OpenPeak, Inc.

Cryptographic Security Module

Software Version: 1.0

FIPS 140-2 Non-Proprietary Security Policy

FIPS Security Level: 1
Document Version: 0.9

Prepared for:

OpenPeak, Inc.
1750 Clint Moore Road
Boca Raton, FL 33487
United States of America

Phone: +1 561 893 7800
Email: info@openpeak.com
http://www.openpeak.com

Prepared by:

Corsec Security, Inc.
13135 Lee Jackson Memorial Hwy., Suite 220
Fairfax, VA 22033
United States of America

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

Table of Contents

1 INTRODUCTION 4
1.1 PURPOSE 4
1.2 REFERENCES 4
1.3 DOCUMENT ORGANIZATION 4

2 CRYPTOGRAPHIC SECURITY MODULE 5
2.1 OVERVIEW 5
2.1.1 SANCTION 5
2.1.2 SECTOR 6
2.1.3 OPENSHOP 6
2.2 MODULE SPECIFICATION 7
2.2.1 Physical Cryptographic Boundary 9
2.2.2 Logical Cryptographic Boundary 11
2.3 MODULE INTERFACES 13
2.4 ROLES, SERVICES, AND AUTHENTICATION 14
2.4.1 Crypto Officer Role 15
2.4.2 User Role 17
2.5 PHYSICAL SECURITY 19
2.6 OPERATIONAL ENVIRONMENT 19
2.7 CRYPTOGRAPHIC KEY MANAGEMENT 20
2.8 EMC/EMI 28
2.9 SELF-TESTS 28
2.9.1 Power-Up Self-Tests 28
2.9.2 Conditional Self-Tests 29
2.9.3 Critical Functions Self-Tests 29
2.10 MITIGATION OF OTHER ATTACKS 29

3 SECURE OPERATION 30
3.1 SECURE MANAGEMENT 30
3.1.1 Initialization 30
3.1.2 Management 30
3.1.3 Zeroization 30
3.2 USER GUIDANCE 31

4 ACRONYMS 32

Table of Figures

FIGURE 1 ENTERPRISE ADAM DEPLOYMENT AND DATA FLOW 5
FIGURE 2 HARDWARE PLATFORM PHYSICAL BLOCK DIAGRAM – CONFIGURATION #1 10
FIGURE 3 HARDWARE PLATFORM PHYSICAL BLOCK DIAGRAM – CONFIGURATION #2 11
FIGURE 4 LOGICAL BLOCK DIAGRAM FOR MODULE RUNNING ON HARDWARE PLATFORM – CONFIGURATION #1 12
FIGURE 5 LOGICAL BLOCK DIAGRAM FOR MODULE RUNNING ON HARDWARE PLATFORM – CONFIGURATION #2 (WITH IOS) 12
FIGURE 6 LOGICAL BLOCK DIAGRAM FOR MODULE RUNNING ON HARDWARE PLATFORM – CONFIGURATION #2 (WITH ANDROID OS) 13

List of Tables

TABLE 1 SECURITY LEVEL PER FIPS 140-2 SECTION 7
TABLE 2 MAPPING OF FIPS 140-2 LOGICAL INTERFACES TO CRYPTOGRAPHIC SECURITY MODULE INTERFACES ON THE HARDWARE PLATFORM – CONFIGURATION #1 14

TABLE 3 MAPPING OF FIPS 140-2 LOGICAL INTERFACES TO CRYPTOGRAPHIC SECURITY MODULE INTERFACES ON THE HARDWARE PLATFORM – CONFIGURATION #2 14
TABLE 4 MAPPING OF CO SERVICES TO INPUTS, OUTPUTS, CSPS, AND TYPE OF ACCESS 15
TABLE 5 MAPPING OF USER SERVICES TO INPUTS, OUTPUTS, CSPS, AND TYPE OF ACCESS 17
TABLE 6 FIPS-APPROVED ALGORITHM IMPLEMENTATIONS 20
TABLE 7 LIST OF CRYPTOGRAPHIC KEYS, CRYPTOGRAPHIC KEY COMPONENTS, AND CSPS 23
TABLE 8 ACRONYMS 32

1 Introduction

1.1 Purpose

This is a non-proprietary Cryptographic Module Security Policy for the Cryptographic Security Module from OpenPeak, Inc. This Security Policy describes how the Cryptographic Security 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 Canada (CSEC) Cryptographic Module Validation Program (CMVP) website at http://csrc.nist.gov/groups/STM/cmvp.

This document also describes how to run the module in a secure FIPS-Approved mode of operation. This policy was prepared as part of the Level 1 FIPS 140-2 validation of the module. The Cryptographic Security Module is referred to in this document as OpenPeak Cryptographic Security Module, 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 OpenPeak website (www.openpeak.com) contains information on the full line of products from OpenPeak.
- 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
- Other supporting documentation as additional references

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

2 Cryptographic Security Module

2.1 Overview

OpenPeak's Advanced Device and Application Management (ADAM™) solution provides a comprehensive mobile device and application management solution hosted on-site or as a cloud-based service. ADAM is composed of three separate components: SANCTION, SECTOR, and OPENSHOP. Each component may be deployed individually to complement pre-existing device and network management configurations, or together as a complete end-to-end Device, Application, and Profile Management solution.

The three components of ADAM and their interactions are shown Figure 1 and are described in the sections that follow.

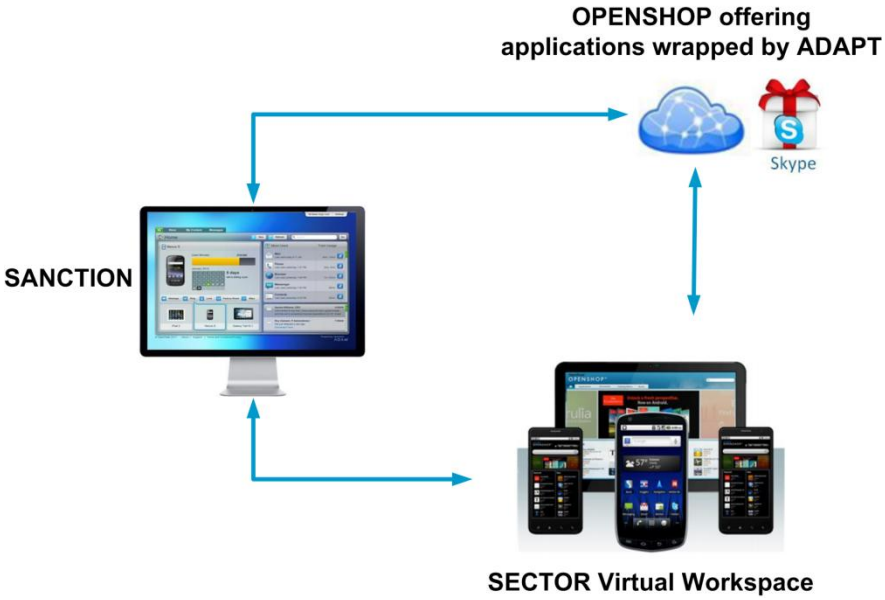

Figure 1 Enterprise ADAM Deployment and Data Flow

2.1.1 SANCTION

The SANCTION component of ADAM is a cross-platform mobile device management (MDM) solution presented to users in a web graphical user interface (GUI). SANCTION provides administrators a cryptographically secure management connection to the web GUI. The SANCTION web GUI provides IT administrators the ability to:

- Provision devices
- Manage inventory
- Control network access
- Require minimum security settings
- Set policies
- Protect corporate data

2.1.2 SECTOR

The SECTOR component of ADAM is an innovative solution allowing enterprises to create a virtual enterprise workspace on a wide variety of mobile devices. With SECTOR, all enterprise data is secured in the enterprise workspace (referred to as the Work Sector) using AES[1] encryption, and the enterprise's IT[2] department has full control over the enterprise workspace. The IT department can control the applications allowed in the Work Sector, set password criteria, lock the Work Sector, enable and disable applications, and completely wipe the Work Sector if the device is lost or an employee leaves the enterprise. SECTOR provides the following capabilities:

- Centralization of secure applications into a common workspace
- Instant switching between workspaces
- Shared normal shortcuts, folders, and widgets in the managed workspace without affecting enterprise workspace security
- Customization of wallpapers and layout for secure workspace
- Addition of secure widgets and shortcuts only to the secure workspace

SECTOR also allows the enterprise to easily add their own custom-built applications to the Work Sector, ensuring that all of enterprise application data is secure. SECTOR currently supports both Android and iOS (iPad/iPhone) platforms. SECTOR makes creating secure applications a simple, quick experience through a proprietary process called ADAPT™. ADAPT automatically wraps and secures unsecure applications for manageability and deployment to enterprises via OPENSHOP and SECTOR. The ADAPT process adds enterprise level security to each application through its ability to:

- Enable/disable use of external storage
- Provide data-at-rest encryption for applications wrapped by ADAPT (encrypted using AES 256-bit)
- Limit secure data sharing to only applications that have been wrapped by ADAPT can share information
- Restrict copy and paste to prevent content from wrapped applications being pasted to unwrapped applications
- Provide Jailbreak/Root detection. This is event is reported in audit records and will be acted on according to policy
- Create application-specific VPN to provide encryption for data-in-motion through IPSec
- Add network access control restrictions based on policy

2.1.3 OPENSHOP

The OPENSHOP component of ADAM is an enterprise storefront allowing enterprises to manage and distribute both home-grown content as well as content provided by third parties that have been wrapped using the ADAPT process. OPENSHOP is platform and device agnostic and can distribute applications and content to a wide variety of devices. It also provides organizations with new application licensing models, giving them greater flexibility in deploying applications.

OPENSHOP presents complete application lifecycle management. Developers can upload applications for testing; enterprise testers pre-view, test, and approve submitted applications; and enterprise applications administrators make approved applications available to the storefront. Life cycle management also includes versioning, device targets, application revocation, revenue sharing as well as extensive business intelligence facilities.

The Cryptographic Security Module is validated at the FIPS 140-2 Section levels shown in Table 1:

[1] AES – Advanced Encryption Standard
[2] IT – Information Technology

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[3]	1
9	Self-tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A

2.2 Module Specification

The Cryptographic Security Module is a software module with a multi-chip standalone embodiment. The overall security level of the module is 1. The Cryptographic Security Module is used by calling applications to provide symmetric/asymmetric cipher operation, signature generation/verification, hashing, cryptographic key generation, random number generation, and message authentication functions. The cryptographic boundary of the Cryptographic Security Module consists of the fipscanister.o object file that is linked with the ADAM solution.

For the purpose of FIPS 140-2 conformance testing, the module was tested and found compliant on the following platforms and environments:

- Ubuntu 12.04 running an Intel Xeon on a Dell PowerEdge T110
- Ubuntu 12.04 running an Intel Xeon on ESXi 5.1 on a Dell PowerEdge T110
- Ubuntu 12.04 running an AMD Opteron on a SuperMicro AS-1011S-mR2
- Ubuntu 12.04 running an AMD Opteron on ESXi 5.1 on a SuperMicro AS-1011S-mR2
- iOS v5 running on an ARMv7-based Apple A5 processor on an iPad3
- iOS v6 running on an ARMv7s-based Apple A6 processor on an iPhone5
- Android v4.1 running on an ARMv7-based Qualcomm Snapdragon processor on a Samsung Galaxy SIII

While no claim can be made as to the correct operation of the module or the security strengths of the generated keys when ported to an operational environment which is not listed on the validation certificate, the vendor affirms that the module remains FIPS-compliant when executed on any of the supported platforms and environments, including the following:

- Android v2.2 running on an ARMv7-based Qualcomm QSD 8250 processor on an HTC Desire
- Android v2.2 running on an ARMv7-based Qualcomm QSD 8250 processor on a Dell Streak
- Microsoft Windows 7 32-bit running on an Intel Celeron (x86) processor
- uClinux 0.9.29 running on an ARMv4-based ARM 922T processor
- Fedora 14 running on an Intel Core i5 (x86) processor (with AES-NI support)

[3] EMI/EMC – Electromagnetic Interference / Electromagnetic Compatibility

- HP-UX 11i (hpux-ia64-cc, 32 bit mode) on an Intel Itanium 2 (IA64) processor
- HP-UX 11i (hpux64-ia64-cc, 64 bit mode) on an Intel Itanium 2 (IA64) processor
- Ubuntu 10.04 running on an Intel Pentium T4200 (x86) processor
- Android 3.0 running on an ARMv7-based NVIDIA Tegra 250 T20 processor
- Linux 2.6.27 running on a PowerPC e300c3 (PPC) processor
- Microsoft Windows 7 64 bit running on an Intel Pentium 4 (x86) processor
- Ubuntu 10.04 running on an Intel Core i5 (x86) processor (with AES-NI support)
- Linux 2.6.33 running on a PowerPC32 e300 (PPC) processor
- Android 2.2 running on an ARMv7-based OMAP 3530 processor
- VxWorks 6.8 running on a TI TNETV1050 (MIPS) processor
- Linux 2.6 running on an ARMv4-based TI TMS320DM6446 processor
- Linux 2.6.32 running on an ARMv7-based TI AM3703CBP processor
- Solaris 10 32 bit running on a SPARC-T3 (SPARCv9) processor
- Solaris 10 64 bit running on a SPARC-T3 (SPARCv9) processor
- Solaris 11 32 bit running on a SPARC-T3 (SPARCv9) processor
- Solaris 11 64 bit running on a SPARC-T3 (SPARCv9) processor
- Solaris 11 32 bit running on an Intel Xeon 5675 (x86) processor
- Solaris 11 64 bit running on an Intel Xeon 5675 (x86) processor
- Solaris 11 32 bit running on an Intel Xeon 5675 (x86) processor (with AES-NI support)
- Solaris 11 64 bit running on an Intel Xeon 5675 (x86) processor (with AES-NI support)
- Oracle Linux 5 64 bit running on an Intel Xeon 5675 (x86) processor
- CascadeOS 6.1 32 bit running on an Intel Pentium T4200 (x86) processor
- CascadeOS 6.1 64 bit running on an Intel Pentium T4200 (x86) processor
- Ubuntu 10.04 32 bit running on an Intel Pentium T4200 (x86) processor
- Ubuntu 10.04 64 bit running on an Intel Pentium T4200 (x86) processor
- Oracle Linux 5 running on an Intel Xeon 5675 (x86) processor (with AES-NI support)
- Oracle Linux 6 running on an Intel Xeon 5675 (x86) processor
- Oracle Linux 6 running on an Intel Xeon 5675 (x86) processor (with AES-NI support)
- Android 4.0 running on an ARMv7-based NVIDIA Tegra 250 T20 processor
- Linux 2.6 running on a Freescale PowerPC-e500 processor
- Apple iOS 5.1 running on an ARMv7-based processor
- WinCE 6.0 running on an ARMv5- TEJ processor
- WinCE 5.0 running on an ARMv7-based processor
- Android 4.0 running on an OMAP processor
- NetBSD 5.1 running on a PowerPC-e500 processor
- NetBSD 5.1 running on an Intel Xeon 550 (x86) processor
- Windows 2008 32-bit under vSphere running on an Intel Xeon E3-1220v2 (x86) processor
- Windows 2008 64-bit under vSphere running on an Intel Xeon E3-1220v2 (x86) processor
- RHEL 6 32-bit under vSphere running on an Intel Xeon E3-1220v2 (x86) processor
- RHEL 6 64-bit under vSphere running on an Intel Xeon E3-1220v2 (x86) processor
- Windows 7 64-bit running on an Intel Core i5-2430M (x86) processor (with AES-NI support)
- Android 4.1 running on an ARMv7-based T1 DM3730 processor
- Android 4.1 running on an ARMv7-based T1 DM3730 processor
- Android 4.2 running on an ARMv7-based Nvidia Tegra 3 processor
- Android 4.2 running on an ARMv7-based Nvidia Tegra 3 processor
- Windows Embedded Compact 7 running on an ARMv7-based Freescale i.MX53xA processor
- Windows Embedded Compact 7 running on an ARMv7-based Freescale i.MX53xD processor
- Android 4.0 running on an ARMv7-based Qualcomm Snapdragon APQ8060 processor
- VMware Horizon Mobile 1.3 under VMware under Android 4.0 running on an ARMv7-based qualcomm MXM8X60 processor
- Apple OS X 10.7 running on an Intel Core i7-3615QM processor
- Apple iOS 5.0 running on an ARMv7-based ARM Cortex A8 processor.

The Cryptographic Security Module is defined as a software cryptographic module and therefore has a logical boundary in addition to a physical boundary. The physical and logical boundaries are outlined in section 2.2.1 and 2.2.2 respectively.

2.2.1 Physical Cryptographic Boundary

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 hardware platform can exist in one of two configurations:

- The hardware platform consists of a motherboard, a Central Processing Unit (CPU), random access memory (RAM), read-only memory (ROM), hard disk(s), hardware case, power supply, and fans. Other devices may be attached to the hardware appliance such as a monitor, keyboard, mouse, floppy drive, DVD drive, fixed disk drive, printer, video adapter, audio adapter, or network adapter. In the validated configuration, the processor is either an Intel or AMD server-class processor.

- The hardware platform consists of a motherboard, a CPU, RAM, ROM, flash memory, mobile device case, battery, and heatsink. In the validated configuration, the processor is an ARMv7-based or ARMv7s-based processor.

Please see Figure 2 and Figure 3 for the two hardware platform configurations.

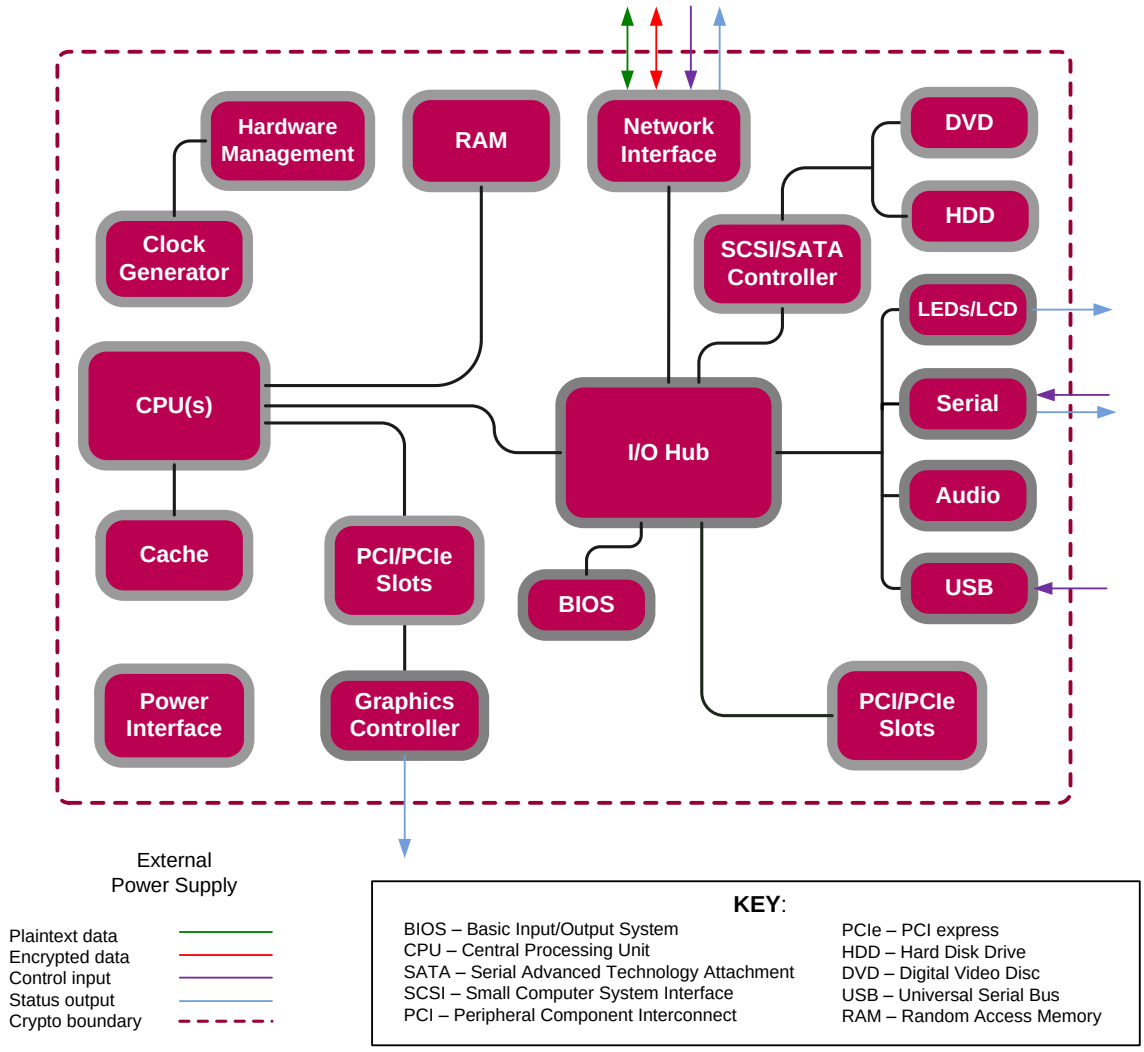

Figure 2 Hardware Platform Physical Block Diagram – Configuration #1

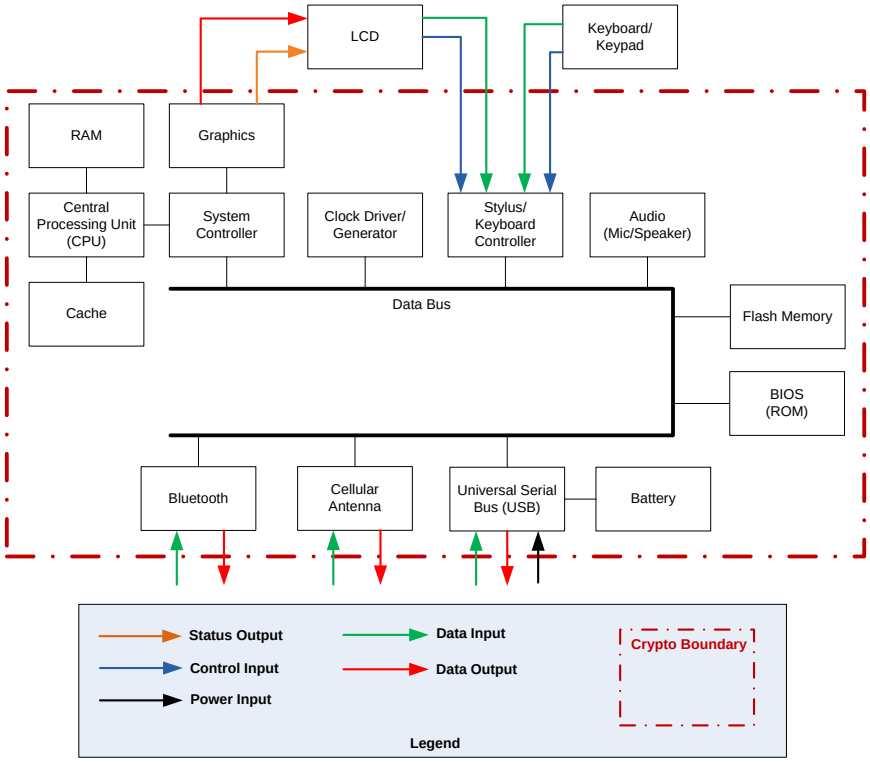

Figure 3 Hardware Platform Physical Block Diagram – Configuration #2

2.2.2 Logical Cryptographic Boundary

The module is a cryptographic object module that provides cryptographic services for the SANCTION, SECTOR, and OPENSHOP software components that make up the ADAM solution. In this document, those applications will be referred to collectively as the "calling application". The module is used by the calling application to provide symmetric and asymmetric cipher operation, signature generation and verification, hashing, cryptographic key generation, random number generation, message authentication functions, and secure key agreement/key exchange protocols. The module is entirely contained within the physical cryptographic boundary described in Figure 2 and Figure 3.

When the module is executing on Ubuntu 12.04 with an Intel Xeon or AMD Opteron processor, the logical cryptographic boundary is shown below in Figure 4.

When the module is executing on iOS v5 with an ARMv7-based Apple A5 processor or iOS v6 with an ARMv7s-based Apple A6 processor, the logical cryptographic boundary is shown below in Figure 5.

When the module is executing on Android v4.1 with an ARMv7-based Qualcomm Snapdragon based processor, the logical cryptographic boundary is shown below in Figure 6.

Figure 4, Figure 5, and Figure 6 show logical block diagrams of the module executing in memory and its interactions with surrounding software components, as well as the module's logical cryptographic boundary.

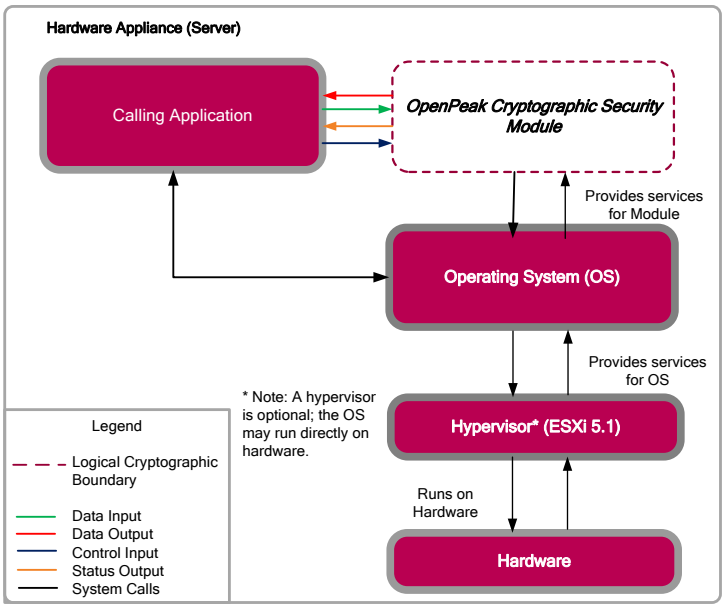

Figure 4 Logical Block Diagram for Module Running on Hardware Platform – Configuration #1

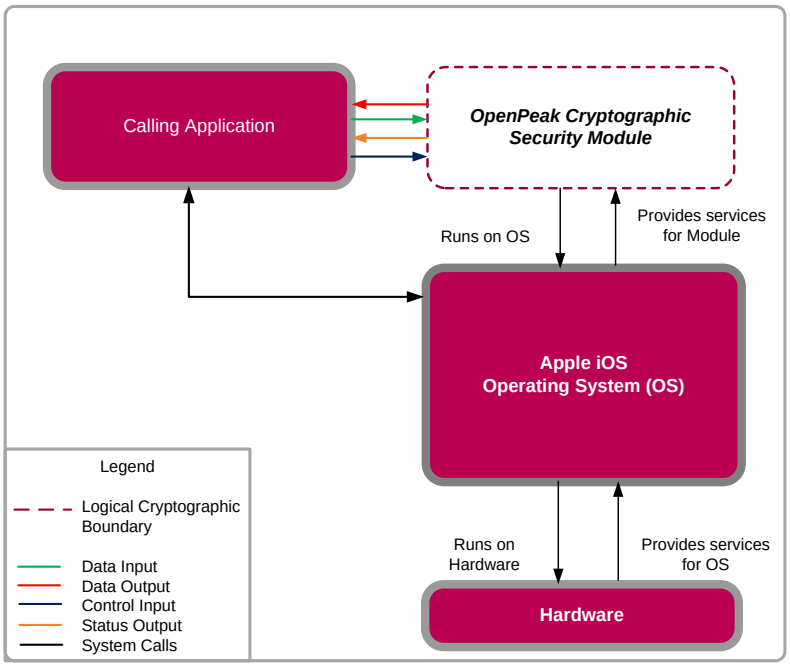

Figure 5 Logical Block Diagram for Module Running on Hardware Platform – Configuration #2 (with iOS)

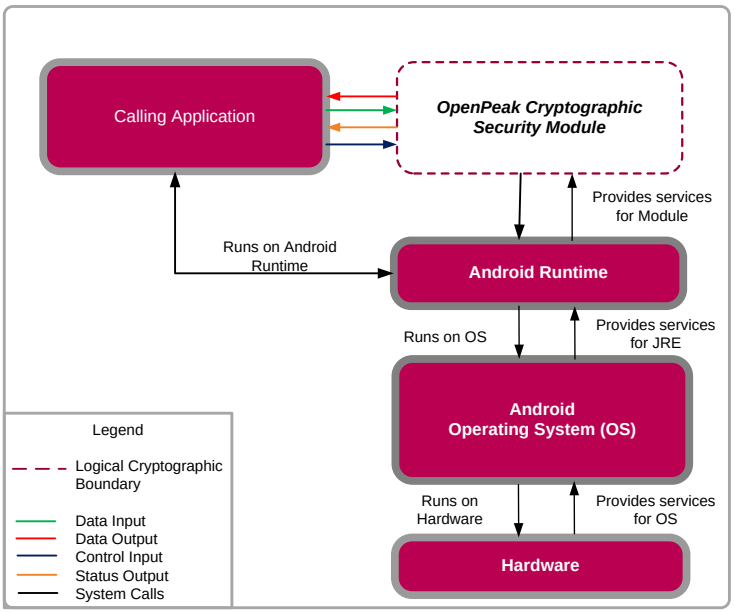

Figure 6 Logical Block Diagram for Module Running on Hardware Platform – Configuration #2 (with Android OS)

2.3 Module Interfaces

The module isolates communications to logical interfaces that are defined in the software as an API[4]. The API interface is mapped to the following four logical interfaces:

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

The module features the physical ports of a host system. Table 2 and Table 3 list all of the physical interfaces implemented and the module's associated logical interfaces.

The module's manual controls, physical indicators, and physical, logical, and electrical characteristics are those of the host system. The platform hardware manufacturers can provide documentation describing their controls, indicators, and characteristics. The module's logical interfaces are at a lower level in the software. The physical data and control input through physical mechanisms is translated into the logical data and control inputs for the software module. All of these physical interfaces are separated into logical interfaces defined by FIPS 140-2, as described in the following Table 2 and Table 3.

[4] API – Application Programming Interface

Table 2 Mapping of FIPS 140-2 Logical Interfaces to Cryptographic Security Module Interfaces on the Hardware Platform – Configuration #1

FIPS 140-2 Interface	Physical Interface	Module Interface (API)
Data Input	Network port, Serial port, USB[5] port	Function calls that accept, as their arguments, data to be used or processed by the module
Data Output	Network port, Serial port, USB port	Arguments for a function call that specify where the result of the function is stored
Control Input	Network port, Serial port, USB port, Power button	Function calls utilized to initiate the module and the function calls used to control the operation of the module.
Status Output	Network port, Serial port, USB port, Graphics controller, LEDs/LCD[6], Audio port	Thrown exceptions for function calls
Power Input	AC Power socket	Not applicable

Table 3 Mapping of FIPS 140-2 Logical Interfaces to Cryptographic Security Module Interfaces on the Hardware Platform – Configuration #2

FIPS 140-2 Interface	Physical Interface	Module Interface (API)
Data Input	USB port, Cellular Antenna, Bluetooth	Function calls that accept, as their arguments, data to be used or processed by the module
Data Output	USB port, Cellular Antenna, Bluetooth	Arguments for a function that specify where the result of the function is stored
Control Input	USB port, LCD Touchscreen, Keyboard/Keypad	Function calls utilized to initiate the module and the function calls used to control the operation of the module.
Status Output	USB port, LCD screen	Return status for function calls
Power Input	USB port	Not applicable

2.4 Roles, Services, and Authentication

The module supports two roles (as required by FIPS 140-2) that operators may implicitly assume: a Crypto Officer role and a User role. The module does not allow multiple concurrent operators in the FIPS-Approved mode of operation. Per section 6.1 of the NIST FIPS 140-2 Implementation Guidance, the calling application that loaded the module is the only operator.

[5] USB – Universal Serial Bus
[6] LCD – Liquid-Crystal Display

2.4.1 Crypto Officer Role

The Crypto-Officer role has access to all of the services of the module and is responsible for the installation of the module on the host device and placing the module into the Approved mode of operation. The module does not allow multiple concurrent operators in the Approved mode of operation. Per section 6.1 of the NIST FIPS 140-2 Implementation Guidance, the calling application that loaded the module is the only operator. Descriptions of the services available to the Crypto-Officer role are provided in Table 4.

Note 1: The following definitions are used for "CSP and Type of Access" column in Table 4:

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

Table 4 Mapping of CO Services to Inputs, Outputs, CSPs, and Type of Access

Service	Description	Input	Output	CSP and Type of Access
Initialize module	Place the module into FIPS-mode, Perform integrity check and power-up self-tests	API call parameters	Status	Integrity check HMAC[7] key – RX
Run self-test on demand	Performs power-up self-tests	API call parameters	Status	None
Show status[8]	Returns the current mode of the module	None	Status	None
Zeroize keys	Zeroizes and de-allocates memory containing sensitive data	Power cycle; unload module	None	AES[9] key – W AES CMAC[10] Key –W Triple-DES[11] key – W Triple-DES CMAC Key – W HMAC key – W RSA[12] private/public key – W DSA[13] private/public key – W EC[14]DSA private/public key – W EC DH[15] public/private keys – W DRBG[16] Seed – W DRBG Entropy – W DRBG 'C' value – W DRBG 'V' value – W ANSI[17] X9.31 RNG seed – W ANSI X9.31 seed key – W

[7] HMAC – (Keyed-)Hash Message Authentication Code
[8] Running the 'show status' command on the module will, along with its operating status, return the version of the module. The value returned by the module will be version 2.0.2. This version is equivalent to version 1.0 of the OpenPeak Cryptographic Security Module.
[9] AES – Advanced Encryption Standard
[10] CMAC – Cipher-based Message Authentication Code
[11] DES – Data Encryption Standard
[12] RSA – Rivest, Shamir, Adleman
[13] DSA – Digital Signature Algorithm
[14] EC – Elliptic Curve
[15] DH – Diffie-Hellman

Service	Description	Input	Output	CSP and Type of Access
Generate random number	Returns the specified number of random bits to the calling application	API call parameters	Status, random bits	DRBG Seed – WRX DRBG Entropy – RX DRBG 'C' value – WRX DRBG 'V' value – WRX ANSI X9.31 RNG seed – WRX ANSI X9.31 seed key – RX
Generate message digest	Compute and return a message digest using SHS algorithms	API call parameters, message	Status, hash	None
Generate keyed hash (HMAC)	Compute and return a message authentication code	API call parameters, key, message	Status, hash	HMAC key – RX
Generate Cipher Hash (CMAC)	Compute and return a cipher message authentication code	API call parameters, key, message	Status, hash	AES CMAC Key – RX Triple-DES CMAC Key – RX
Generate symmetric key	Generate and return the specified type of symmetric key (Triple-DES or AES)	API call parameters	Status, key	AES key – W Triple-DES Key – W
Symmetric encryption	Encrypt plaintext using supplied key and algorithm specification (Triple-DES or AES)	API call parameters, key, plaintext	Status, ciphertext	AES key – RX Triple-DES key – RX
Symmetric decryption	Decrypt ciphertext using supplied key and algorithm specification (Triple-DES or AES)	API call parameters, key, ciphertext	Status, plaintext	AES key – RX Triple-DES key – RX
Generate asymmetric key pair	Generate and return the specified type of asymmetric key pair (RSA, DSA, or ECDSA)	API call parameters	Status, key pair	RSA private/public key – W DSA private/public key – W ECDSA private/public key – W

[16] DRBG – Deterministic Random Bit Generator
[17] ANSI – American National Standards Institute

Service	Description	Input	Output	CSP and Type of Access
Key Agreement	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).	API call parameter	Status, keys	EC DH Public/Private keys – WRX
Key Wrapping (Key Transport[18])	Perform key wrap with RSA public key, AES key, and Triple-DES Key	API call parameter	Plaintext	RSA Public Key – RX AES Key – RX Triple-DES Key – RX
Signature Generation	Generate a signature for the supplied message using the specified key and algorithm (RSA, DSA, or ECDSA)	API call parameters, key, message	Status, signature	RSA private key – RX DSA private key – RX ECDSA private key – RX
Signature Verification	Verify the signature on the supplied message using the specified key and algorithm (RSA, DSA, or ECDSA)	API call parameters, key, signature, message	Status	RSA public key – RX DSA public key – RX ECDSA public key – RX

2.4.2 User Role

The User role has access to all of the services of the module. The module does not allow multiple concurrent operators in the Approved mode of operation. Per section 6.1 of the NIST FIPS 140-2 Implementation Guidance, the calling application that loaded the module is the only operator. Descriptions of the services available to the User role are provided in Table 5.

Note 1: The following definitions are used for "CSP and Type of Access" column in Table 5:

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

Table 5 Mapping of User Services to Inputs, Outputs, CSPs, and Type of Access

Service	Description	Input	Output	CSP and Type of Access
Run self-test on demand	Performs power-up self-tests	API call parameters	Status	None
Show status[19]	Returns the current mode of the module	None	Status	None

[18] "Key transport" 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 OpenPeak Cryptographic Security Module. By policy, the calling application is required to wrap keys of equal or greater strength.

[19] Running the 'show status' command on the module will, along with its operating status, return the version of the module. The value returned by the module will be version 2.0.2. This version is equivalent to version 1.0 of the OpenPeak Cryptographic Security Module.

Service	Description	Input	Output	CSP and Type of Access
Zeroize keys	Zeroizes and de-allocates memory containing sensitive data	Power cycle; unload module	None	AES key – W AES CMAC Key –W Triple-DES key – W Triple-DES CMAC Key – W HMAC key – W RSA private/public key – W DSA private/public key – W ECDSA private/public key – W EC DH public/private keys – W DRBG Seed – W DRBG Entropy – W DRBG 'C' value – W DRBG 'V' value – W ANSI X9.31 RNG seed – W ANSI X9.31 seed key – W
Generate random number	Returns the specified number of random bits to the calling application	API call parameters	Status, random bits	DRBG Seed – WRX DRBG Entropy – RX DRBG 'C' value – WRX DRBG 'V' value – WRX ANSI X9.31 RNG seed – WRX ANSI X9.31 seed key – RX
Generate message digest	Compute and return a message digest using SHS algorithms	API call parameters, message	Status, hash	None
Generate keyed hash (HMAC)	Compute and return a message authentication code	API call parameters, key, message	Status, hash	HMAC key – RX
Generate Cipher Hash (CMAC)	Compute and return a cipher message authentication code	API call parameters, key, message	Status, hash	AES CMAC Key – RX Triple-DES CMAC Key – RX
Generate symmetric key	Generate and return the specified type of symmetric key (Triple-DES or AES)	API call parameters	Status, key	AES key – W Triple-DES Key – W
Symmetric encryption	Encrypt plaintext using supplied key and algorithm specification (Triple-DES or AES)	API call parameters, key, plaintext	Status, ciphertext	AES key – RX Triple-DES key – RX
Symmetric decryption	Decrypt ciphertext using supplied key and algorithm specification (Triple-DES or AES)	API call parameters, key, ciphertext	Status, plaintext	AES key – RX Triple-DES key – RX
Generate asymmetric key pair	Generate and return the specified type of asymmetric key pair (RSA, DSA, or ECDSA)	API call parameters	Status, key pair	RSA private/public key – W DSA private/public key – W ECDSA private/public key – W

Service	Description	Input	Output	CSP and Type of Access
Key Agreement	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).	API call parameter	Status, key keys	ECDH Public/Private keys – WRX
Key Wrapping (Key Transport[20])	Perform key wrap with RSA public key, AES key, and Triple-DES Key	API call parameter	Plaintext	RSA Public Key – RX AES Key – RX Triple-DES Key – RX
Signature Generation	Generate a signature for the supplied message using the specified key and algorithm (RSA, DSA, or ECDSA)	API call parameters, key, message	Status, signature	RSA private key – RX DSA private key – RX ECDSA private key – RX
Signature Verification	Verify the signature on the supplied message using the specified key and algorithm (RSA, DSA, or ECDSA)	API call parameters, key, signature, message	Status	RSA public key – RX DSA public key – RX ECDSA public key – RX

2.5 Physical Security

The Cryptographic Security Module is a software module, which FIPS defines as a multi-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 Cryptographic Security Module was tested and found compliant with the FIPS 140-2 requirements on the following operating systems and processors:

- Ubuntu 12.04 running an Intel Xeon on a Dell PowerEdge T110
- Ubuntu 12.04 running an Intel Xeon on ESXi 5.1 on a Dell PowerEdge T110
- Ubuntu 12.04 running an AMD Opteron on a SuperMicro AS-1011S-mR2
- Ubuntu 12.04 running an AMD Opteron on ESXi 5.1 on a SuperMicro AS-1011S-mR2
- iOS v5 running on an ARMv7-based Apple A5 processor on an iPad3
- iOS v6 running on an ARMv7s-based Apple A6 processor on an iPhone5
- Android v4.1 running on an ARMv7-based Qualcomm Snapdragon processor on a Samsung Galaxy SIII

All cryptographic keys and CSPs are under the control of the host OS (and hypervisor, where applicable), which protects the keys and CSPs against unauthorized disclosure, modification, and substitution. The

[20] "Key transport" 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 OpenPeak Cryptographic Security Module. By policy, the calling application is required to wrap keys of equal or greater strength.

module only allows access to keys and CSPs through its APIs. The module performs a Software Integrity Test using a FIPS-Approved message authentication code (HMAC SHA-1).

2.7 Cryptographic Key Management

The module implements the FIPS-Approved algorithms listed in Table 6 below.

Table 6 FIPS-Approved Algorithm Implementations

Algorithm	Certificate Number
Symmetric Key	
AES[21] Encryption and Decryption in ECB[22], CBC[23], CTR[24], CFB1[25], CFB8, CFB128, and OFB[26] modes with 128-,192-, and 256-bit key sizes	#2489
AES CCM Encryption and Decryption for 128 / 192 / 256-bit key sizes	#2489
AES GCM Encryption and Decryption for 128 / 192 / 256-bit key sizes	#2489
XTS[27,28,29]-AES Encryption and Decryption for 128 / 256[30]-bit key sizes.	#2489
Triple-DES in ECB, CBC, CTR, CFB1, CFB8, CFB64, and OFB mode for keying option 1 (3 keys)	#1526
Asymmetric Key	
RSA (ANSI X9.31) Key Generation with 2048 / 3072 / 4096-bit modulus; Signature Generation with 2048 / 3072 / 4096-bit modulus; Signature Verification with 1024 /1536 / 2048 / 3072 / 4096-bit modulus	#1283
RSA (PKCS[31] #1 v1.5) Signature Generation with 2048 / 3072 / 4096-bit modulus; Signature Verification with 1024 /1536 / 2048 / 3072 / 4096-bit modulus	
RSA (PSS[32]) Signature Generation with 2048 / 3072 / 4096-bit modulus; Signature Verification with 1024 / 1536 / 2048 / 3072 / 4096-bit modulus	
DSA[33] (FIPS 186-3) Key Generation with 2048 / 3072-bit modulus	#768
DSA Signature Generation and Verification	#768
ECDSA[34] Key Generation with All NIST Defined B, K, and P Curves	#417
ECDSA Signature Generation and Verification	#417

[21] AES – Advance Encryption Service
[22] ECB – Electronic Code Book
[23] CBC – Cipher Block Chaining
[24] CTR – Counter
[25] CFB – Cipher Feedback
[26] OFB – Output Feedback
[27] XTS – XEX-based tweaked-codebook mode with ciphertext stealing
[28] XEX – XOR-Encrypt-XOR
[29] XOR – Exclusive Or
[30] AES-XTS splits the supplied block ciphers in half; therefore users wanting AES 256 and AES 128 encryption will need to choose key sizes of 512 bits and 256 bits respectively.
[31] PKCS – Public Key Cryptography Standard
[32] PSS – Probabilistic Signature Scheme
[33] DSA – Digital Signature Algorithm
[34] ECDSA – Elliptic Curve DSA

Algorithm	Certificate Number
Hashing and Message Authentication	
SHA[35]-1, SHA-224, SHA-256, SHA-384, and SHA-512	#2107
HMAC[36] with SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512	#1531
CMAC Generate and Verify for AES and Triple-DES	#2489 and #1526
Random Number Generator	
ANSI X9.31 Appendix A.2.4 PRNG[37]	#1206
NIST SP800-90A DRBG[38] (Hash-based)	#347
NIST SP800-90A DRBG (HMAC, no reseed)	
NIST SP800-90A DRBG (CTR with AES)	
NIST SP800-90A DRBG (Dual EC, P-256, P-384, P-521)	
Key Agreement Scheme	
EC DH (SP800-56A) using P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, and B-571 curves	#88
Key Transport Scheme	
RSA encrypt/decrypt with 2048/3072/4096-bit modulus	Non-approved, but allowed

***NOTE**: The following algorithms listed in the table above are considered "deprecated" or "legacy-use". For details regarding algorithm deprecation, please refer to NIST Special Publication 800-131A.*

- *ANSI X9.31 random number generation*
- *1024-bit DSA digital signature verification*
- *1024/1536-bit RSA digital signature verification*

Caveat:

- 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)
- RSA (key wrapping; key establishment methodology provides between 112 and 150 bits of encryption strength; non-compliant less than 112 bits of encryption strength)

The module also includes the following non-compliant algorithms:

- 1024/1536-bit RSA key generation
- 1024/1536-bit RSA signature generation
- 1024/8192/16384-bit RSA encrypt/decrypt
- EC DH key agreement using P-192, K-163, and B163 curves

Entropy:

The module generates cryptographic keys whose strengths are modified by available entropy (no assurance of the minimum strength of generated keys). It is up to the calling application to provide sufficient entropy sources to be registered as callback functions for random number generation. For the ANSI X9.31 PRNG mechanism, the callback function used must provide 128 bits of entropy. For the SP 800-90A DRBGs, the callback functions must provide the minimum amount of entropy specified in SP 800-90A Table 2 (for the

[35] SHA – Secure Hash Algorithm
[36] HMAC – Keyed-Hash Message Authentication Code
[37] PRNG – Pseudo Random Number Generator
[38] DRBG – Deterministic Random Bit Generator

Hash_DRBG and HMAC_DRBG), Table 3 (for the CTR_DRBG) and Table 4 (for the Dual_EC_DRBG). Those functions must return an error if the minimum entropy strength cannot be met.

.

The module supports the critical security parameters (CSPs) listed below in Table 7.

Table 7 List of Cryptographic Keys, Cryptographic Key Components, and CSPs

CSP	CSP Type	Generation / Input	Output	Storage	Zeroization	Use
AES key	AES128, 192, 256 bit key Only AES XTS_128 (256 bits), and XTS_256 (512 bits) keys[40] are supported.	Internally generated via Approved SP 800-90A DRBG or ANSI X9.31 PRNG; or Input via API call parameter	Output in plaintext via hardware appliance or mobile device INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Encryption, decryption
AES GCM[41] key	AES GCM 128, 192, 256 bit key	Internally generated via Approved SP 800-90A DRBG or ANSI X9.31 PRNG; or Input via API call parameter	Output in plaintext via hardware appliance or mobile device INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Encryption, decryption
AES GCM initialization vector	128 bit value	Input via API	Never	Plaintext in volatile memory	Unload module; API call; Remove Power	Initialization vector for AES GCM
AES CMAC key	AES CMAC 128, 192, 256 bit key	Internally generated via Approved SP 800-90A DRBG or ANSI X9.31 PRNG; or Input via API call parameter	Output in plaintext via hardware appliance or mobile device INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Signature generation and verification

[40] XTS-AES Key – The Crypto Officer and User roles must ensure that the length of the data unit for any instance of an implementation of XTS-AES shall not exceed 2^{20} blocks.
[41] AES GCM IV – 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.

.

CSP	CSP Type	Generation / Input	Output	Storage	Zeroization	Use
Triple-DES key	Triple-DES 192 bit key (keying option 1)	Internally generated via Approved SP 800-90A DRBG or ANSI X9.31 PRNG; or Input via API call parameter	Output in plaintext via hardware appliance or mobile device INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Encryption, decryption
Triple-DES CMAC key	Triple-DES CMAC 192 bit key (keying option 1)	Internally generated via Approved SP 800-90A DRBG or ANSI X9.31 PRNG; or Input via API call parameter	Output in plaintext via hardware appliance or mobile device INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Signature generation and verification
HMAC key	HMAC 160, 224, 256, 384, or 512 – bit key	Internally generated via Approved SP 800-90A DRBG or ANSI X9.31 PRNG; or Input via API call parameter	Output in plaintext via hardware appliance or mobile device INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Message Authentication with SHS
RSA private key	RSA 2048, 3072, or 4096 bit key	Internally generated via Approved SP 800-90A DRBG or ANSI X9.31 PRNG; or Input via API call parameter	Output in plaintext via hardware appliance or mobile device INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Signature generation, decapsulation

.

CSP	CSP Type	Generation / Input	Output	Storage	Zeroization	Use
RSA public key	RSA 1024, 1536, 2048, 3072, 4096, 8192, or 16384 bit key	Internally generated via Approved SP 800-90A DRBG or ANSI X9.31 PRNG; or Input via API call parameter	Output in plaintext via hardware appliance or mobile device INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Signature verification, encapsulation
DSA private key	DSA 224 or 256 bit key	Internally generated via Approved SP 800-90A DRBG or ANSI X9.31 PRNG; or Input via API call parameter	Output in plaintext via hardware appliance or mobile device INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Signature generation
DSA public key	DSA 1024, 2048, or 3072 bit key	Internally generated via Approved SP 800-90A DRBG or ANSI X9.31 PRNG; or Input via API call parameter	Output in plaintext via hardware appliance or mobile device INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Signature verification
EC DSA private key	EC DSA 224 or 256 bit key	Internally generated via Approved SP 800-90A DRBG or ANSI X9.31 PRNG; or Input via API call parameter	Output in plaintext via hardware appliance or mobile device INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Signature generation

.

CSP	CSP Type	Generation / Input	Output	Storage	Zeroization	Use
EC DSA public key	EC DSA 1024, 2048, or 3072 bit key	Internally generated via Approved SP 800-90A DRBG or ANSI X9.31 PRNG; or Input via API call parameter	Output in plaintext via hardware appliance or mobile device INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Signature verification
EC DH private key	EC DH private key agreement key	Internally generated via Approved SP 800-90A DRBG or ANSI X9.31 PRNG; or Input via API call parameter	Output in plaintext via hardware appliance or mobile device INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Used by host application
EC DH public key	EC DH public key agreement key	Internally generated via Approved SP 800-90A DRBG or ANSI X9.31 PRNG; or Input via API call parameter	Output in plaintext via hardware appliance or mobile device INT Path	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Used by host application
DRBG Seed	Random data – 440 or 880 bits	Generated internally using nonce along with DRBG entropy input.	Never	Keys are not persistently stored by the module	Unload module; API call; Remove Power	Seeding material for SP 800-90A DRBGs
DRBG Entropy	256 bit value	Externally Generated[42]; Input via API	Never	Plaintext in volatile memory	Unload module; API call; Remove Power	Entropy material for SP 800-90A DRBGs

[42] The module employs a non-deterministic random number generator specific to the host operating system, which is outside of the logical cryptographic boundary. Please see the Vendor Evidence document, VE07.13.01, for more information on the entropy sources of the module.

.

CSP	CSP Type	Generation / Input	Output	Storage	Zeroization	Use
DRBG 'C' Value	Internal state value	Internally Generated	Never	Plaintext in volatile memory	Unload module; API call; Remove Power	Used for Hash_DRBG
DRBG 'V' Value	Internal state value	Internally Generated	Never	Plaintext in volatile memory	Unload module; API call; Remove Power	Used for Hash_DRBG, HMAC_DRBG, and CTR_DRBG
DRBG 'Key' Value	Internal state value	Internally Generated	Never	Plaintext in volatile memory	Unload module; API call; Remove Power	Used for HMAC_DRBG and CTR_DRBG
ANSI X9.31 Appendix A.2.4 PRNG seed	128-bit random value	Internally generated	Never exits the module	Plaintext in volatile memory	Rebooting the modules	Seeding the FIPS-Approved ANSI X9.31 PRNG
ANSI X9.31 Appendix A.2.4 PRNG key	AES 128-bit key	Internally generated	Never exits the module	Plaintext in volatile memory	Rebooting the modules	Seeding the FIPS-Approved ANSI X9.31 PRNG

2.8 EMC/EMI

The Cryptographic Security Module is a software module. Therefore, the only electromagnetic interference produced is that of the host platform on which the module resides and executes. FIPS 140-2 requires that the host systems on which FIPS 140-2 testing is performed meet the Federal Communications Commission (FCC) EMI and EMC requirements for business use as defined in Subpart B, Class A of FCC 47 Code of Federal Regulations Part 15. However, because all of the tested platforms are Class B devices and meet the requirements for home use, the EMI and EMC requirements are met. Additionally, all systems sold in the United States must meet these applicable FCC requirements.

2.9 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, expected error status, and error resolution.

2.9.1 Power-Up Self-Tests

The power-up self-tests are invoked by the FIPS_mode_set(), which invokes the fips_mode_module_set() function. If the power-up self-tests execute successfully, this function will return an integer value of "1" to the calling application to indicate success; this function will return an integer value of"0" to the calling application to indicate failure.

A failure encountered during any of the following power-up self-tests will cause the module to enter the Critical Error state, and an internal flag is set to preventing any invocation of the module's cryptographic algorithms. The module's running process is aborted and the Crypto Officer must unload the module from memory and reinitialize the module or the Crypto Officer must power down or restart the host system or host mobile device to clear the error state.

The Cryptographic Security Module performs the following self-tests at power-up:

- Software integrity check (HMAC-SHA-1)
- Known Answer Tests (KATs)
    - AES-ECB KAT
    - AES-CCM KAT
    - AES-GCM KAT
    - XTS-AES KAT
    - AES CMAC KAT
    - Triple DES KAT
    - Triple-DES CMAC KAT
    - HMAC KAT with SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512
    - RSA Digital Signature Generation KAT (2048 bit, SHA-256, PKCS#1)
    - RSA Digital Signature Verification KAT (2048 bit, SHA-256, PKCS#1)
    - DSA PCT[43]
    - ECDSA PCT
    - ANSI X9.31 Appendix A.2.4 PRNG KAT
    - NIST SP800-90A DRBG KATs (all modes)
        - CTR_DRBG: AES, 256 bit with and without derivation function
        - HASH_DRBG: SHA256
        - HMAC_DRBG: SHA256
        - Dual_EC_DRBG: P-256 and SHA256)
    - EC DH KAT

[43] PCT – Pairwise Consistency Test

2.9.2 Conditional Self-Tests

Conditional self-tests are performed by the module whenever a new random number is generated or when a new RSA, DSA. or ECDSA key pair is generated. If any of the conditional self-tests fail, the module will log the error and halt all processing and set the internal failure flag, ensuring that data output from the module is inhibited. The CO must restart the host system or unload and reload the calling application to clear a continuous RNG test failure.

The Cryptographic Security Module performs the following conditional self-tests:

- Continuous RNG Test for the SP800-90A DRBG
- Continuous RNG Test for the ANSI X9.31 Appendix A.2.4 PRNG
- RSA pairwise consistency test for key pair generation
- DSA pairwise consistency test for key pair generation
- ECDSA pairwise consistency test for key pair generation

2.9.3 Critical Functions Self-Tests

The Cryptographic Security Module implements the four SP 800-90A DRBGs as its random number generators. Each of the DRBGs employ four critical functions which must also be tested on a regular basis to ensure the security of the SP 800-90A DRBG. If any of the critical function self-tests fail, the module will log the error and halt all system processing and set the internal failure flag, ensuring that data output from the module is inhibited. In order to resolve a cryptographic self-test error, the CO shall reset the host system of the module or unload and reload the calling application that invoked the module. Therefore, the following critical function tests are also implemented by the cryptographic module:

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

2.10 Mitigation of Other Attacks

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

3 Secure Operation

The Cryptographic Security Module meets Level 1 requirements for FIPS 140-2. The sections below describe how to place and keep the module in FIPS-Approved mode of operation.

3.1 Secure Management

The Cryptographic Security Module is distributed only as part of OpenPeak's ADAM software applications (SANCTION, SECTOR, and OPENSHOP) and is not distributed as a separate binary. The Crypto Officer is responsible for the installation of SECTOR on host systems running Android and iOS, and is responsible for the installation of OPENSHOP and SANCTION on a host systems running Ubuntu. Successfully completing installation of the ADAM software components will complete the initial set up for the Cryptographic Security Module.

3.1.1 Initialization

The module supports only an Approved mode of operation. The module requires an initialization sequence (see IG 9.5): the calling application invokes the FIPS_mode_set(), which invokes the fips_mode_module_set() function, which returns a "1" for success and "0" for failure. If the fips_mode_set() function call fails, then all cryptographic services will fail indefinitely. The calling application can call the fips_module_mode() function to determine if the module has been successfully configured for the Approved mode.

When the SANCTION, SECTOR, and OPENSHOP components are started, the module runs its power-up self-tests which includes software integrity test that checks the integrity of the module by using an HMAC SHA-1 digest. If the integrity check succeeds, then the module performs KATs on all the required algorithms. When the module passes all of the power-up self-tests, the module is in its FIPS-Approved mode of operation. If any self-test fails, the module enters a critical error state, ceasing all cryptographic functionality, and throws an exception to the calling application. The module must be reinstalled to leave the critical error state.

The ADAM components of SANCTION and OPENSHOP allow the Crypto Officer to perform power-up self-tests on demand by power-cycling the host system. Crypto Officers may invoke the power-up self-tests on demand on the ADAM component SECTOR by power-cycling the mobile device.

3.1.2 Management

Since the Crypto Officer cannot directly interact with the module, no specific management activities are required to ensure that the module runs securely; the module only executes in a FIPS-Approved mode of operation. If any irregular activity is noticed or the module is consistently reporting errors, then OpenPeak, Inc. Customer Support should be contacted.

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

3.1.3 Zeroization

The module does not persistently store any keys or CSPs. All ephemeral keys used by the module are zeroized upon reboot, or session termination.

.

3.2 User Guidance

Only the module's cryptographic functionalities are available to the User. Users are responsible to use only the services that are listed in Table 5 above. Although the User does not have any ability to modify the configuration of the module, they should report to the Crypto Officer if any irregular activity is noticed.

4 Acronyms

Table 8 provides definitions for the acronyms used in this document.

Table 8 Acronyms

Acronym	Definition
ADAM	Advanced Device and Application Management
AES	Advanced Encryption Service
ANSI	American National Standards Institute
API	Application Programming Interface
CBC	Cipher-Block Chaining
CDH	Cofactor Diffie-Hellman
CFB	Cipher Feedback
CMVP	Cryptographic Module Validation Program
CPU	Central Processing Unit
CSEC	Communications Security Establishment Canada
CSP	Critical Security Parameter
CTR	Counter
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Algorithm
DVD	Digital Video Disk
EC	Elliptic Curve
ECB	Electronic Codebook
ECDSA	Elliptic Curve DSA
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FCC	Federal Communications Commission
FIPS	Federal Information Processing Standard
GCM	Galois/Counter Mode
GUI	Graphical User Interface
HMAC	Keyed-Hash Message Authentication Code
iOS	iPhone Operating System
KAT	Known Answer Test
LCD	Liquid Crystal Display
MDM	Mobile Device Management

.

Acronym	Definition
NIST	National Institute of Standards and Technology
NVLAP	National Voluntary Laboratory Accreditation Program
PCT	Pairwise Consistency Test
PKCS	Public Key Cryptography Standard
PRNG	Pseudo Random Number Generator
PSS	Probabilistic Signature Scheme
RAM	Random Access Memory
RNG	Random Number Generator
ROM	Read Only Memory
RSA	Rivest Shamir and Adleman
SHA	Secure Hash Algorithm
TCP	Transmission Control Protocol
USB	Universal Serial Bus

Prepared by:
Corsec Security, Inc.

13135 Lee Jackson Memorial Highway, Suite 220
Fairfax, VA 22033
United States of America

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