Cryptographic Module for SecureOS® v9.7.1

by

Secure Computing Corporation

FIPS 140-2 Non-Proprietary Security Policy

Version 1.0

Level 1 Validation

November 9, 2007

This document may be reproduced in its entirety without modification

Table of Contents

1. Introduction

This document is the non-proprietary security policy for the Cryptographic Module for SecureOS®. This document was prepared as part of the Federal Information Processing Standard (FIPS) 140-2 Level 1 validation process.

FIPS 140-2, *Security Requirements for Cryptographic Modules*, describes the requirements for cryptographic modules. For more information about the FIPS 140-2 standard and the cryptographic module validation process see http://csrc.nist.gov/cryptval/.

2. Module Specification

The Cryptographic Module for SecureOS® (hereafter referred to as the CMSOS) is a software library supporting FIPS-approved cryptographic algorithms. For the purposes of the FIPS 140-2 level 1 validation, the Cryptographic Module for SecureOS® v9.7.1 is a single shared library file named *libcrypto.so.* This module provides a C-language application program interface (API) for use by other processes that require cryptographic functionality.

For FIPS 140-2 purposes the CMSOS is classified as a multi-chip standalone module. The *logical* cryptographic boundary of the CMSOS is the shared library file itself. The *physical* cryptographic boundary of the CMSOS is the enclosure of the computer system on which it is executing. The CMSOS performs no communications other than with the process that calls it. It makes no network or interprocess connections and creates no files.

The CMSOS was tested on a *Sidewinder® G2 Security Appliance, Model 2150D,* with *SecureOS v6.1* and a *Sidewinder®, Model 2150D with SecureOS* v7.

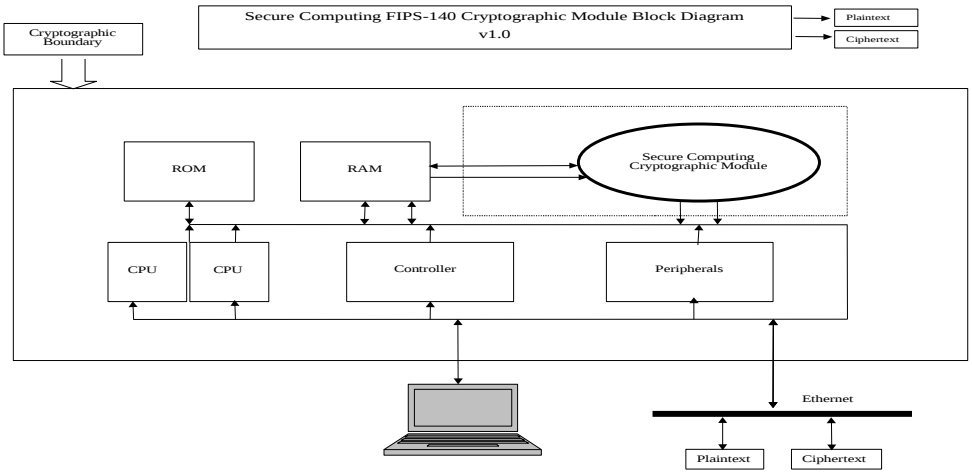

Figure 2

2.1 Roles and Services

The CMSOS meets all FIPS 140-2 level 1 requirements for Roles and Services, implementing both Crypto-User and Crypto-Officer roles. As allowed by FIPS 140-2, the CMSOS does not support user authentication for those roles. Only one role may be active at a time and the CMSOS does not allow concurrent operators.

The User and Crypto Officer roles are implicitly assumed by the entity accessing services implemented by the CMSOS. The Crypto Officer can install and initialize the CMSOS. The Crypto Officer role is implicitly entered when installing the CMSOS or performing system administration functions on the host operating system.

- User Role: Loading the CMSOS and calling any of the API functions. This role has access to all of the services provided by the CMSOS.

- Crypto-Officer Role: Installation of the CMSOS on the host computer system. This role is assumed implicitly when the system administrator installs the CMSOS library

file.

Service	*Role*	*CSP*	*Access*
Symmetric encryption/decryption	User, Crypto Officer	symmetric key	read/write/execute
Key transport	User, Crypto Officer	asymmetric private key	read/write/execute
Digital signature	User, Crypto Officer	asymmetric private key	read/write/execute
Symmetric key generation	User, Crypto Officer	symmetric key	read/write/execute
Asymmetric key generation	User, Crypto Officer	asymmetric private key	read/write/execute
Keyed Hash (HMAC)	User, Crypto Officer	HMAC SHA-1 key	read/write/execute
Message digest (SHS)	User, Crypto Officer	none	read/write/execute
Random number generation	User, Crypto Officer	seed key	read/write/execute
Show status	User, Crypto Officer	none	execute
Module initialization	User, Crypto Officer	none	execute
Self test	User, Crypto Officer	none	execute
Zeroize	User, Crypto Officer	symmetric key, asymmetric key, HMAC-SHA-1 key, seed key	

Table 2.1

2.2 Ports and Interfaces

The physical ports of the CMSOS are the same as the computer system on which it is executing. The logical interface is a C-language application program interface (API).

The Data Input interface consists of the input parameters of the API functions. The Data Output interface consists of the output parameters of the API functions. The Control Input interface consists of the actual API functions. The Status Output interface includes the return values of the API functions.

FIPS Interface	*Physical Port*	*Module Interface*
Data Input	Gigabit Ethernet ports	API input parameters
Data Output	Gigabit Ethernet ports	API output parameters
Control Input	Keyboard, Serial port, Ethernet port	API function calls
Status Output	Keyboard, Serial port, Ethernet port	API return codes
Power Input	PCI Compact Power Connector	N/A

Table 2.2

2.3 Self Tests

The CMSOS performs both power-up self tests at module initialization and continuous condition tests during operation. Input, output, and cryptographic functions cannot be performed while the CMSOS is in a self-test or error state as the module is single threaded and will not return to the calling application until the power-up self tests are complete. If the power-up self tests fail subsequent calls to the module will fail and thus no further cryptographic operations are possible.

Power-Up Self Tests

Algorithm	**Test**
AES	KAT
Triple-DES	KAT
DSA	pairwise consistency test, sign/verify
RSA	KAT
PRNG	KAT
HMAC-SHA-1	KAT

Algorithm	Test
HMAC-SHA-224	KAT
HMAC-SHA-256	KAT
HMAC-SHA-384	KAT
HMAC-SHA-512	KAT
SHA-1	KAT[1]
SHA-224	KAT[1]
SHA-256	KAT[1]
SHA-384	KAT[1]
SHA-512	KAT[1]
module integrity	HMAC-SHA-1

Table 2.3a

Conditional Self Tests

Algorithm	Test
DSA	pairwise consistency
RSA	pairwise consistency
PRNG	continuous test

Table 2.3b

A single initialization call, `FIPS_mode_set,` is required to initialize the CMSOS for operation in the FIPS 140-2 Approved mode. When the CMSOS is in FIPS mode all security functions and cryptographic algorithms are performed in Approved mode.

The FIPS mode initialization is performed when the application invokes the `FIPS_mode_set` call which returns a "1" for success and "0" for failure. Interpretation of this return code is the responsibility of the host application. Prior to this invocation the

1 Tested as part of the HMAC known answer tests.

CMSOS is uninitialized in the non-FIPS mode by default.

The `FIPS_mode_set` function verifies the integrity of the runtime executable using a HMAC-SHA-1 digest computed at build time. If this computed HMAC-SHA-1 digest matches the stored known digest then the power-up self-test, consisting of the algorithm specific Pairwise Consistency and Known Answer tests, is performed. If any component of the power-up self-test fails an internal global error flag is set to prevent subsequent invocation of any cryptographic function calls. Any such power-up self test failure is a hard error that can only be recovered by reinstalling the CMSOS[2]. If all components of the power-up self-test are successful then the CMSOS is in FIPS mode. The power-up self-tests may be performed at any time with a separate function call, `FIPS_selftest`. This function call also returns a "1" for success and "0" for failure, and interpretation of this return code is the responsibility of the host application.

A power-up self-test failure can only be cleared by a successful `FIPS_mode_set` invocation. No operator intervention is required during the running of the self-tests.

2.4 Mitigation of Other Attacks

The CMSOS does not contain additional security mechanisms beyond the requirements for FIPS 140-2 level 1 cryptographic modules.

2.5 Physical Security

The CMSOS is comprised of software only and thus does not claim any physical security.

2 The FIPS_mode_set() function could be re-invoked but such re-invocation does not provide a means from recovering from an integrity test or known answer test failure.

3. Secure Operation

The *SecureOS*® operating systems segregate user processes into separate process spaces. Each process space is an independent virtual memory area that is logically separated from all other processes by the operating system software and hardware. The CMSOS functions entirely within the process space of the process that invokes it, and thus satisfies the FIPS 140-2 requirement for a single user mode of operation.

The CMSOS is installed on the *Sidewinder*® *G2 Security Appliance* and *Sidewinder*® by the vendor during the manufacturing process, using the vendors established configuration management and quality assurance process. A complete revision history of the source code from which the CMSOS was generated is maintained in a version control database. The HMAC-SHA1 of the CMSOS library file as tested by the CMVP is verified after installation of the CMSOS file on the *Sidewinder*® *G2 Security Appliance* or *Sidewinder*®.

Upon initialization of the CMSOS, the module will run its power-up self tests. Successful completion of the power-up self tests ensures that the module is operating in the FIPS mode of operation.

As the CMSOS has no way of managing keys, any keys that are input or output from applications utilizing the module must be input or output in encrypted form using FIPS approved algorithms.

The self-tests can be called on demand by reinitializing the module using the `FIPS_mode_set` function call, or alternatively using the `FIPS_selftest` function call.

The *Sidewinder*® *G2 Security Appliance* and *Sidewinder*® are supplied to customers as a turn-key system with no customer or end-user access to the CMSOS.

4. Cryptographic Key Management

4.1 Key Generation

The CMSOS supports generation of DH, DSA, and RSA public-private key pairs. The CMSOS employs an ANSI X9.31 compliant random number generator for creation of asymmetric and symmetric keys.

4.2 Key Storage

Public and private keys are provided to the CMSOS by the calling process, and are destroyed when released by the appropriate API function calls. The CMSOS does not perform persistent storage of keys.

4.3 Key Access

An authorized application as user (the Crypto-User) has access to all key data generated during the operation of the CMSOS.

4.4 Key Protection and Zeroization

Keys residing in internally allocated data structures can only be accessed using the CMSOS defined API. The operating system protects memory and process space from unauthorized access. Zeroization of sensitive data is performed automatically by API function calls for intermediate data items, and on demand by the calling process using CMSOS provided API function calls provided for that purpose.

Only the process that creates or imports keys can use or export them. No persistent storage of key data is performed by the CMSOS. All API functions are executed by the invoking process in a non-overlapping sequence such that no two API functions will execute concurrently.

The calling process can perform key zeroization of keys by calling an API function.

4.5 Cryptographic Algorithms

Since the module is a software toolkit, it does not ship with keys or CSPs, nor does it store or manage keys or CSPs. It is the responsibility of the host application to manage key generation using the module API and to perform other key management duties including key storage.

The CMSOS supports the following FIPS approved or allowed algorithms:

Algorithm	***Validation Certificate***	***Usage***	***Keys/CSPs***
AES	#552	encrypt/decrypt	AES keys 128, 192, 256 bits
TDES	#548	encrypt/decrypt	Triple-DES keys 168 bits
Diffie-Hellman	(allowed in FIPS mode)	key agreement	Diffie-Hellman (key agreement; key establishment methodology provides 80 to 256 bits of encryption strength; non-compliant for less than 80 bits of encryption strength)

Algorithm	*Validation Certificate*	*Usage*	*Keys/CSPs*
DSA	#225	sign and verify	DSA keys 1024 bits
PRNG	#320	random number generation	PRNG seed value and seed key 64 bits
RSA (X9.31, PKCS #1.5, PSS)	#248	sign and verify	RSA keys 1024 to 16384 bits
RSA encrypt/decrypt	(allowed in FIPS mode)	key transport, key wrapping	RSA (key wrapping; key establishment methodology provides 80 to 256 bits of encryption strength; non-compliant for less than 80 bits of encryption strength)
SHA-1	#617	hashing	N/A
SHA-224	#617	hashing	N/A
SHA-256	#617	hashing	N/A
SHA-384	#617	hashing	N/A
SHA-512	#617	hashing	N/A
HMAC-SHA1	#293	message integrity	N/A
HMAC-SHA224	#293	message integrity	N/A
HMAC-SHA256	#293	message integrity	N/A
HMAC-SHA384	#293	message integrity	N/A
HMAC-SHA512	#293	message integrity	N/A

Table 4.5a

The Diffie-Hellman (key agreement, key establishment) methodology supports 80 to 256 bits of encryption strength (non-compliant for less than 80 bits of encryption strength). The RSA key wrapping methodology supports 80 to 256 bits of encryption strength (non-compliant for less than 80 bits of encryption strength).

The CMSOS supports the following non-FIPS approved algorithms:

Algorithm		***Usage***
MD5	(allowed in FIPS mode for TLS)	TLS interoperability

Table 4.5b