system power, and API call	Signature generation, decryption
RSA public key	RSA 2048, and 3072-bit key	Internally Generated via approved DRBG; Input via module API in plaintext through GPC INT Path	Output in plaintext via GPC INT Path	Not persistently stored by the module	Restart host application, Cycle host system power, and API call	Signature verification, encryption
HMAC key	256-bit or 512-bit HMAC Key	Internally Generated via approved DRBG; Input via module API in plaintext through GPC INT Path	None	Not persistently stored by the module	Restart host application, Cycle host system power, and API call	Used for message authentication
HMAC key	256-bit HMAC Key	Hardcoded within the module	None	Stored within the module	NA	Used for message authentication
HMAC key (DBRG implementation)	256-bit HMAC Key	Internally Generated via approved DRBG; Input via module API in plaintext through GPC INT Path	None	Not persistently stored by the module	Restart host application, Cycle host system power, and API call	Used for message authentication

CSP	CSP Type	Generation / Input	Output	Storage	Zeroization	Use
Data Protection Key	Maximum block size of 2^32 bytes	Generated internally via PBKDF2	Does not exit the module.	Stored in plaintext in volatile memory	Restart host application, Cycle host system power, and API call	Used for storage applications
PBKDF2 password	8 bytes of random data	Input in plaintext via API parameter	Does not exit the module	Not persistently stored by the module	Restart host application, Cycle host system power, and API call	Password input to PBKDF2 function

2.8 Self-Tests

Cryptographic self-tests are performed by the module when the module is first powered up and loaded into memory as well as when a random number or asymmetric key pair is created. The following sections list the self-tests performed by the module, their expected error status, and error resolutions.

2.8.1 Power-Up Self-Tests

Power-up self-tests are automatically performed by the module before performing any other operation. Power-up self-tests can be run manually by shutting down and reloading the module. During the execution of self-tests, data output from the module is inhibited.

If any self-tests fail, the module will set a global static error flag and will enter a critical error state. After entering an error state, all subsequent calls to the module will be rejected, ensuring that data output from the module is inhibited. In order to resolve a cryptographic self-test error, the module must be restarted by restarting the host system. If the error persists, the application must be reinstalled.

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

- Software integrity check using HMAC SHA-256 digest verification
- Known Answer Tests (KATs)
    - AES-CTR Encrypt KAT
    - AES-CTR Decrypt KAT
    - AES-ECB Encrypt KAT
    - AES-ECB Decrypt KAT
    - AES-GCM Encrypt KAT
    - AES-GCM Decrypt KAT
    - SHA-256 KAT
    - SHA-512 KAT
    - HMAC SHA-256 KAT
    - HMAC SHA-512 KAT
    - RSA Signature Generation KAT
    - RSA Signature Verification KAT
    - HMAC SHA-256 KAT (DRBG implementation)
    - NIST SP800-90A HMAC DRBG KAT

2.8.2 Conditional Self-Tests

Conditional self-tests are performed by the module whenever a new random number or asymmetric key pair is generated. If an error is encountered, the module will set a global static error flag and will enter a critical error state. After entering an error state, all subsequent calls to the module will be rejected, ensuring that data output from the module is inhibited. In order to resolve a cryptographic self-test error, the module must be restarted by shutting down and reloading the module. If the error persists, the application must be reinstalled.

The FCM performs the following conditional self-tests:

- Continuous RNG Test for the SP 800-90A HMAC DRBG
- Continuous RNG Test for NDRNG
- RSA Pairwise Consistency Test for Key Generation

2.8.3 Critical Functions Self-Tests

The SP 800-90A HMAC DRBG employed by the cryptographic module includes the following critical functions: instantiation, generation, reseed, and uninstantiation. Each function is tested by the module during the module's power-up self-tests. If any of the self-tests fail, the module will set a global static error flag and will enter a critical error state. After entering an error state, all subsequent calls to the module will be rejected, ensuring that data output from the module is inhibited. In order to resolve a cryptographic self-test error, the module must be restarted by restarting the host application. If the error persists, the application must be reinstalled.

The FCM performs the following critical functions tests:

- DRBG Instantiate Critical Function Test
- DRBG Generate Critical Function Test
- DRBG Reseed Critical Function Test
- DRBG Uninstantiate Critical Function Test

2.9 Mitigation of Other Attacks

This section is not applicable. The module does not claim to mitigate any other attacks.

3. Secure Operation

The FIPS Crypto Module meets Level 1 requirements for FIPS 140-2.

3.1 Secure Management

There are no special instructions for installing the product for the FIPS-Approved mode of operation. The product must be installed in at least one of the tested and validated operational environments:

- HP Z230 desktop with an Intel Core 17 processor running Windows 7 OS
- Intel Server System R1304GZ4GC running an Intel Xeon E5 processor running a CentOS 7.1

The FCM is configured to operate in the FIPS-Approved mode of operation.

3.1.1 Secure Initialization

The module sits at rest until initiated through an API call by the CO. Upon initialization, the module performs a suite of power-up self-tests to ensure the integrity and correct operation of the cryptographic services. Power-up self-tests are performed when the API calls the CryptoImpl_Initialize() function within the CryptoAbstract.cpp. On Windows, DLLMain() calls cryptoImpl_initialize() automatically when the DLL is loaded into a process. DLLMain() is called automatically by Windows any time a DLL is loaded into a process. On CentOS, a static C++ object (CryptoAbstractAutoInitializer) is loaded and its constructor calls cryptoImpl_initialize(). All tests must pass in order for the module to operate in the FIPS-Approved mode.

3.1.2 Zeroization

During shutdown, reboot, or through a specific API call, the FCM will perform a shutdown function. This function zeroizes all keys and releases all resources allocated for operation of the module.

3.2 User Guidance

The user shall adhere to the guidelines of this Security Policy. Per guidance of this Security Policy, the User does not have any ability to install or configure the module. Operators in the User role are able to perform the services listed in Table 4 above. Users are responsible for reporting any irregular activity they notice to the CO. During operation, the User may check the status of the module by attempting to run any User service. If the service executes, the module is operating in FIPS-Approved mode.

The user, through the API, passes in the PBKDF2 password. The password must be 8 bytes in length and a combination of upper-case and lower-case letters (52 letters) and number, 0 – 9 (10 numbers). There are sixty-two different letters and numbers that can be used, in any order. With this combination of letters and numbers, the probability of guessing this password is 1:62^8 or 1:$2.18x10^{14}$. The key derived by the PBKDF2 is used solely for storage purposes.

4. Acronyms

Table 7 provides definitions for the acronyms used in this document.

Table 7 – Acronyms

Acronym	Definition
AES	Advanced Encryption System
API	Application Programming Interface
BYOD	Bring Your Own Device
CentOS	Community Enterprise Operating System
CMVP	Cryptographic Module Validation Program
CO	Cryptographic Officer
CPU	Central Processing Unit
CSE	Communications Security Establishment
CSP	Critical Security Parameter
CTR	Counter
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FIPS	Federal Information Processing Standard
FCM	FIPS Crypto Module
GCM	Galois/Counter Mode
GPC	General Purpose Computer
GUI	Graphical User Interface
HMAC	(keyed-) Hash Message Authentication Code
HP	Hewlett-Packard
IETF	Internal Engineering Task Force
KAAP	Key Management, Authentication, Audit, and Policy Management
KAT	Known Answer Test
LCD	Liquid Crystal Display
LED	Light-emitting Diode
MAC	Message Authentication Code
NDRNG	Non-Deterministic Random Number Generator
NIST	National Institute of Standards and Technology
OS	Operating System
PBKDF2	Password-Based Key Derivation Function 2

Acronym	Definition
PSS	Probabilistic Signature Scheme
RAM	Random Access Memory
RFC	Request for Comment
RNG	Random Number Generator
RSA	Rivest Shamir and Adleman
SHA	Secure Hash Algorithm
SP	Special Publication
USB	Universal Serial Bus

Prepared by:
Corsec Security, Inc.

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

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