Federal Information Processing Standard (FIPS) 140-2
Design Assurance Level-3

Good Technology, Inc.
FIPS Crypto on Windows Mobile
Version 4.7.0.50906

FIPS 140-2 Non-Proprietary Security Policy

OCTOBER 2, 2012

DOCUMENT VERSION CONTROL

VERSION	DATE	AUTHOR (S)	DESCRIPTION	REASON FOR CHANGE
1.0	11 Jan 2004	Prathaban Selvaraj Good Technology	Initial Version	
1.1	06 Aug 2005	Richard Levenberg Good Technology	Modifications for Windows CE 4.2	
1.2	20 Sept 2005	Richard Levenberg Good Technology	Key Zeroization update	Update for Key Zeroization
1.3	13 Oct 2005	Richard Levenberg Good Technology	Minor edits	
1.4	25 Oct 2005	Rory Saunders CEAL, CygnaCom Solutions, Inc.	Minor edits	Incorporated previous NIST/CSE comments
1.5	08 Feb 2006	Rory Saunders CEAL, CygnaCom Solutions, Inc.	Minor edits	Incorporated previous NIST/CSE comments
1.6	22 Apr 2009	Vasanthan Gunaratnam Good Technology	Minor edits	Updates for Win CE 5.2 in accordance with IG G.5
1.7	24 June 2009	Krishna Puttagunta Good Technology	Minor Edits.	Changes required for Design Assurance 3 Certification
1.8	20 Jan 2010	Sriram Krishnan Good Technology	Minor Edits.	Updates for iPhone 3.0 & 3.1 in accordance with IG G.5

1.9	05 May 2010	Sriram Krishnan Good Technology	Minor Edits.	Updates for iPhone 3.2 and Android 1.6, 2.0 & 2.1 in accordance with IG G.5
2.0	11 Oct 2010	Sriram Krishnan Good Technology	Minor Edits.	Updates for iPhone 4.0 & 4.1 and Android 2.2 in accordance with IG G.5
2.1	18 Jan 2011	Sriram Krishnan Good Technology	Minor Edits.	Updates for iPhone 4.2 in accordance with IG G.5
2.2	18 Jan 2011	Sriram Krishnan Good Technology	Minor Edits.	Updates for iOS 4.3 in accordance with IG G.5
2.3	17 Oct 2011	Rick Pitz Good Technology	Minor Edits.	Updates for iOS 5 in accordance with IG G.5
2.4	13 March 2012	Rick Pitz Good Technology	Minor Edits.	Updates for iOS 5.1 and Windows Phone 7.5 in accordance with IG G.5
2.5	19 Sep 2012	Rick Pitz Good Technology	Minor Edits	Updates for iOS 6.0 in accordance with IG G.5
2.6	2 Oct 2012	Rick Pitz Good Technology	Minor Edits	Updates for Android 2.3, 3.0, 4.0, 4.1 in accordance with IG G.5

TABLE OF CONTENTS

1. Introduction

1.1 Purpose

The FIPSCrypto on Windows Mobile cryptographic module is a Software Dynamic Link Library (DLL) module that implements the Triple-DES, AES, SHA-1 and HMAC-SHA-1 algorithms. This non-proprietary Security Policy describes how the crypto module meets the security requirements of FIPS 140-2 Level 1 and how to securely operate the module.

This document also contains the details regarding crypto module required for meeting FIPS 140-2 Level 1 design assurance level 3 certification.

The cryptographic module enables Good products to securely establish a continuously synchronized wireless connection to corporate systems. Users can instantly access up-to-date corporate email; secure attachments, contacts, calendar, notes and tasks, and other information when traveling.

1.2 References

For more information on Good Technology and the Good products visit http://www.good.com.

Detailed information on the FIPS140-2 standard can be found at the NIST web site, http://csrc.nist.gov/cryptval.

2. Cryptographic Module Specification

The FIPSCrypto on Windows Mobile, version 4.7.0.50906 is validated against FIPS 140-2 Level 1, Design Assurance Level-3 to run on Windows CE 5.2 devices. The cryptographic module is a software module. The module was tested on a Samsung SGH-i907 device running a Windows CE operating system version 5.2. The module is classified as a multi-chip standalone module. The logical cryptographic boundary contains the software modules that comprise the FIPSCrypto dynamic link library. The physical boundary of the module is defined as the enclosure of the handheld on which the module executes.

3. Cryptographic Module Ports and Interfaces

The physical ports to the cryptographic module are standard I/O ports found on the handheld device such as a USB port, wireless infrared radio, and Graphical Display controller. The logical interface to the module is an Application Programming Interface (API). The function calls, that represent the services provided by the module, act as the Control Input Interface. The parameters to the API act as the Data Input Interface. The parameters returned from the API act as the Data Output Interface. The Status Output interface is the error code and return values provided by each function in the API.

Interface	Logical Interface	Physical Port
Data Input	Parameters to the API	Wireless Infrared Radio, Key Pad controller, Graphical Display Controller, USB Port, SD slot port, microphone port, camera port
Data Output	Parameters returned from the API	Wireless Infrared Radio, Key Pad Controller, Graphical Display Controller, USB port, SD slot port, speaker/earpiece/headset port
Control Input	Exported API calls	Key Pad Controller, button controller, USB port
Status Output	Error code and return values provided by each function in the API	Wireless Infrared Radio, Graphical Display Controller, speaker/earpiece/headset port
Power	N/A	Battery Port

Table 1 Ports And Interface Mapping

Interface	Parameters
Data Input	Key, Key Length, Algorithm Context, Plain text, Cipher-text, Encode/Decode flag, IV, IV Length, Plain text Length, Cipher-text Length, Padding Mode, Counter, Counter length, Hash Input Data, Hash Input Data Length, Num Bytes, Buffer size
Data Output	Cipher-text block, Algorithm Context, Plain text block, Context, Cipher text, Cipher-text Len, Plaintext, Plaintext Len, Digest, MAC value
Control Input	Aes_enc_key, Aes_enc_blk, Aes_dec_Key, Aes_dec_blk, SetKey, SetIV, SetCtr, Encode, Decode, getOutputLen, A_DES_EDE3_CBCEncryptInit, A_DES_EDE3_CBCEncryptyUpdate, A_DES_EDE3_CBCEncryptFinal, A_DES_EDE3_CBCDecryptInit, A_DES_EDE3_CBCDecryptUpdate, A_DES_EDE3_CBCDecryptFinal, A_SHAInit, A_SHAUpdate, A_SHAFinal, A_SHACopyContext, SetKey, GetMAC, GetMAC_N

Status Output	Getfipsenabled, Getfipstestsrun, Getfipstestspassed CRYPTOERR_OK, CRYPTOERR_INVALIDENCODEKEY, CRYPTOERR_INVALIDDECODEKEY, CRYPTOERR_INVALIDKEY, CRYPTOERR_INVALIDDATA, CRYPTOERR_INVALIDIV, CRYPTOERR_INVALIDPADDING, CRYPTOERR_ENCODEFAIL, CRYPTOERR_DECODEFAIL, CRYPTOERR_INVALIDCTR, CRYPTOERR_BUFFERTOOSMALL, CRYPTOERR_FAIL, CRYPTOERR_INVALIDHMACKEY, CRYPTOERR_CANCEL, AE_OUTPUT_LEN, AE_INPUT_LEN

Table 2. Interface and Parameter Mapping

4. Roles, Services and Authentication

4.1 Roles

The cryptographic module is a single operator software module that supports two authorized roles.

Roles
User Role
Crypto-Officer Role

Table 3. Roles

4.1.1 The Crypto-Officer Role

The operator takes on the role of a Crypto-Officer to perform tasks like, module installation and zeroization of the module. Other tasks performed by the Crypto-Officer include key entry, initiate the power-on self-tests on demand and check the status of the cryptographic module. The Crypto-Officer role has authorized access to the Triple-DES, AES, SHA-1 and HMAC-SHA-1 algorithms.

4.1.1.1 The Crypto-Officer Guide

The Good OTA(over the air) Software is used by the Crypto-Officer to install the cryptographic module onto the handheld device in a secure environment via the HTTPS download and secure OTA process. The Crypto-Officer starts up the OTA software, gives PIN and starts download. The OTA application starts the software installation process that copies the cryptographic module onto the handheld. Keys are installed onto the handheld as a part of this process. Upon completion of the installation process the module performs its power-on self-tests and enters an initialized state or error state. The Crypto-Officer can then request services from the module. The Crypto-Officer has the exclusive rights to perform Key Entry operations.

.

4.1.2 The User Role

An operator can assume the User Role and access the cryptographic algorithms provided in the module, which are AES, Triple-DES, SHA-1 and HMAC-SHA-1.

4.1.2.1 The User Guide

The User can request services from the cryptographic module using the module's Logical interface. The User Role has authorized access to the Triple-DES, AES, SHA-1 and HMAC-SHA-1 algorithms. The User can also initiate self-tests and check the status of the module. The cryptographic module provides information about the status of a requested operation to the user through the Status Output Interface. The following status codes are defined for the module.

Status Codes	**Information**
CRYPTOERR_OK	Operation completed successfully.
CRYPTOERR_INVALIDENCODEKEY	The key used for performing encryption operations is invalid.
CRYPTOERR_INVALIDDECODEKEY	The key used for performing decryption operations is invalid.
CRYPTOERR_INVALIDKEY	The key used for performing cryptographic operations is invalid.
CRYPTOERR_INVALIDDATA	The input data passed to the cryptographic module is invalid.
CRYPTOERR_INVALIDIV	The Initialization Vector input to the module is invalid.
CRYPTOERR_INVALIDPADDING	The padding of the encrypted blob is invalid.
CRYPTOERR_ENCODEFAIL	The encryption operation failed.
CRYPTOERR_DECODEFAIL	The decryption operation failed.
CRYPTOERR_INVALIDCTR	The Counter value input to the module is invalid.
CRYPTOERR_BUFFERTOOSMALL	The size of the buffer passed to the module is too small to perform the requested operation.
CRYPTOERR_INVALIDHMACKEY	The key used by the HMAC-SHA-1 is invalid.
CRYPTOERR_CANCEL	The module is in an error state. Check if the power-on self-tests have passed.
CRYPTOERR_FAIL	The module is in an error state. Check if the power-on self-tests have passed.
AE_CANCEL	The module is in an error state. Check if the power-on self-tests have passed.
AE_OUTPUT_LEN	The size of the buffer passed to the module is too small to perform the requested operation.
AE_INPUT_LEN	The size of the input data is invalid.
FIPS enabled : True	The module is in the FIPS mode and sets its FIPSenabled flag to True.

FIPS tests run : True	The module performs its self-tests and sets the FIPStestsrun flag to True.
FIPS tests passed : True	The module has successfully passed the FIPS self tests and sets its FIPStestspassed flag to true.

Table 4. Status Codes

The operator of the module can also determine its status from the debugger screen. Enter the debugger screen by typing ‘DEBUG’ on the keypad and then type ‘fips’ on the command line and <ENTER>. Each of the modules’ API functions is tested and the results are printed to the screen.

4.2 Services

The services provided by the cryptographic module are listed in the following table.

Services	**Cryptographic Keys and CSPs**	**Role** (CO, User, Both)	**Access** (R/W/X)
AES Encryption	AES secret Key	CO, User	X
AES Encryption: Key Entry	AES secret Key	CO	X
AES Decryption	AES secret Key	CO, User	X
AES Decryption: Key Entry	AES secret Key	CO	X
Triple-DES Encryption	Triple-DES secret Key	CO, User	X
Triple-DES Encryption: Key Entry	Triple-DES secret Key	CO	X
Triple-DES Decryption	Triple-DES secret Key	CO, User	X
Triple-DES Decryption: Key Entry	Triple-DES secret Key	CO	X
SHA-1 Hashing	N/A	CO, User	X
HMAC-SHA-1	HMAC-SHA-1 Key	CO, User	X
HMAC-SHA-1: Key Entry	HMAC-SHA-1 Key	CO	X
Show Status	N/A	CO, User	X
Perform Self tests	N/A	CO, User	X

Table 5. Services, Roles, Access

The following table presents a mapping of each cryptographic service provided by the module to its logical interface and the role assumed by the operator of the module to request those services.

Service	**Logical Interface**	**Role**

AES Encryption	aescrypt.c : aes_enc_blk aescbc.cpp : Encode aescbc.cpp : SetIV aescbc.cpp : getContext aesctr.cpp : SetCtr aesctr.cpp : getContext aesctr.cpp : Encode aesctr.cpp : getOutputLen	User, CO
AES Encryption: Key Entry	aescrypt.c : aes_enc_key aescbc.cpp : SetKey aesctr.cpp : SetKey	CO
AES Decryption	aescrypt.c : aes_dec_blk aescbc.cpp : Decode aescbc.cpp : SetIV aescbc.cpp : getContext aesctr.cpp : SetCtr aesctr.cpp : getContext aesctr.cpp : Encode aesctr.cpp : getOutputLen	User, CO
AES Decryption: Key Entry	aescrypt.c : aes_dec_key aescbc.cpp : SetKey aesctr.cpp : SetKey	CO
Triple-DES Encryption: Key Entry	desedee.cpp : A_DES_EDE3_CBCEncryptInit	CO
Triple-DES Encryption	desedee.cpp : A_DES_EDE3_CBCEncryptUpdate desedee.cpp : A_DES_EDE3_CBCEncryptFinal	User, CO
Triple-DES	deseded.cpp : A_DES_EDE3_CBCDecryptInit	CO
Decryption: Key Entry		
Triple-DES Decryption	deseded.cpp : A_DES_EDE3_CBCDecryptUpdate deseded.cpp : A_DES_EDE3_CBCDecryptFinal	User, CO
SHA-1 Hashing	gdsha.cpp : A_SHAInit gdsha.cpp : A_SHAUpdate gdsha.cpp : A_SHAFinal gdsha.cpp : A_SHACopyContext	User, CO
HMAC-SHA-1	Sha1HMAC.cpp : GetMAC Sha1HMAC.cpp : GetMAC_N	User, CO
HMAC-SHA-1: Key Entry	Sha1HMAC.cpp : SetKey	CO
Show Status	FipsCryptoPPC.cpp : getfipsenabled FipsCryptoPPC.cpp : getfipstestspassed FipsCryptoPPC.cpp : getfipstestsrun	User, CO
Self Tests	FipsCryptoPPC.cpp : InitializeFips	User, CO

Table 6. Services And Logical Interface Mapping

4.2.1 Approved Mode Of Operation

The module only provides an Approved Mode Of Operation. No special configuration is required to operate the module in a FIPS 140-2 mode. In this mode all authorized roles can call the FIPS 140-2 approved algorithms and services.

4.3 Authentication

The cryptographic module is validated at FIPS 140-2 Level 1 and does not provide role authentication for the authorized roles. The operator assumes these roles implicitly when invoking these services.

5. Physical Security

The cryptographic module is a software module that operates on the Windows CE 4.2 and/or 5.2 platform. The Windows CE 4.2 and 5.2 handheld devices use production grade components.

6. Operational Environment

The FIPS Cypto on Windows Mobile module was tested on Windows CE 5.2 using an ARM-based processor. Good Technology further affirms that the module will operate on any processor supporting Windows CE in accordance with FIPS 140-2 Implementation Guidance G.5. Windows CE is commercially shipped as Windows Mobile 5.0/6.0/6.1 and 6.5.

Please **NOTE:** Good's crypto library version number (4.7.0.50906) is unique and is no way relevant to the Good version number or Windows Mobile Operating System version that Good resides on.

As per FIPS 140-2 IG G.5 implementation guidance, Good affirms that for the iOS operating system 3.0, 3.1, 3.2, 4.0, 4.1, 4.2, 4.3, 5.0, 5.1 and 6.0 for the cryptographic implementation Good uses the same cryptographic source code that was used for Windows Mobile 5.0/6.0/6.1 and 6.5 operating systems.

As per FIPS 140-2 IG G.5 implementation guidance, Good affirms that for the Android operating system 1.6, 2.0, 2.1, 2.2, 2.3, 3.0, 4.0, and 4.1 for the cryptographic implementation Good uses the same cryptographic source code that was used for Windows Mobile 5.0/6.0/6.1 and 6.5 operating systems.

As per FIPS 140-2 IG G.5 implementation guidance, Good affirms that for the Windows Phone operating system 7.5, for the cryptographic implementation. Good uses the same cryptographic source code that was used for Windows Mobile 5.0/6.0/6.1 and 6.5 operating systems.

7. Cryptographic Key Management

7.1 Key Generation

The cryptographic module does not perform key generation.

7.2 Key Input/Output

The keys are electronically input into the module in plain-text form by the Crypto-Officer. Keys are not output from the module.

7.3 Key Storage

The module does not provide persistent storage for the keys used by the algorithms. The HMAC-SHA-1 key used for the integrity check is hard-coded into the module's executable code.

7.4 Key Zeroization

The keys are stored in memory on the device during the execution of an encryption/decryption or HMAC-SHA-1 calculation. At the completion of the calculation, the keys are zeroized. The other key is the HMAC-SHA-1 key used to perform the integrity check. The operator can zeroize this key by hard resetting the device on which the cryptographic module is operating.

7.5 Cryptographic Algorithms

The algorithms implemented by this module are listed below.

Algorithm	Certificate Number
Triple-DES (Triple Data Encryption Standard)	# 879
AES (Advanced Encryption Standard)	# 1219
SHA-1 (Secure Hash Algorithm)	# 1122
HMAC-SHA-1 (Keyed-Hashing Message Authentication Code)	# 712

8. EMI/EMC

The cryptographic module is a software module. The module runs on Windows CE 5.2 devices. The tested device meets applicable Federal Communication Commission (FCC) Electromagnetic Interference and Electromagnetic Compatibility requirements for business use.

9. Self Tests

Power On Tests

The cryptographic module performs algorithmic self-tests at startup time to ensure that the module is functioning properly. It also performs an integrity check using an approved HMAC-SHA-1 algorithm to validate the integrity of the module. These tests are initiated without user intervention at startup time or can be initiated by the user by restarting the device. The self-tests consist of a set of known answer tests to validate the working of the AES, Triple-DES, SHA-1 and HMAC-SHA-1 algorithms.

10. Mitigation of Other Attacks

The module is not designed to mitigate any other attacks.

11. Secure Operation

A configuration management system is set up using CVS (Concurrent Versioning System) to identify each component of the cryptographic module including documentation using a unique identification number. The Crypto-Officer installs the cryptographic module in FIPS 140-2 mode in a secure environment. The module implements only FIPS 140-2 approved algorithms and hence all cryptographic services provided by the module are FIPS 140-2 compliant. All the critical security functions performed by the module are tested at start-up or on demand. The module's integrity is also tested to prevent tampering, using an approved HMAC-SHA-1 algorithm.

12. Design Assurance Level 3 Details

12.1 Procedures for maintaining security while distributing and delivering versions of the cryptographic modules to authorized operators

The FIPSCrypto.dll module is bundled as part of other files used in providing GMM (Good Mobile Messaging) and/or Good Mobile Application software on device. These files are put into self extracting installation file called CAB file which is downloaded and extracted over the air by custom installer.

The end user follows the steps outlined below to install the cryptographic module:

1. User gets mail from admin with a secured PIN mentioned in email.
2. Using a browser on device, User goes to https://get.good.com/ and downloads OTA stub. Once downloaded the OTA stub will launch automatically.
3. The OTA stub will ask for PIN and start downloading the CAB file once the PIN is verified.
4. OTA stub extracts the CAB file, the extracted files will include the FIPSCrypto.dll along with other files.

The delivery of content from server is done over SSL connection and once downloaded the OTA agent does security checks to ensure that the package is not corrupted or modified.

The sequence below explains the internal security operations involved in ensuring integrity of downloaded modules:

1. All files including the FIPSCrypto.dll are packaged together in a cab during build time.
2. During build process the Cab is signed by Good private key.
3. The Checksum of cab is computed.
4. Cab is M2M signed.

5. The signed Cab and checksum is uploaded to the publicly hosted webstore .
6. Client side Installer downloads the cab and verifies the checksum along with signature
7. Installer executes the cab for GMM installation
8. Windows mobile, checks the M2M signature of the cab.

The above sequences foolproofs and validates that security is ensured by GOOD while distributing and delivering versions of the cryptographic modules to authorized operators ***through a process that combines*** SSL, signing of CAB files and verification of check sum when downloaded.