



## **Federal Information Processing Standard (FIPS) 140-2**

### **Good Technology**

Good FIPSCrypto

FIPS 140-2 Non-Proprietary Security Policy

January 05, 2007

©Good Technology, Inc.2006. All rights reserved.

**Good, Good Technology, and the Good logo are trademarks of Good Technology, Inc. in the United States and/or other countries. Good Technology, Inc., and its products are not related to, sponsored by, or affiliated with Research In Motion Limited. All other trademarks and service marks contained herein are the property of their respective owners.**

**This document maybe freely reproduced and distributed whole and intact including this Copyright Notice**



## DOCUMENT VERSION CONTROL

VERSION	DATE	AUTHOR (S)	DESCRIPTION	REASON FOR CHANGE
1.0	11 Jan 2004	Prathaban Selvaraj Good Technology	Initial Version	
1.1	28 Aug 2006	Daphne Won Good Technology	Made changes related to adding Symbian 9.1.  Removed unnecessary references to Desktop Software	
1.2	12 Sep 2006	Daphne Won Good Technology	Updates based on feedback from Ravi Iyers (Security Product Mgr at Good)	
1.3	15 Sep 2006	Daphne Won Good Technology	Main changes were made on Sections 1 and 2. Section 7.5 (Cryptographic Algorithms) was added.	
1.4	13 October 2006	Daphne Won Good Technology	Updated configuration management software to be Performe.	
1.5	3 November 2006	Daphne Won Good Technology	In Table 6, FIPSCryptoSymbian.cpp is used for Symbian 9.1	
1.6	13 December 2006	Daphne Won Good Technology	Updated the Symbian algorithm certificate numbers	
1.7	05 January 2007	Daphne Won Good Technology	Added original version of the cert	



## TABLE OF CONTENTS

<b>1. INTRODUCTION.....</b>	<b>7</b>
1.1 PURPOSE .....	7
ENABLES SOLUTIONS FROM GOOD TECHNOLOGY THAT REQUIRES ENCRYPTION.....	7
1.2 REFERENCES .....	7
<b>2. CRYPTOGRAPHIC MODULE SPECIFICATION.....</b>	<b>7</b>
<b>3. CRYPTOGRAPHIC MODULE PORTS AND INTERFACES.....</b>	<b>8</b>
<b>4. ROLES, SERVICES AND AUTHENTICATION.....</b>	<b>9</b>
4.1 ROLES.....	9
4.1.1 <i>The Crypto-Officer Role</i> .....	9
4.1.2 <i>The User Role</i> .....	10
4.2 SERVICES .....	11
4.2.1 <i>Approved Mode Of Operation</i> .....	13
4.3 AUTHENTICATION .....	13
<b>5. PHYSICAL SECURITY .....</b>	<b>13</b>
<b>6. OPERATIONAL ENVIRONMENT.....</b>	<b>14</b>
<b>7. CRYPTOGRAPHIC KEY MANAGEMENT .....</b>	<b>14</b>
7.1 KEY GENERATION.....	14
7.2 KEY INPUT/OUTPUT.....	14
7.3 KEY STORAGE.....	14
7.4 KEY ZEROIZATION.....	14
7.5 CRYPTOGRAPHIC ALGORITHMS .....	15
<b>8. EMI/EMC .....</b>	<b>15</b>
<b>9. SELF TESTS .....</b>	<b>15</b>
<b>10. MITIGATION OF OTHER ATTACKS .....</b>	<b>16</b>
<b>11. SECURE OPERATION.....</b>	<b>16</b>

# **1. Introduction**

## **1.1 Purpose**

The Good FIPSCrypto 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.

The cryptographic module enables solutions from Good Technology that require encryption. The code which constitutes the cryptographic module is common to Windows CE and Symbian platforms.

Enables solutions from Good Technology that requires encryption.

## **1.2 References**

For more information on Good Technology and the GoodLink product 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 Good FIPSCrypto cryptographic module is validated against FIPS 140-2 Level 1 with version 20040220 to run on the Windows CE operating system and version 4.9.1 to run on the Symbian operating system. 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 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.

<b>Interface</b>	<b>Logical Interface</b>	<b>Physical Port</b>
Data Input	Parameters to the API	Wireless Radio, Key Pad controller, Graphical Display Controller, USB Port, IrDA port, SD slot port, microphone port, Bluetooth, WiFi
Data Output	Parameters returned from the API	Wireless 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, IrDA port
Status Output	Error code and return values provided by each function in the API	Wireless Radio, Graphical Display Controller, speaker/earpiece/headset port, LED
Power	N/A	Battery port

Table 1. Ports And Interface Mapping

<b>Interface</b>	<b>Parameters</b>
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_CBCEncryptUpdate, 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.

<b>Roles</b>
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 Crypto Officer installs the cryptographic module onto the handheld device in a secure environment. 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 Triple DES, AES, 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.

<b>Status Codes</b>	<b>Information</b>
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.

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	Role (CO, User, Both)	Access (R/W/X)
AES Encryption	CO, User	X
AES Encryption: Key Entry	CO	X
AES Decryption	CO, User	X
AES Decryption: Key Entry	CO	X
Triple DES Encryption	CO, User	X
Triple DES Encryption: Key Entry	CO	X
Triple DES Decryption	CO, User	X
Triple DES Decryption: Key Entry	CO	X
SHA-1 Hashing	CO, User	X
HMAC-SHA-1	CO, User	X
HMAC-SHA-1: Key Entry	CO	X
Show Status	CO, User	X

Perform Self tests	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.

<b>Service</b>	<b>Logical Interface</b>	<b>Role</b>
AES Encryption	aescript.c : aes_enc_blk aesbc.cpp : Encode aesbc.cpp : SetIV aesbc.cpp : getContext aesctr.cpp : SetCtr aesctr.cpp : getContext aesctr.cpp : Encode aesctr.cpp : getOutputLen	User, CO*
AES Encryption: Key Entry	aescript.c : aes_enc_key aesbc.cpp : SetKey aesctr.cpp : SetKey	CO
AES Decryption	aescript.c : aes_dec_blk aesbc.cpp : Decode aesbc.cpp : SetIV aesbc.cpp : getContext aesctr.cpp : SetCtr aesctr.cpp : getContext aesctr.cpp : Encode aesctr.cpp : getOutputLen	User, CO
AES Decryption: Key Entry	aescript.c : aes_dec_key aesbc.cpp : SetKey aesctr.cpp : SetKey	CO
TDES Encryption: Key Entry	desedee.cpp : A_DES_EDE3_CBCEncryptInit	User
TDES Encryption	desedee.cpp : A_DES_EDE3_CBCEncryptUpdate desedee.cpp : A_DES_EDE3_CBCEncryptFinal	User, CO
TDES Decryption: Key Entry	deseded.cpp : A_DES_EDE3_CBCCDecryptInit	CO
TDES Decryption	deseded.cpp : A_DES_EDE3_CBCCDecryptUpdate deseded.cpp : A_DES_EDE3_CBCCDecryptFinal	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	For Windows CE: FipsCryptoPPC.cpp : getfipsenabled FipsCryptoPPC.cpp : getfipstestspassed FipsCryptoPPC.cpp : getfipsteststrun For Symbian: FipsCryptoSymbian.cpp : getfipsenabled FipsCryptoSymbian.cpp : getfipstestspassed FipsCryptoSymbian.cpp : getfipsteststrun	User, CO
Self Tests	N/A	User, CO

Table 6. Services And Logical Interface Mapping

\*CO = Crypto Officer Role

#### 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 evaluated 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 Microsoft Windows CE 4.2 and Symbian 9.1 platforms. The Windows CE 4.2 and Symbian 9.1 handheld devices use production grade components.

## **6. Operational Environment**

The operational environment consists of Windows CE 4.2 and Symbian 9.1, pre-emptive multi-tasking operating systems running on ARM-based processors.

## **7. Cryptographic Key Management**

### **7.1 Key Generation**

The cryptographic module does not perform key generation.

### **7.2 Key Input/Output**

The keys are 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 stored in the module.

### **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 removing the user data from the handheld. Selecting this option for a handheld running a FIPS certified crypto module would cause the module to be zeroized. The operator may also zeroize this key by hard resetting the device on which the cryptographic module is operating. On Symbian 9.1, the operator may also zeroize this key by uninstalling the module.

## 7.5 Cryptographic Algorithms

The algorithms implemented by this module for Windows CE are listed below.

Algorithm	Certificate Number
Triple-DES (Triple Data Encryption Standard)	# 240
AES (Advanced Encryption Standard)	# 134
SHS (Secure Hash Algorithm) (SHA-1)	# 217
HMAC-SHA-1 (Keyed-Hashing Message Authentication Code)	# 217 (vendor certified)

The algorithms implemented by this module for Symbian are listed below.

Algorithm	Certificate Number
Triple-DES (Triple Data Encryption Standard)	# 491
AES (Advanced Encryption Standard)	# 477
SHS (Secure Hash Algorithm) (SHA-1)	# 545
HMAC-SHA-1 (Keyed-Hashing Message Authentication Code)	# 234

## 8. EMI/EMC

The cryptographic module is a software module. The module runs on the Windows CE 4.2 and Symbian 9.1 devices. The tested devices meet 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 resetting the device. The self-tests consist of a set of known answer tests to validate the working of the Triple DES, AES, 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 Perforce 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.