Diversinet Corp.

Diversinet Java Crypto Module for Mobile

Software Version: 2.0

FIPS 140-2 Non-Proprietary Security Policy

FIPS Security Level: 1
Document Version: 1.0

Prepared for:

Diversinet Corp.
2235 Sheppard Avenue East, Suite 1700
Toronto, Ontario M2J5B5
Canada

Phone: +1 (416) 756-2324
Email: sales@diversinet.com
http://www.diversinet.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

Table of Figures

List of Tables

1 Introduction

1.1 Purpose

This is a non-proprietary Cryptographic Module Security Policy for the Diversinet Java Crypto Module for Mobile from Diversinet Corp. This Security Policy describes how the Diversinet Java Crypto Module for Mobile 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 Diversinet Java Crypto Module for Mobile is referred to in this document as the Diversinet Java Crypto Module for Mobile, the 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 Diversinet website (www.diversinet.com) contains information on the full line of products from Diversinet.
- 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 Diversinet. With the exception of this Non-Proprietary Security Policy, the FIPS 140-2 Submission Package is proprietary to Diversinet and is releasable only under appropriate non-disclosure agreements. For access to these documents, please contact Diversinet.

2 Diversinet Java Crypto Module for Mobile

2.1 Overview

The Diversinet MobiSecure® Platform enables enterprises to rapidly deploy HIPAA[1]-compliant mobile health applications to anyone, anywhere, on any mobile device.

MobiSecure® Platform offers convenient and secure management of critical data and personal information, from a mobile phone, using SMS[2], or a tablet or PC[3], directly over the Internet and wireless networks, utilizing SOAP[4]. Furthermore, it provides organizations the necessary tools to deploy customized mobile applications that meet their business and brand needs. MobiSecure® Platform uses two-factor authentication technology for user and device authentication and state-of-the-art encryption techniques for data protection. Its uniqueness and strengths lies in the ability for organizations to securely publish content to a wide range of different mobile phones. In addition, it enables the customization of both the application configuration as well as the user interface over-the-air without the need of re-deployment. Figure 1, below, illustrates the MobiSecure® Platform.

The security of MobiSecure® Platform is formed around in-depth security design principles that provide controls at multiple levels of data storage, access and transfer. Although Diversinet's focus and expertise is in designing and implementing high security standards at the software architecture, application and cryptographic levels, Diversinet works with its customers to define the overall security strategy, which includes operations security; communication and network security; access controls; business continuity and disaster recovery; and compliance with privacy regulations.

[1] HIPAA – Health Insurance Portability and Accountability Act
[2] SMS – Short Message Service
[3] PC – Personal Computer
[4] SOAP – Simple Object Access Protocol

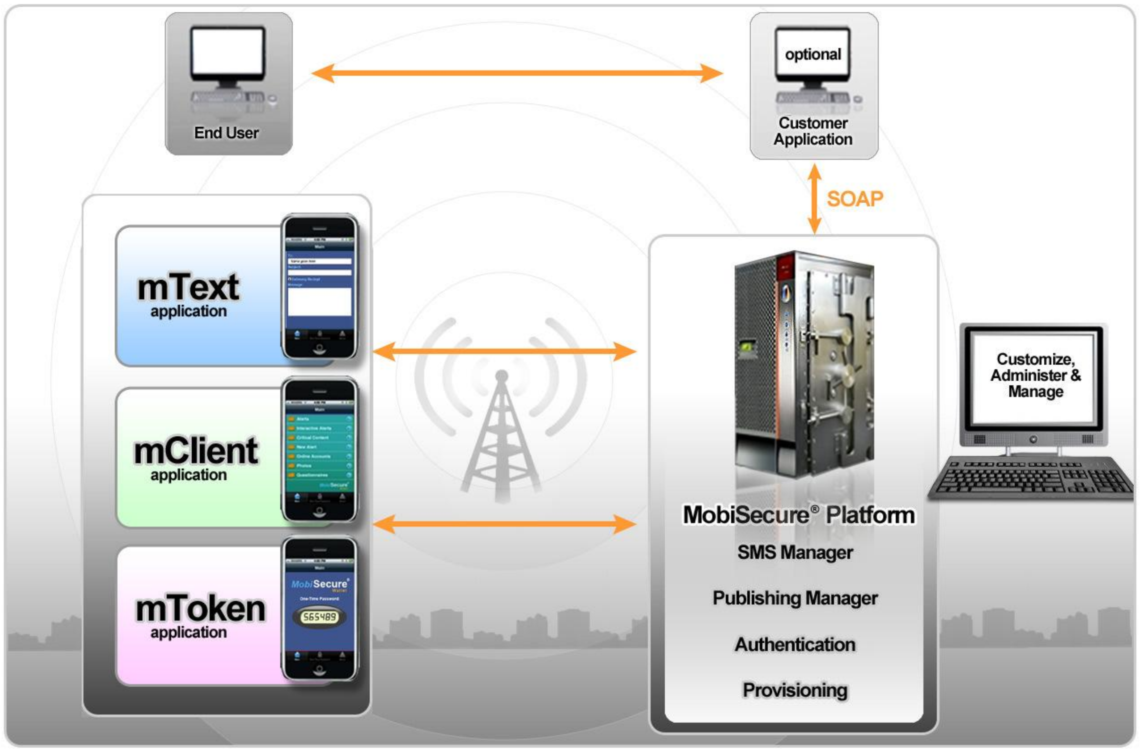

Figure 1 – Diversinet MobiSecure® Platform

The Diversinet Java Crypto Module for Mobile is the FIPS-validated module that provides cryptographic functionality for the MobiSecure® Platform. The Diversinet Java Crypto Module for Mobile is validated at the following FIPS 140-2 Section levels:

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
14	Cryptographic Module Security Policy	1

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

2.2 Module Specification

The Diversinet Java Crypto Module for Mobile is a software module with a multi-chip standalone embodiment. The overall security level of the module is level 1. The following sections will define the physical and logical boundary of the module.

The cryptographic module is a single APK[6] binary, named DiversinetCryptoService.apk, that provides cryptographic and secure communication services for other applications developed by Diversinet. In this document, those applications will be referred to as the calling application. The module is used as a JCE[7] and JCA[8] by the calling application to provide encryption, decryption, hash generation and verification, certificate generation and verification, random bit generation, and message authentication functions.

2.2.1 Physical Cryptographic Boundary

As a software cryptographic module, there are no physical security mechanisms implemented; the module must rely on the physical characteristics of the host platform. The physical cryptographic boundary of the Diversinet Java Crypto Module for Mobile is defined by the hard enclosure around the host platform on which it executes. Figure 2 below shows the physical block diagram of the module which runs on a Qualcomm QSD8250 Snapdragon chipset with a 1 GHz[9] Scorpion processor, running Android OS[10] v2.2.

[6] APK – Android Application Package
[7] JCE – Java Cryptography Extension
[8] JCA – Java Cryptography Architecture
[9] GHz – Gigahertz
[10] OS – Operating System

.

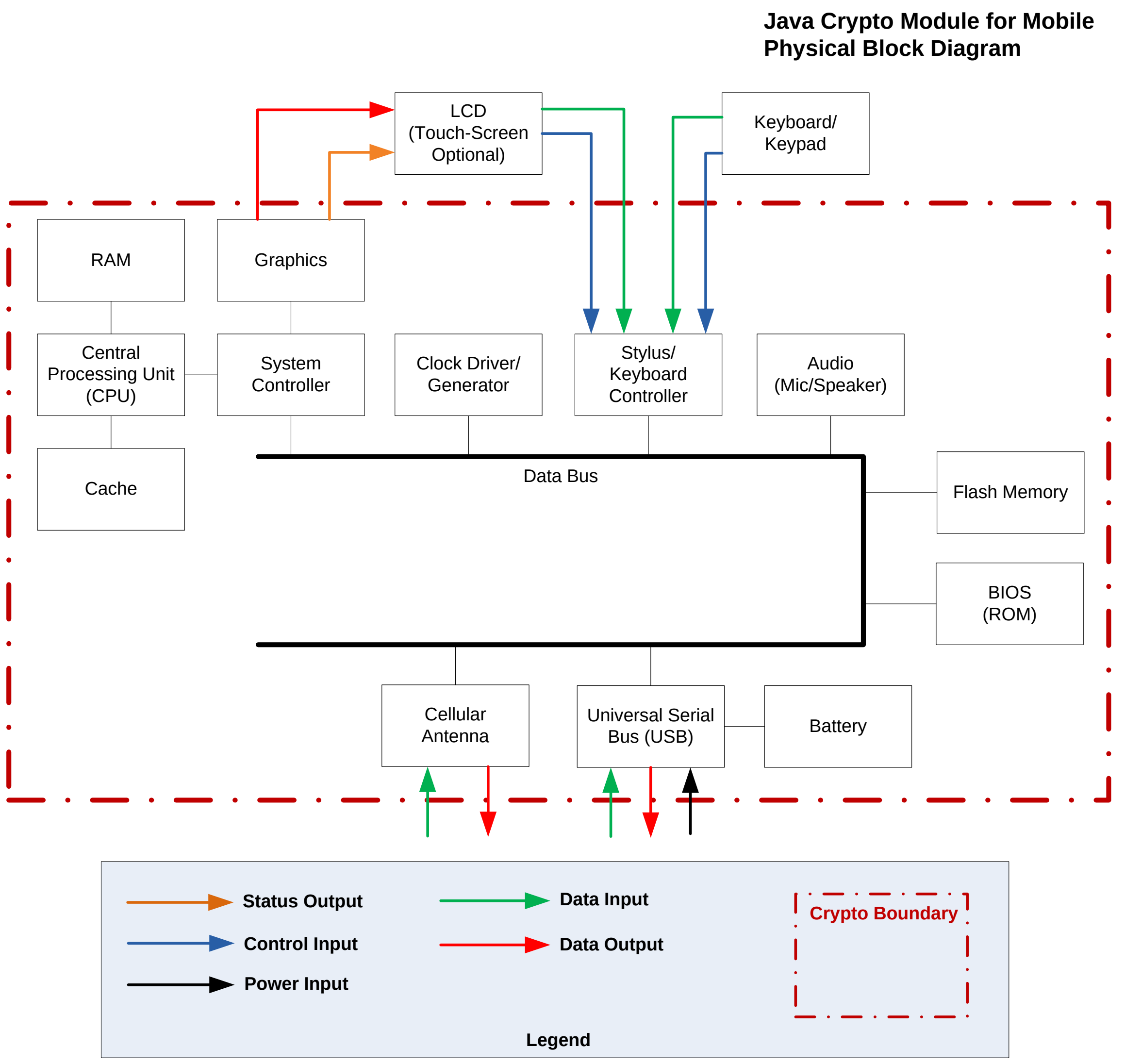

Figure 2 – GPC Server Block Diagram

2.2.2 Logical Cryptographic Boundary

Figure 3, below, shows a logical block diagram of the module. The module's logical cryptographic boundary (also illustrated in Figure 3) encompasses all functionality provided by the module as described in this document. The module takes the form of a single shared library that can be called by calling applications, executing in the Dalvik virtual machine, to provide cryptographic services.

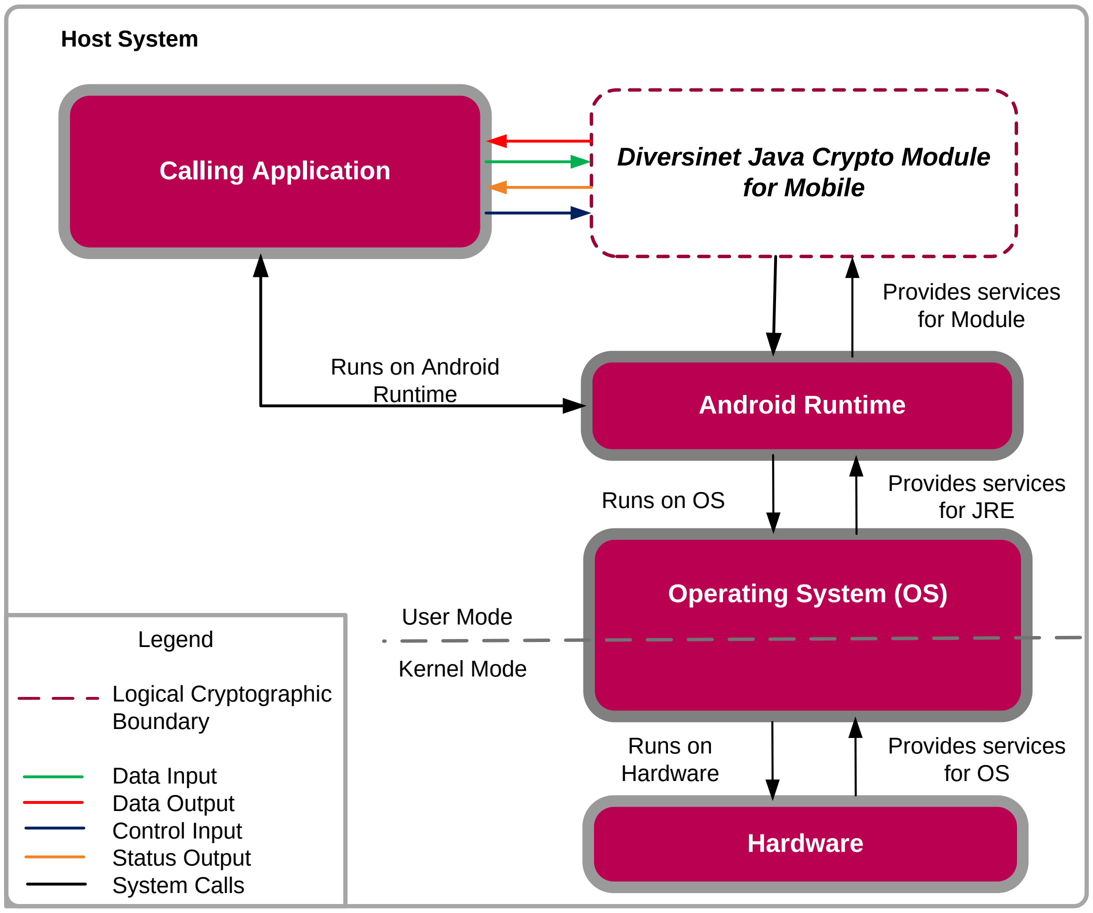

Figure 3 – Diversinet Java Crypto Module for Mobile Logical Block Diagram

2.3 Module Interfaces

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

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

As a software module, the module has no physical characteristics. Thus, the module's manual controls, physical indicators, and physical and electrical characteristics are those of the host platform.

All of these physical interfaces of the host platform are separated into logical interfaces defined by FIPS 140-2, as described in the following table:

.

Table 2 – FIPS 140-2 Logical Interface Mappings

FIPS 140-2 Interface	Physical Interface	Module Interface (API)
Data Input	USB[11] port, Cellular Antenna	Method calls that accept, as their arguments, data to be used or processed by the module
Data Output	USB port, Cellular Antenna	Arguments for a method that specify where the result of the method is stored
Control Input	USB port, LCD[12] screen, Keyboard/Keypad	Method calls utilized to initiate the module and the method calls used to control the operation of the module.
Status Output	USB port, LCD screen	Return status for method calls
Power Input	USB port	Not Applicable

2.4 Roles and Services

There are two roles in the module (as required by FIPS 140-2) that operators may assume: a Crypto Officer role and a User role.

2.4.1 Crypto Officer Role

The Crypto Officer role has the ability to initialize the module, determine module status, and zeroize all module keys and CSPs[13]. Descriptions of the services available to the Crypto Officer role are provided in Table 3, below. Please note that the keys and CSPs listed in the table indicate the type of access required using the following notation:

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

Table 3 – Crypto Officer Services

Service	Description	CSP and Type of Access
OnCreate()	Loads module and calls power-up self-tests	N/A
getStatus()	Returns the current mode of the module	N/A

[11] USB – Universal Serial Bus
[12] LCD – Liquid-Crystal Display
[13] CSP – Critical Security Parameter

.

Service	Description	CSP and Type of Access
Zeroize keys	Format flash memory of host mobile device and reset to factory settings	AES[14] key – W Triple DES[15] key – W HMAC[16] key – W RSA[17] private key – W RSA public key – W

Key zeroization by the Crypto-Officer will take the form of a full format of the host mobile device's flash memory and an operation to restore it to factory settings. This command is not a part of the module's API. The mobile device's OS provides this service as part of its API.

2.4.2 User Role

The User role has the ability to generate all CSPs supported by the module and perform encryption, decryption, and verification operations with the generated CSPs. Descriptions of the services available to the User role are provided in Table 4, below.

Table 4 – User Services

Service	Description	CSP and Type of Access
sha1()	Compute and return a message digest using the SHA[18]-1 algorithm	None
sha256()	Compute and return a message digest using the SHA-256 algorithm	None
sha512()	Compute and return a message digest using the SHA-512 algorithm	None
selfTests()	Performs power-up self-tests	Integrity check HMAC key – RX AES key – X Triple DES key – X RSA private key – X RSA public key – X HMAC keys – X DRBG[19] seed/key – X
hmac()	Compute and return a message authentication code using HMAC SHA-1, HMAC SHA-256, or HMAC SHA-512	HMAC key – RX
encrypt()	Encrypt plaintext using supplied key and algorithm specification (Triple DES or AES)	AES key – RX Triple DES key – RX

[14] AES – Advanced Encryption Standard
[15] DES – Data Encryption Standard
[16] HMAC – Hashed (Keyed-) Message Authentication Code
[17] RSA – Rivest, Shamir, Adleman
[18] SHA – Secure Hash Algorithm
[19] DRBG – Deterministic Random Bit Generator

.

Service	Description	CSP and Type of Access
decrypt()	Decrypt ciphertext using supplied key and algorithm specification (Triple DES or AES)	AES key – RX Triple DES key – RX
generateNextBytes()	Generate random bits of the requested length	DRBG seed/key – X
rsaSign()	Generate a signature for the supplied message using the specified key and algorithm	RSA private key – RX
rsaVerify()	Verify the signature on the supplied message using the specified key and algorithm	RSA public key – RX
rsaEncrypt()	Encrypt a block of data using the specified key	RSA private key – RX
rsaDecrypt()	Decrypt a block of data using the specified key	RSA public key – RX

2.4.3 Authentication

The module does not support any authentication mechanism. Operators of the module implicitly assume a role based on the service of the module being invoked. Since all services offered by the module can only be used by either the Crypto Officer or the User, the roles are mutually exclusive. Thus, when the operator invokes a Crypto Officer role service, he implicitly assumes the Crypto Officer role. When the operator invokes a User role service, he implicitly assumes the User role.

2.5 Physical Security

The Diversinet Java Crypto Module for Mobile 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 Diversinet Java Crypto Module for Mobile was tested and found compliant with the FIPS 140-2 requirements on the following operating system:

- Qualcomm QSD8250 Snapdragon chipset with a 1 GHz Scorpion processor running Android OS v2.2.

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

The Diversinet Java Crypto Module for Mobile was tested on Android v2.2, using an Qualcomm Snapdragon processor. Diversinet further affirms that the module will operate on any processor supporting Android v1.6 (or later) in accordance with FIPS 140-2 Implementation Guidance G.5.

As per FIPS 140-2 IG G.5 implementation guidance, Diversinet affirms that for, Blackberry OS v4.2 and higher, for the cryptographic implementation, Diversinet uses the same cryptographic source code that was used for Android v2.2 operating systems.

As per FIPS 140-2 IG G.5 implementation guidance, Diversinet affirms that, for the Java ME MIDP environment, for the cryptographic implementation, Diversinet uses the same cryptographic source code that was used for Android v2.2 operating systems.

2.7 Cryptographic Key Management

The module implements the FIPS-Approved algorithms listed in Table 5 below.

Table 5 – FIPS-Approved Algorithm Implementations

Algorithm	Certificate Number
Symmetric Key	
AES: ECB[20], CBC[21], and CTR[22] mode for 128, 192, and 256 bit key sizes	#1966
Triple DES: CBC mode for 1 and 2 keying option	#1277
Asymmetric Key	
RSA PSS[23] signature generation/verification: 1024- , 2048-bit (w/ SHA-1, SHA-256, SHA-512)	#1018
Secure Hashing Standard (SHS)	
SHA-1, SHA-256, SHA-512	#1724
Message Authentication	
HMAC SHA-1, HMAC SHA-256, HMAC SHA-512	#1186
Random Number Generator	
NIST SP800-90 DRBG (Counter-based)	#176

NOTE: The following security functions have been deemed "deprecated" or "restricted" by NIST. Please refer to NIST Special Publication 800-131A for further details.

- The use of two-key Triple DES for encryption is **restricted** after December 31, 2010.
- The use of key lengths providing 80 bits of security strength for digital signature generation is **deprecated** after December 31, 2010.

Additionally, the module utilizes the following non-FIPS-approved algorithm implementations:

  -
- RSA encrypt/decrypt
  - RSA encrypt/decrypt functionality is not FIPS-Approved, but is Allowed in FIPS mode

[20] ECB – Electronic Codebook
[21] CBC – Cipher-Block Chaining
[22] CTR – Counter
[23] PSS – Probabilistic Signature Scheme

.

The module supports the critical security parameters (CSPs) listed below in Table 6.

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

CSP/Key	CSP/Key Type	Generation / Input	Output	Storage	Zeroization	Use
HMAC Integrity key	HMAC key	Never	Never output from the module	In module binary	N/A	Software integrity check
AES key	AES 128-, 192-, 256-bit key	Input electronically in plaintext	Never output from the module	Plaintext in volatile memory	N/A	Encryption, decryption
Triple DES key	Triple DES 168-bit key	Input electronically in plaintext	Never output from the module	Plaintext in volatile memory	N/A	Encryption, decryption
HMAC key	HMAC key	Input electronically in plaintext	Never output from the module	Plaintext in volatile memory	N/A	Message Authentication with SHS
RSA private key	RSA 1024-, 2048-bit key	Input electronically in plaintext	Never output from the module	Plaintext in volatile memory	N/A	Signature generation, decryption
RSA public key	RSA 1024-, 2048-bit key	Input electronically in plaintext	Never output from the module	Plaintext in volatile memory	N/A	Signature verification, encryption
DRBG seed	1000-bit entropy value	Internally generated from system entropy and SHA-256 hash	Never output from the module	Plaintext in volatile memory	N/A	Used to seed the DRBG
DRBG nonce	64-bit counter value	Internally generated from system entropy	Never output from the module	Plaintext in volatile memory	N/A	Used to seed the DRBG

2.8 Self-Tests

2.8.1 Power-Up Self-Tests

The Diversinet Java Crypto Module for Mobile performs the following self-tests at power-up:

- Software integrity check (HMAC SHA-1)
- Known Answer Tests (KATs)
  - AES KAT
  - Triple-DES KAT
  - RSA KAT for sign/verify
  - SHA-1 HMAC KAT
  - SHA-2 HMAC KAT
  - NIST SP800-90 DRBG KAT

Note that the HMAC KATs with SHA-1, SHA-256, and SHA-512 utilize (and thus test) the full functionality of the SHA-1 and SHA-2s; thus, no independent KATs for SHA-1 and SHA-2s implementations are required. Likewise, since the KAT for HMAC SHA-2 will test HMAC SHA-256 and HMAC SHA-512, no separate test is required.

2.8.2 Conditional Self-Tests

The Diversinet Java Crypto Module for Mobile performs the following conditional self-tests:

- NIST SP800-90 DRBG Continuous test

2.8.3 Critical Function Tests

The Diversinet Java Crypto Module for Mobile performs the following critical function tests:

- NIST SP800-90 DRBG Health test

2.9 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 Diversinet Java Crypto Module for Mobile 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 Diversinet Java Crypto Module for Mobile is distributed as part of Diversinet's system applications. The module is installed when a calling application checks for it. If the module is not present, then the calling application will install it.

3.1.1 Initialization

The module is initialized when the calling application tries to connect to the module. The Android Runtime calls a method, onCreate(), after it loads the module into memory. This is a method implemented by the Android Runtime to bind the module. Upon initialization of the module, the module runs its power-up self-tests which includes software integrity test that checks the integrity of the module, and known-answer tests for each cryptographic algorithm. To perform the integrity test, the method extracts several files, AndroidManifest.xml, resources.arsc, icon.png, and classes.dex, from the module The .dex file contains the bytecode for the Dalvik virtual machine to execute the code. The method runs an HMAC SHA-1 hash on these files, and compares it to a pre-calculated, stored value. If both HMAC SHA-1 hashes match, then the module passes the integrity test. Conversely, if both values do not match, then the self-test fails. Upon failure of the integrity test, the module enters the critical error state to indicate the failure of the integrity test. When the module passes all of the self-tests, the module is in its FIPS-Approved mode of operation. If any algorithm self-test fails, the module enters the critical error state. In order to recover, the module must be reinstalled and reloaded. Power-up self-tests can also be performed on demand by invoking the selfTests() method or by cycling the power on the host platform.

The module is loaded by applications which make calls to the module as a stand-alone cryptographic library.

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 Diversinet Corp. customer Support should be contacted.

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 4 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

This section describes the acronyms.

Table 7 – Acronyms

Acronym	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
APK	Android Application Package
CBC	Cipher-Block Chaining
CMVP	Cryptographic Module Validation Program
CSEC	Communications Security Establishment Canada
CSP	Critical Security Parameter
CTR	Counter
DES	Data Encryption Standard
DVD	Digital Versatile Disc
DRBG	Deterministic Random Bit Generator
ECB	Electronic Codebook
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FIPS	Federal Information Processing Standard
GHz	Gigahertz
HIPAA	Health Insurance Portability and Accountability Act
HMAC	(Keyed-)Hash Message Authentication Code
GPC	General-Purpose Computer
JAR	Java Archive
JCA	Java Cryptography Architecture
JCE	Java Cryptography Extension
JDK	Java Development Kit
KAT	Known Answer Test
NIST	National Institute of Standards and Technology
OS	Operating System
PC	Personal Computer
PKCS	Public-Key Cryptography Standard
PSS	Probabilistic Signature Scheme
RSA	Rivest, Shamir, and Adleman

.

Acronym	Definition
SATA	Serial Advanced Technology Attachment
SCSI	Small Computer System Interface
SHA	Secure Hash Algorithm
SHS	Secure Hashing Standard
SMS	Short Message Service
SOAP	Simple Object Access Protocol
Triple DES	Triple Data Encryption Standard
USB	Universal Serial Bus

Prepared by:
Corsec Security, Inc.

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

Phone: (703) 267-6050
Email: info@corsec.com
http://www.corsec.com