McAfee, Inc.
McAfee Endpoint Encryption Client Windows Cryptographic Module 1.0

and McAfee Endpoint Encryption Client Preboot Cryptographic Module 1.0

FIPS 140-2 Non-Proprietary Security Policy

Level 1 Validation

Document revision 020, January 2012

Prepared for McAfee, Inc. by

McAfee, Inc.
2821 Mission College Blvd.
Santa Clara, CA 95054
888.847.8766
www.mcafee.com

Rycombe Consulting Limited
http://www.rycombe.com
+44 1273 476366

Contents

1 Introduction 4
1.1 Identification 4
1.2 Purpose 4
1.3 References 4
1.4 Document Organization 4
1.5 Document Terminology 5
2 McAfee Endpoint Encryption Client 5
2.1 Overview 6
2.2 Module Specification 6
2.2.1 Hardware, Software and Firmware components 6
2.2.2 Cryptographic Boundary 6
2.2.3 Scope of Evaluation 8
2.2.4 Cryptographic Algorithms 8
2.2.5 Components excluded from the security requirements of the standard 9
2.3 Physical ports and logical interfaces 9
2.4 Roles, Services and Authentication 10
2.4.1 Roles 10
2.4.2 Services 10
2.4.3 Authentication 14
2.5 Physical Security 14
2.6 Operational Environment 14
2.7 Cryptographic Key Management 15
2.7.1 Random Number Generators 15
2.7.2 Key Generation 15
2.7.3 Key Table 15
2.7.4 Key Destruction 17
2.7.5 Access to Key Material 18
2.8 Self-Tests 19
2.8.1 Power-up self-tests 19
2.8.2 Conditional self-tests 19
2.9 Design Assurance 19
2.10 Mitigation of Other Attacks 20
3 FIPS Mode of Operation 20
3.1 Deploying in FIPS mode 21

Figures

Figure 1 Block Diagram of the Cryptographic Boundary (Pre-boot environment) 7
Figure 2 Block Diagram of the Cryptographic Boundary (Windows environment) 7
Figure 3 Security Level specification per individual areas of FIPS 140-2 8
Figure 4 Approved Algorithms 9
Figure 5 Module Interfaces 10

Figure 6 Roles....................10
Figure 7 User Services....................13
Figure 8 Crypto Officer Services....................14
Figure 9 Module Cryptographic Keys and CSPs....................15
Figure 10 Key Table part 1....................16
Figure 11 Key Table part 2....................17
Figure 12 Access to keys by services....................18
Figure 13 Power-up self-tests....................19
Figure 14 Conditional self-tests....................19

1 Introduction

This section identifies the cryptographic module; describes the purpose of this document; provides external references for more information; and explains how the document is organized.

1.1 Identification

Module Name	McAfee, Inc. McAfee Endpoint Encryption Client Windows Cryptographic Module McAfee, Inc. McAfee Endpoint Encryption Client Preboot Cryptographic Module
Module Version	1.0
Software Version	6.1.3

1.2 Purpose

This is the non-proprietary FIPS 140-2 Security Policy for the McAfee Endpoint Encryption Client Cryptographic Module, also referred to as "the module" within this document. This Security Policy details the secure operation of McAfee Endpoint Encryption Client Cryptographic Module as required in Federal Information Processing Standards Publication 140-2 (FIPS 140-2) as published by the National Institute of Standards and Technology (NIST) of the United States Department of Commerce.

1.3 References

For more information on McAfee Endpoint Encryption please visit:
http://www.mcafee.com/us/products/data-protection/endpoint-encryption.aspx.
For more information on NIST and the Cryptographic Module Validation Program (CMVP), please visit http://csrc.nist.gov/groups/STM/cmvp/index.html.

1.4 Document Organization

This Security Policy document is one part of the FIPS 140-2 Submission Package. This document outlines the functionality provided by the module and gives high-level details on the means by which the module satisfies FIPS 140-2 requirements. With the exception of this Non-Proprietary Security Policy, the FIPS 140-2 Submission documentation may be McAfee, Inc. proprietary or otherwise controlled and releasable only under appropriate non-disclosure agreements. For access to these documents, please contact McAfee, Inc.

The various sections of this document map directly onto the sections of the FIPS 140-2 standard and describe how the module satisfies the requirements of that standard.

1.5 Document Terminology

TERM	DESCRIPTION
AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions. Seven instructions for accelerating different sub-steps of the AES algorithm included in some Intel and AMD microprocessors.
API	Application Programming Interface
CAVP	Cryptographic Algorithm Validation Program
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameters
DLL	Dynamic Link Library
DLM	Dynamic Link Module (a type of DLL used in the Pre-boot environment)
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Algorithm
FIPS	Federal Information Processing Standard
GPC	General Purpose Computer
GUI	Graphical User Interface
IPC	Inter-process communication
MBR	Master Boot Record
McAfee ePO	McAfee ePolicy Orchestrator: A McAfee software installation to allow configuration and management of a McAfee Endpoint Encryption deployment
OS	Operating System
Pre-boot environment	The operating environment of a GPC before the operating system is loaded
RSA	An algorithm for public-key cryptography. Named after Rivest, Shamir and Adleman who first publicly described it.
SHA	Secure Hash Algorithm
SP	Security Policy
Storage Media	Any media for which Cryptographic Module protection in the form of data encryption is required. Storage Media include internal and external hard drives, memory sticks and floppy disks.
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security
XML	Extensible Markup Language

2 McAfee Endpoint Encryption Client

This section provides the details of how the module meets the FIPS 140-2 requirements.

2.1 Overview

The module provides cryptographic services to McAfee Endpoint Encryption products. These endpoint products are managed by McAfee Electronic Policy Orchestrator (ePO). Among other functions, ePO deploys and manages the endpoint software, including the cryptographic module.

The module is packaged as a DLL when running under Microsoft Windows or as a DLM when operating in the pre-boot environment.

2.2 Module Specification

2.2.1 Hardware, Software and Firmware components

There are no specific hardware or firmware requirements for the module. The module is a software-only module which resides on a General Purpose Computer (see Figure 2).

It is packaged as two distinct binary images, depending on the operating environment in which it operates:

File Name	Operating Environment
MfeEpeCrypto.dll	Microsoft Windows
MfeEpeCrypto.dlm	Pre-boot environment

These two variants are based on a common core of functionality. The only differences in implementation functionality relate to the collection of the entropy used to seed the SP800-90 compliant DRBG and one AES implementation.

The pre-boot module uses entropy derived from user generated inputs such as mouse movements and keyboard key presses, whereas the Windows module uses entropy derived from network activity.

The low-level disk handler AES implementation is only applicable to the Windows environment.

As these two packages are independent of one another, they are effectively different modules. To clearly discriminate between the two variants, the Windows (DLL) variant is known as the McAfee Endpoint Encryption Client Windows Cryptographic Module 1.0, and the Preboot (DLM) variant is known as the McAfee Endpoint Encryption Client Preboot Cryptographic Module 1.0. However, except in the distinct areas clearly described above, the two modules are identical and in those cases, the term “the module” applies equally to either variant.

2.2.2 Cryptographic Boundary

The cryptographic boundary of the module is the case of the General Purpose Computer (GPC) on which it is installed. See Figure 2. The module is a software module running in a pre-boot or Windows operating environment on a general-purpose computer. The processor of this platform executes all software. All software components of the module are persistently stored within the device and, while executing, are stored in the device local RAM.

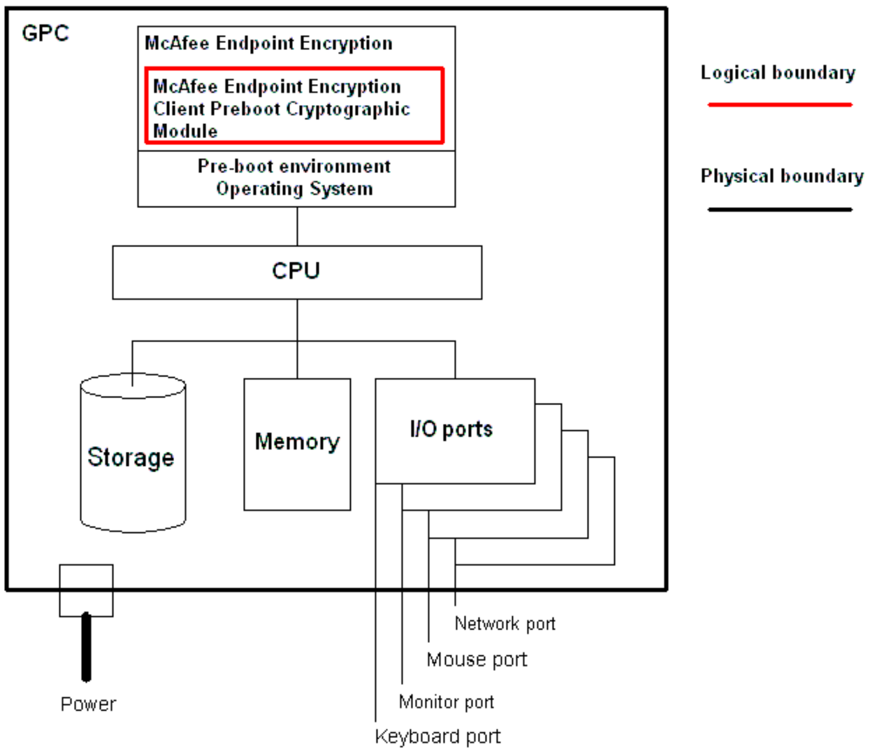

GPC
McAfee Endpoint Encryption
McAfee Endpoint Encryption Client Preboot Cryptographic Module
Pre-boot environment Operating System
CPU
Storage
Memory
I/O ports
Network port
Mouse port
Monitor port
Keyboard port
Power
Logical boundary
Physical boundary

Figure 1 Block Diagram of the Cryptographic Boundary (Pre-boot environment)

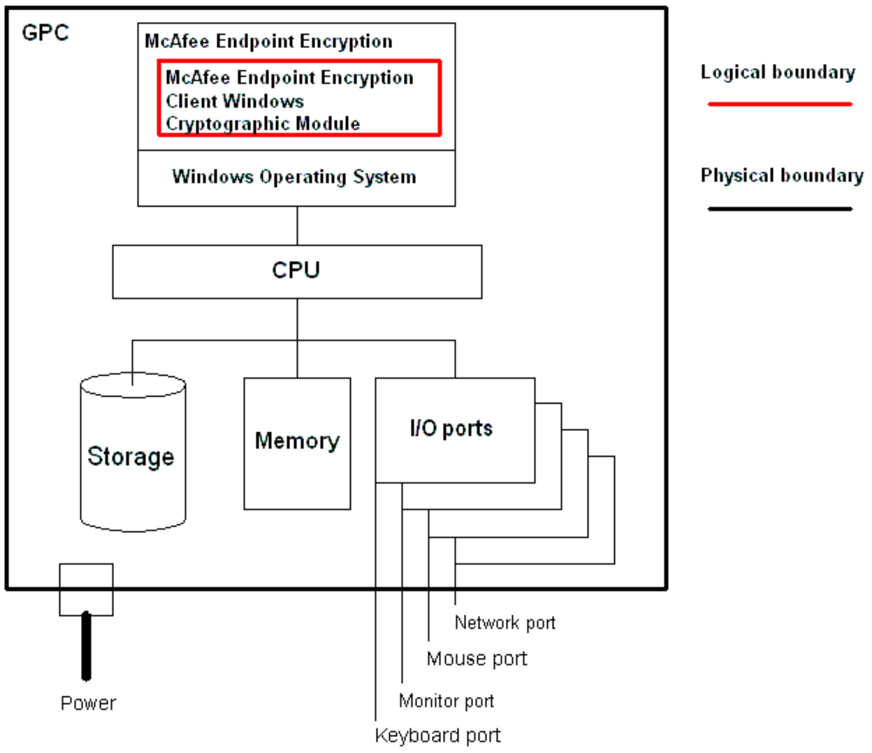


Figure 2 Block Diagram of the Cryptographic Boundary (Windows environment)

2.2.3 Scope of Evaluation

The cryptographic module meets the overall requirements applicable to Level 1 security of FIPS 140-2, with Design Assurance at Level 3.

SECURITY REQUIREMENTS SECTION	LEVEL
Cryptographic Module Specification	1
Module Ports and Interfaces	1
Roles, Services and Authentication	1
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	3
Mitigation of Other Attacks	N/A

Figure 3 Security Level specification per individual areas of FIPS 140-2

2.2.4 Cryptographic Algorithms

The following table provides details of the approved algorithms that are included within the module:

ALGORITHM TYPE	ALGORITHM	CAVP CERTIFICATE	USE
Hashing	SHA-1 SHA-256	#1653 (SHA-1 and SHA-256) #1654 (SHA-256 only)	SHA-1 is not used in the FIPS mode of operation. It is a constituent of the non-approved PKCS#5 algorithm. Constituent of DRBG and also used in the module integrity test.
Message authentication code	HMAC	#1124 #1125	Module integrity testing and in the random number generator.
Random number generation	HMAC SHA256 DRBG	#156	Symmetric and Asymmetric key generation
Symmetric key	AES-256 – CBC and CFB128. Encrypt and decrypt	#1881 #1882 #1883 (Windows only)	Service provided to encrypt and decrypt blocks of data. As it is a user service and so may also be used to encrypt private

ALGORITHM TYPE	ALGORITHM	CAVP CERTIFICATE	USE
			keys to provide key wrapping (cert #1881 only).

Figure 4 Approved Algorithms

Notes:

There are three implementations of AES-256 in the module, one to support each of the following:

- MFEi_crypto_sym_encryptor service
- EPEi_crypto_disk_algs service interface "get_interface", a copy of the MfeEEAlg module algorithms – this can run on processors with or without AES-NI capability. However, it will only use AES-NI instructions if run on AES-NI enabled processors.
- EPEi_crypto_disk_algs service interface "get_hook_code", a copy of the algorithm code to support the cryptographic algorithm requirements of the INT 13 handler (Windows only)

RSA encrypt/decrypt is used for key transport giving 112 bits of security strength. This service is non-approved but allowed.

The entropy used to seed the random number generator is an NDRNG that is a non-Approved algorithm used in FIPS mode.

The following non-approved algorithms are included within the module but are not used when the module is operating in FIPS mode:

- RC5 (RC5-12 and RC5-18).
- PKCS#5 (a password hashing algorithm using SHA-1), offered as a non-approved service to users of the module.
- A non-FIPS AES implementation.

2.2.5 Components excluded from the security requirements of the standard

There are no components excluded from the security requirements of the standard.

2.3 Physical ports and logical interfaces

The module is classified as a multi-chip standalone module for FIPS 140-2 purposes. The module's physical boundary is that of the device on which it is installed. The device shall be running a supported operating system (OS) and supporting all standard interfaces, including keys, buttons and switches, and data ports.

The module provides its logical interfaces via Application Programming Interface (API) calls. This logical interface exposes services (described in section 2.4.2) that the User and operating system utilize directly.

The logical interfaces provided by the module are mapped onto the FIPS 140-2 logical interfaces: data input, data output, control input, and status output as follows:

FIPS 140-2 Logical Interface	Module Mapping
Data Input	Parameters passed to the module via API calls
Data Output	Data returned from the module via API calls
Control Input	API Calls and/or parameters passed to API calls
Status Output	Information received in response to API calls
Power Interface	There is no separate power or maintenance access interface beyond the power interface provided by the GPC itself

Figure 5 Module Interfaces

2.4 Roles, Services and Authentication

2.4.1 Roles

The Cryptographic Module implements both a Crypto Officer role and a User role. Roles are assumed implicitly upon accessing the associated services. Section 2.4.2 summarizes the services available to each role.

Role	Description
Crypto Officer	The administrator of the module having full configuration and key management privileges.
User	General User of the module

Figure 6 Roles

2.4.2 Services

Most of the services provided by the module are provided via access to API calls using interfaces exposed by the module.

However, some of the services, such as power-up module integrity testing are performed automatically and so have no function API, but do provide status output.

Service	API	Description	Service Input	Service Output
Asymmetric encryption	MFEi_crypto_asym_encryptor	Provides RSA encryption and decryption.	Encryption: Public key and plaintext data passed in. Decryption: Private key and encrypted data	Encryption: Encrypted data passed out. Decryption: Plaintext data passed out.

SERVICE	API	DESCRIPTION	SERVICE INPUT	SERVICE OUTPUT
			passed in.	Service can also output algorithm status information.
	MFEi_crypto_asym_encryptor_var_pad	Provides RSA encryption but allows the padding to be specified.	Encryption: Public key and plaintext data passed in. Decryption: Private key and encrypted data passed in.	Encryption: Encrypted data passed out. Decryption: Plaintext data passed out. Service can also output algorithm status information.
Key generation	MFEi_crypto_asym_key_gen	Provides RSA key generation. The RSA keys are calculated from large numbers generated by the FIPS approved DRBG seeded by a non-approved NDRNG.	None	Service methods exist to generate a key pair and to separately output the public key and the private key.
	MFEi_crypto_asym_key_gen2	Provides RSA key generation but allows the keys to be encoded in various formats. The RSA keys are calculated from large numbers generated by the FIPS approved DRBG seeded by a non-approved NDRNG.	Format required for keys	Service methods exist to generate a key pair and to separately output the public key and the private key.
	MFEi_rand_source	Generates symmetric keys and accepts seed data to add to the random pool. Entropy data is used to	For seeding: Entropy data and a count of the number of entropy bytes:	For Seeding: None.

SERVICE	API	DESCRIPTION	SERVICE INPUT	SERVICE OUTPUT
		seed the DRBG which is used to generate the symmetric keys.	For generation: A count of the number of bytes required.	For generation: A block of random data.
Symmetric encryption	MFEi_crypto_enc_data	Utility class to hold encrypted data returned by the symmetric encryptor.	Data buffer, size of data and algorithm type ID	Data buffer, size of data, and algorithm type ID
	MFEi_crypto_sym_encryptor	Provides symmetric algorithm encryption and decryption. Supports AES256 (CBC and CFB128 modes).	Encryption: Algorithm ID, key, plaintext data, Initial Value (IV) Decryption: Key, encrypted data, Initial Value (IV)	Encryption: Encrypted data Decryption: Decrypted data Status: Algorithm information and key size
	EPEi_crypto_disk_algs	Provides access to a copy of the MfeEEAlg symmetric disk encryption algorithms (statically linked into this, the MfeEpeCrypto module). Supports only FIPS AES 256 when the module is operated in FIPS mode. The RC5 legacy algorithm (both 12 round and 18 round) is supported when not operating in FIPS mode, as is a non-FIPS compliant AES implementation.	None	Status: Results of Disk Driver encryption algorithm self-tests. Interface pointer providing access to Disk Driver encryption algorithms.
Hashing	MFEi_crypto_digest_gen	Allows hashing of arbitrary data. Supports SHA1 and SHA256.	Data and algorithm	Digest of the data.
Self-tests	MFEi_crypto_self_tests	Runs all the KATs on the algorithms exported by MfeCryptoLib library.	None	Service outputs a Boolean result code, TRUE if all of the self-tests

Service	API	Description	Service Input	Service Output
				passed, otherwise FALSE.
	N/A	The power-up software integrity test is run automatically when the module is loaded and started. The continuous random number generator test is also run automatically.	None	If passes, writes the string "Passed" to the IntegrityStatus "MfeEpeCrypto. Dll" registry key, otherwise writes the string "Failed" to this registry key and initiates a stop error.
Show Status	N/A	Status is returned in response to individual service API calls and at the completion of the self-tests.	None	Module status.

Figure 7 User Services

Service	API	Description	Service Input	Service Output
Installation	N/A	At some point after the installation process, the product is activated once suitable policy has been received from ePO. The module is deployed to the client PC from ePO. User services are then used to generate keys and encrypt the endpoint GPC storage.	None	Installed module
Uninstallation	N/A	Prior to the uninstallation product, user services are used to decrypt the endpoint GPC services and then remove the endpoint software, including the cryptographic module software.	None	Uninstalled module
Key Zeroization	N/A	Keys are zeroized using the	None	All keys zeroized.

SERVICE	API	DESCRIPTION	SERVICE INPUT	SERVICE OUTPUT
		zeroization procedure. This is descried in section 2.7.4.		

Figure 8 Crypto Officer Services

2.4.3 Authentication

The module has been evaluated at FIPS 140-2 level 1 and no claims are made for authentication.

2.5 Physical Security

The Cryptographic Module is a software-only cryptographic module and therefore the physical security requirements of FIPS 140-2 do not apply.

2.6 Operational Environment

The Cryptographic Module has been tested on and found to be conformant with the requirements of FIPS 140-2 overall Level 1 on the following GPC operating systems:

- McAfee Endpoint Encryption Preboot OS running on Intel Core i3 without AES-NI
- McAfee Endpoint Encryption Preboot OS running on Intel Core i5 with AES-NI
- McAfee Endpoint Encryption Preboot OS running on Intel Core i7 with AES-NI
- Windows XP 32-bit running on Intel Core i3 without AES-NI
- Windows 7 64-bit running on Intel Core i3 without AES-NI
- Windows Vista 32-bit running on Intel Core i5 with AES-NI
- Windows Vista 64-bit running on Intel Core i7 with AES-NI
- Windows 7 32-bit running on Intel Core i5 with AES-NI
- Windows 7 64-bit running on Intel Core i7 with AES-NI

The module is also capable of running on the following platforms but has not been tested during this evaluation and no compliance is being claimed on these platforms:

- Windows Server 2008 (32-bit and 64-bit) with SP1
- Windows Server 2008 R2 (64-bit only)
- Windows 2003 Server (32-bit only) with SP2
- Windows 2003 Server R2 (32-bit only) with SP2

The cryptographic module runs in its own operating system threads. This provides it with protection from all other processes, preventing access to all keys, intermediate key generation values, and other CSPs.

The task scheduler and architecture of the operating system maintain the integrity of the cryptographic module.

The module supports only one single user and only one operator can have access to the device that contains the module at a time.

2.7 Cryptographic Key Management

2.7.1 Random Number Generators

The module contains an approved HMAC SHA256 SP800-90 approved DRBG.

2.7.2 Key Generation

Keys generated internally are generated by the HMAC SHA256 DRBG seeded by system entropy. Checks are made to ensure that the quality of the entropy remains high enough to be used to seed the DRBG.

The entropy comes from a promiscuous socket. This is used to provide sampling points for discrete high speed counters in the Windows and Preboot environments. Only the least significant bits of these counters are sampled and statistical tests produce results showing that this provides random data of sufficient entropy.

2.7.3 Key Table

The following tables list all of the keys and CSPs within the module, describe their purpose, and describe how each key is generated, entered and output, stored and destroyed.

KEY	PURPOSE
Data Key	To encrypt and decrypt data using the symmetric encryption services.
Public Key	To encrypt data using one of the asymmetric encryption services.
Private Key	To decrypt data previously encrypted by the Public Key.
HMAC DRBG CSPs: Key, V, seed and entropy	These are variables used internally by the HMAC DRBG that are required by Implementation Guidance 14.5 to be listed in the Cryptographic Module Security Policy document. There is no initial seed, but the algorithm is reseeded from a non-approved NDRNG used in FIPS mode.
HMAC-SHA-256 Integrity Key	A fixed and hard coded key used in the module integrity checking power-up self-test

Figure 9 Module Cryptographic Keys and CSPs

KEY	KEY LENGTH/STRENGTH	GENERATION/ESTABLISHMENT	STORAGE LOCATION
Data Key	AES 256 bit	Generated using Key generation service	Not stored within the module.
Public Key	RSA 2048 bit	Generated using Key generation	Not stored within the module.

		service	
Private Key	RSA 2048 bit	As per Public Key	Not stored within the module.
HMAC DRBG “Key” CSP	512 bits	Initial value of 64 bytes all set to “0x00”	Not stored within the module.
HMAC DRBG “V” CSP	512 bits	Initial value of 64 bytes all set to “0x01”	Not stored within the module.
HMAC DRBG seed and entropy CSPs	2048 bits	non-approved NDRNG	Not stored within the module.
HMAC-SHA-256 Integrity Key	256 bits	Fixed key	Not stored within the module.

Figure 10 Key Table part 1

Key	Are Keys supplied Encrypted or Plaintext?	Entry/Output	Destruction
Data Key	Plaintext	Used as key in symmetric encryption service.	Zeroized using the key zeroization service.
Public Key	Plaintext	Passed as a parameter to asymmetric encryption service.	Zeroized using the key zeroization service.
Private Key	Plaintext	Passed as a parameter to asymmetric encryption service.	Zeroized using the key zeroization service.
HMAC DRBG “Key” CSP	N/A	N/A	Zeroized using the key zeroization service.
HMAC DRBG “V” CSP	N/A	N/A	Zeroized using the key zeroization service.
HMAC DRBG seed and entropy CSPs	N/A	N/A	Zeroized using the key zeroization service.
HMAC-SHA-256 Integrity Key	N/A	N/A	Zeroized using the key zeroization service.

Figure 11 Key Table part 2

2.7.4 Key Destruction

All key material managed by the module can be zeroized using the key zeroization procedure. This requires uninstallation of the cryptographic module and reformatting the hard drive on which it was installed.

The CO should uninstall the module and then reformat the hard drive on which it was installed and overwrite it at least once. The operator should remain present during this process. This process meets the requirements of IG 7.9 for key zeroization.

Reformatting the hard drive will remove any encrypted or public keys from the hard disk.

In this way all key material and CSPs are zeroized. There are no user-accessible plaintext keys or CSPs in the module.

2.7.5 Access to Key Material

The following table shows the access that an operator has to specific keys or other critical security parameters when performing each of the services relevant to his/her role.

KEY	DATA KEY	PUBLIC KEY	PRIVATE KEY	HMAC DRBG KEY	HMAC DRBG V	INTEGRITY KEY
User Services						
Asymmetric encryption		WU	WU			
Key generation	R	R	R	U	U	
Symmetric encryption	RWU					
Hashing						
Self-tests						U
Show Status						
Crypto Office Services						
Installation	U					
Uninstallation	U					
Key Zeroization	W	W	W	W	W	W

Figure 12 Access to keys by services

Access Rights

Blank	Not Applicable
R	Read
W	Write
U	Use

Note: Key zeroization zeroes all keys and CSPs, this is a “write” operation in that all keys are overwritten with zeroes.

2.8 Self-Tests

The module implements both power-up and conditional self-tests as required by FIPS 140-2. The following two sections outline the tests that are performed.

2.8.1 Power-up self-tests

OBJECT	TEST
SHA-1	Known answer test
SHA-256	Known answer test
AES-256	A separate encryption and decryption known answer test for each of the AES implementations within the module
HMAC DRBG	Known answer test
Module software	HMAC SHA-256 Integrity Check

Figure 13 Power-up self-tests

Note: A known answer test for HMAC is not required because the module performs an HMAC-SHA-256-DRBG known answer test. The module performs a separate known answer test for each of its two SHA-256 implementations.

2.8.2 Conditional self-tests

EVENT	TEST	CONSEQUENCE OF FAILURE
Module requests a random number from the FIPS Approved SP800-90 DRBG	A continuous random number generator test	Random number is not generated and module enters an error state.
Entropy is supplied to the FIPS approved SP800-90 DRBG	A continuous random number generator test on the entropy NDRNG	Entropy is not added and module enters an error state
RSA key pair generation	Pairwise consistency test	Key is not generated and module enters an error state.

Figure 14 Conditional self-tests

2.9 Design Assurance

McAfee, Inc. employ industry standard best practices in the design, development, production and maintenance of the McAfee Endpoint Encryption product, including the FIPS 140-2 module.

This includes the use of an industry standard configuration management system that is operated in accordance with the requirements of FIPS 140-2, such that each configuration item that forms part of the module is stored with a label corresponding to the version of the module and that the module and all of its associated documentation can be regenerated from the configuration management system with reference to the relevant version number.

Design documentation for the module is maintained to provide clear and consistent information within the document hierarchy to enable transparent traceability between corresponding areas throughout the document hierarchy, for instance, between elements of this Cryptographic Module Security Policy (CMSP) and the design documentation.

Guidance appropriate to an operator's Role is provided with the module and provides all of the necessary assistance to enable the secure operation of the module by an operator, including the Approved security functions of the module.

Delivery of the Cryptographic Module to customers from the vendor is via the internet. When a customer purchases a license to use the Cryptographic Module software, they are issued with a grant number as part of the sales process. This is then used as a password to allow them to download the software that they have purchased. The delivery channel is protected using secured sockets. Once the Cryptographic Officer has downloaded the cryptographic module, it is his responsibility to ensure its secure delivery to the users that he is responsible for.

2.10 Mitigation of Other Attacks

The module does not mitigate any other attacks.

3 FIPS Mode of Operation

The module is set into FIPS mode by setting the FipsMode registry key to a non-zero value.

PATH	NAME	TYPE	VALUE
32-bit Windows: HKEY_LOCAL_MACHINE\ Software\ McAfee Endpoint Encryption\ Fips 64-bit Windows: HKEY_LOCAL_MACHINE\ Software\Wow6432Node\ McAfee Endpoint Encryption\ Fips	FipsMode	DWORD	0 = Not in FIPS mode Any non-zero value indicates FIPS mode operation.

When the module is not in FIPS mode:

- The power-up integrity checks are not performed.

- Seeding of the DRBG by a non-approved NDRNG is not required.
- The RC5 legacy algorithm can be used.
- The PKCS#5 algorithm may be used.
- RSA encryption may be used

When the module is in FIPS mode:

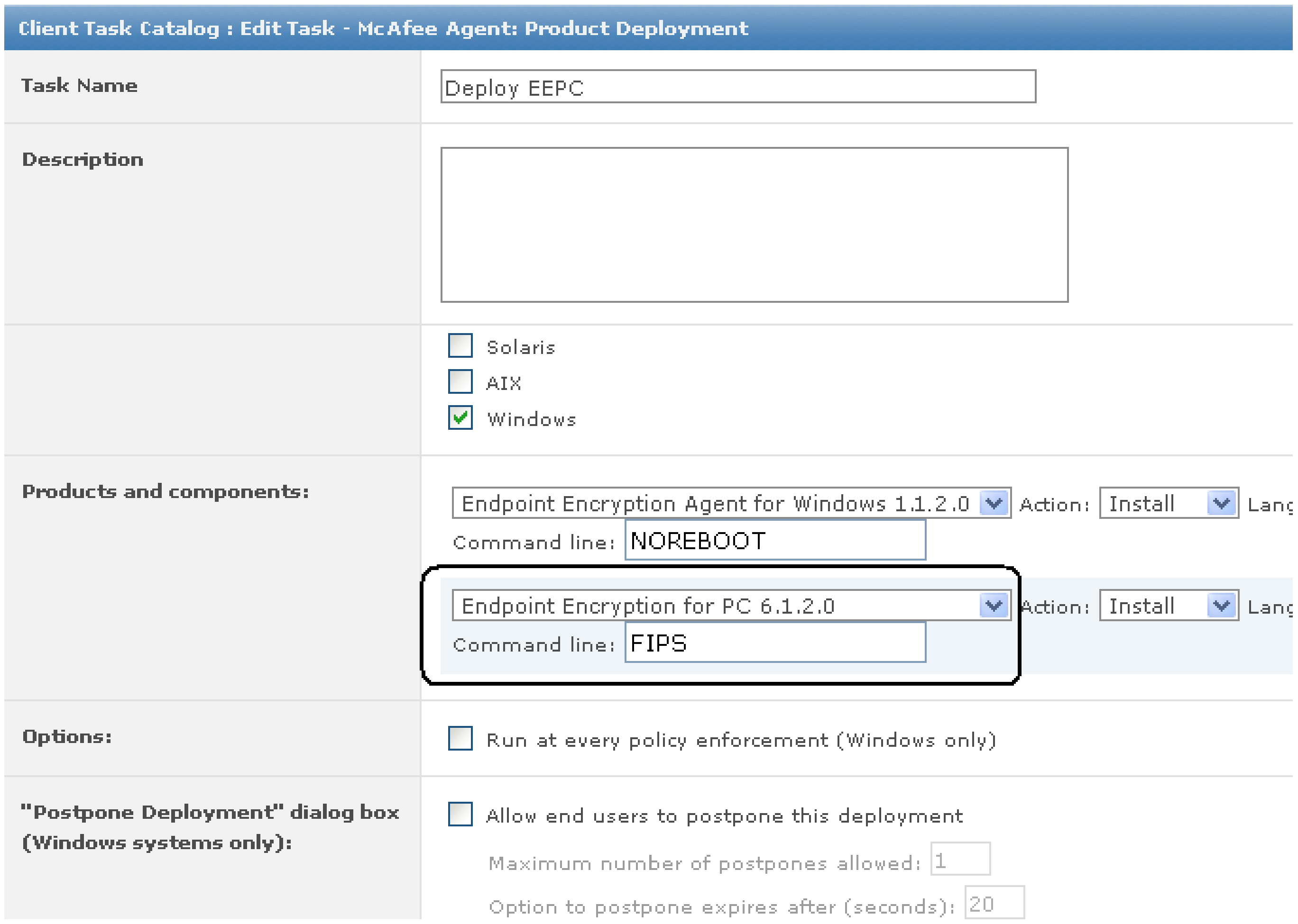
- The power-up integrity checks are performed.
- The DRBG is seeded by an internal non-approved NDRNG.
- Additional checks are made on the entropy.
- The RC5 legacy algorithm cannot be used.
- The PKCS#5 algorithm may not be used.
- RSA may only be used for key wrapping.

For the module to operate in FIPS mode, if the Asymmetric encryption service is used, this must be used with 2048 bit RSA keys.

The writable memory areas of the Module (data and stack segments) are accessible only by the Endpoint Encryption application so that the Module is in "single user" mode, i.e. only the application has access to that instance of the Module.

3.1 Deploying in FIPS mode

When the module is deployed by ePO, it is possible to set it into a FIPS mode or non-FIPS mode of operation automatically using the ePO product deployment wizard, by adding “FIPS” to the command line of the deployment task for Endpoint Encryption.



Client Task Catalog : Edit Task - McAfee Agent: Product Deployment

Task Name	Deploy EEPC
Description	
	☐ Solaris ☐ AIX ✓ Windows
Products and components:	Endpoint Encryption Agent for Windows 1.1.2.0 — Action: Install — Lang Command line: NOREBOOT Endpoint Encryption for PC 6.1.2.0 — Action: Install — Lang Command line: FIPS
Options:	☐ Run at every policy enforcement (Windows only)
"Postpone Deployment" dialog box (Windows systems only):	☐ Allow end users to postpone this deployment Maximum number of postpones allowed: 1 Option to postpone expires after (seconds): 20