Symantec Corporation

Symantec Cross-Platform Cryptographic Module

SW Version: 1.0

FIPS 140-2 Non-Proprietary Security Policy

FIPS Security Level: 1
Document Version: 1.0

Prepared for:

Symantec Corporation
350 Ellis Street
Mountain View, CA 94043
United States of America

Phone: +1 (650) 527-8000
http://www.symantec.com

Prepared by:

Corsec Security, Inc.
13135 Lee Jackson Memorial Hwy, Suit 220
Fairfax, VA 22033
United States of America

Phone: +1 (703) 267-6050
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 Symantec Cross-Platform Cryptographic Module from Symantec Corporation. This Security Policy describes how the Symantec Cross-Platform Cryptographic Module 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 Symantec Cross-Platform Cryptographic Module is referred to in this document as SymCPM, the cryptographic 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 Symantec website (http://www.symantec.com) contains information on the full line of products from Symantec.
- 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
- Other supporting documentation as additional references

This Security Policy and the other validation submission documentation were produced by Corsec Security, Inc. under contract to Symantec. With the exception of this Non-Proprietary Security Policy, the FIPS 140-2 Submission Package is proprietary to Symantec and is releasable only under appropriate non-disclosure agreements. For access to these documents, please contact Symantec.

2 SymCPM

This section describes the Symantec Cross-Platform Cryptographic Module from Symantec Corporation.

2.1 Overview

As one of the world's largest software companies, Symantec offers a comprehensive portfolio of security, storage, and systems management solutions. Symantec has a customer base that includes 99% of Fortune 1000 companies. It also has a broad range of product offerings that range from consumer virus protection to enterprise class Security Operation Centers (SOCs) that are staffed 24/7. Through their software and services, they help consumers and organizations protect information and infrastructure against more risks at more points, more completely and efficiently than any other company. With an arsenal that includes engineering centers, global patents, and cutting-edge research, customers can have what only a global leader like Symantec can provide - confidence in a connected world.

2.1.1 Symantec Security Information Manager

The first Symantec solution to take advantage of SymCPM is the Symantec Security Information Manager. SSIM is a high availability enterprise class software solution, whose primary purpose is to preempt or detect security incidents while providing the framework to both respond and demonstrate compliance. SSIM accomplishes this through its integrated log management, distributed architecture, and automated updates from Symantec's Global Information Network (GIN), which offers real-time intelligence on the latest vulnerabilities and threats from around the world.

SSIM collects security information, called events, from a broad range of applications, services, and security products. It then converts that information into actionable intelligence by using its built-in asset management function for prioritization, and then applying its sophisticated rule-based correlation engine on a normalized event stream. Event data is easily managed and quickly retrieved using SSIM's specialized form of event detail storage, which uses proprietary indexing and compression. Large amounts of diverse event data can be centralized in online or archived event stores using direct-attached storage (DAS), network-attached storage (NAS) or storage area network (SAN). SSIM also embeds a high performance relational database to store both summarized events and data pertaining to incidents, tickets, assets, rules, vulnerabilities, workflow, and reports. This allows for trend reporting and custom SQL[1] queries, with drill down capability into the appropriate event archives.

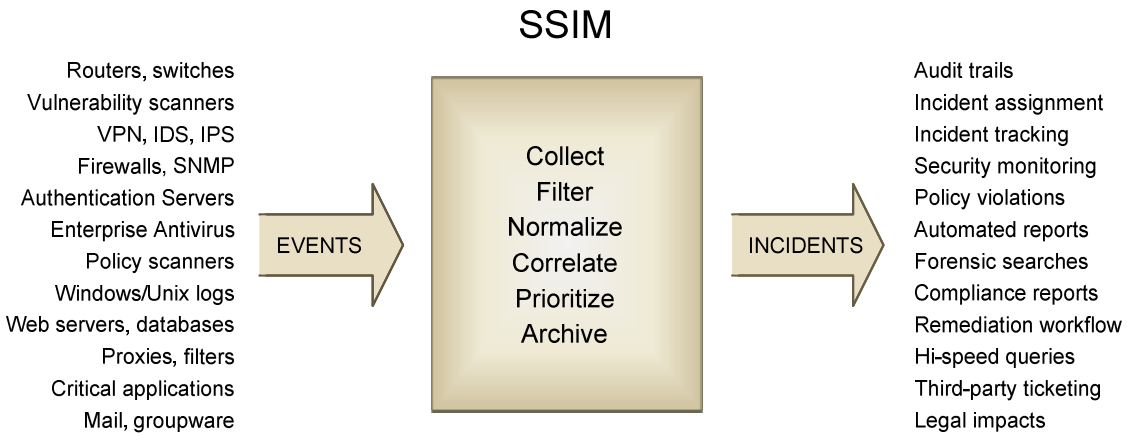

Figure 1 – Symantec Security Information Manager Workflow

[1] SQL – Structured Query Language

The SSIM solution consists of the following major components:

- **Information Manager** – comprises the core functionality of a SSIM deployment. It installs on a standard server platform that supports RHEL (Red Had Enterprise Linux). It aggregates and processes event data for correlation, incident management, and archival. It also consists of a comprehensive report engine and an embedded LDAP (Lightweight Directory Access Protocol) directory for centralized access control and multi-domain management. The various roles of the Information Manager (Collection, Correlation, Archival, and Service Provider) can be distributed in local clusters and then federated for fail safety, higher scalability, and global deployments.
- **Event Collector** – gathers and filters events from event sources. Collectors are installed directly on a security point product or in strategic locations with access to security events. They use application specific *sensors* that retrieve events from a file, database, or syslog. SSIM provides multi-vendor support with over 200 predefined collectors for popular products. It also has universal collectors that can be customized for unique event sources.
- **Event Agent** – handles the communication path between event collectors and the Information Manager. It is a Java-based application that is installed alongside each collector. The agent forwards both raw event data and events that have been filtered and aggregated.
- **Information Manager Console** – provides a bidirectional administrative GUI[2] to the Information Manager. It is a Java-based application used by administrators, analysts, and service desk systems to perform security monitoring functions, such as incident management, reports, and rule definition. It presents both high-level and detailed views of critical security information.
- **Web Configuration Interface** and **OpenSSH –** provide administrative access to the Information Manager server via a web browser or a remote secure shell (SSH) connection.

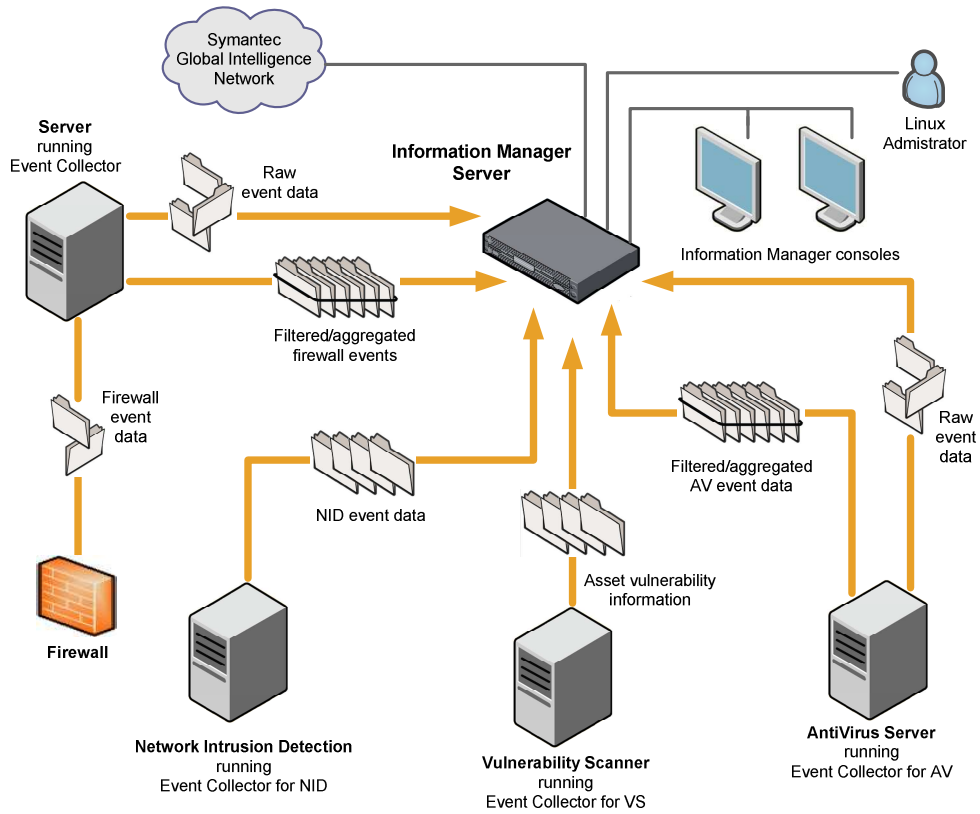

Figure 2 – SSIM Architecture Overview

[2] GUI – Graphical User Interface

2.1.2 Symantec Cross-Platform Cryptographic Module Security

The Symantec Cross-Platform Cryptographic Module, SymCPM, is a general-purpose cryptographic library that resides on various Symantec application components, including SSIM's Event Agent and the Information Manager server and console. It provides the cryptographic services required by TLS[3] for secure communication between SSIM components. The module includes implementations of the following FIPS-Approved algorithms:

- Advanced Encryption Standard (AES)
- Triple Data Encryption Algorithm (TDEA or Triple-DES[4])
- Secure Hash Algorithm (SHA)
- (Keyed-) Hash Message Authentication Code (HMAC)
- Digital Signature Algorithm (DSA)
- RSA[5] signature generation and verification
- SP 800-90 Deterministic Random Bit Generator (DRBG)

The Symantec Cross-Platform Cryptographic Module operates in either a FIPS Approved or non-FIPS Approved mode of operation. It 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[6]	1
9	Self-tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A

2.2 Module Specification

The Symantec Cross-Platform Cryptographic Module is a software module with a multi-chip standalone embodiment. The overall security level of the module is 1. SymCPM is implemented in the C programming language and consists of shared libraries that are linked with SSIM application components. It is designed to execute on a host system with a General Purpose Computer (GPC) hardware platform. The following sections define the physical and logical boundary of the SymCPM module.

[3] TLS – Transport Layer Security
[4] DES – Data Encryption Standard
[5] RSA – Rivest, Shamir, Adleman
[6] EMI/EMC – Electromagnetic Interference / Electromagnetic Compatibility

2.2.1 Physical Cryptographic Boundary

As a software cryptographic module, there are no physical protection mechanisms implemented. Therefore, the module must rely on the physical characteristics of the host system. The physical boundary of the cryptographic module is defined by the hard enclosure around the host system on which it runs. The module supports the physical interfaces of a GPC, including the integrated circuits of the system board, the CPU, network adapters, RAM, hard disk, device case, power supply, and fans. Other devices may be attached to the GPC, such as a display monitor, keyboard, mouse, printer, or storage media. See Figure 3 below for a standard GPC block diagram.

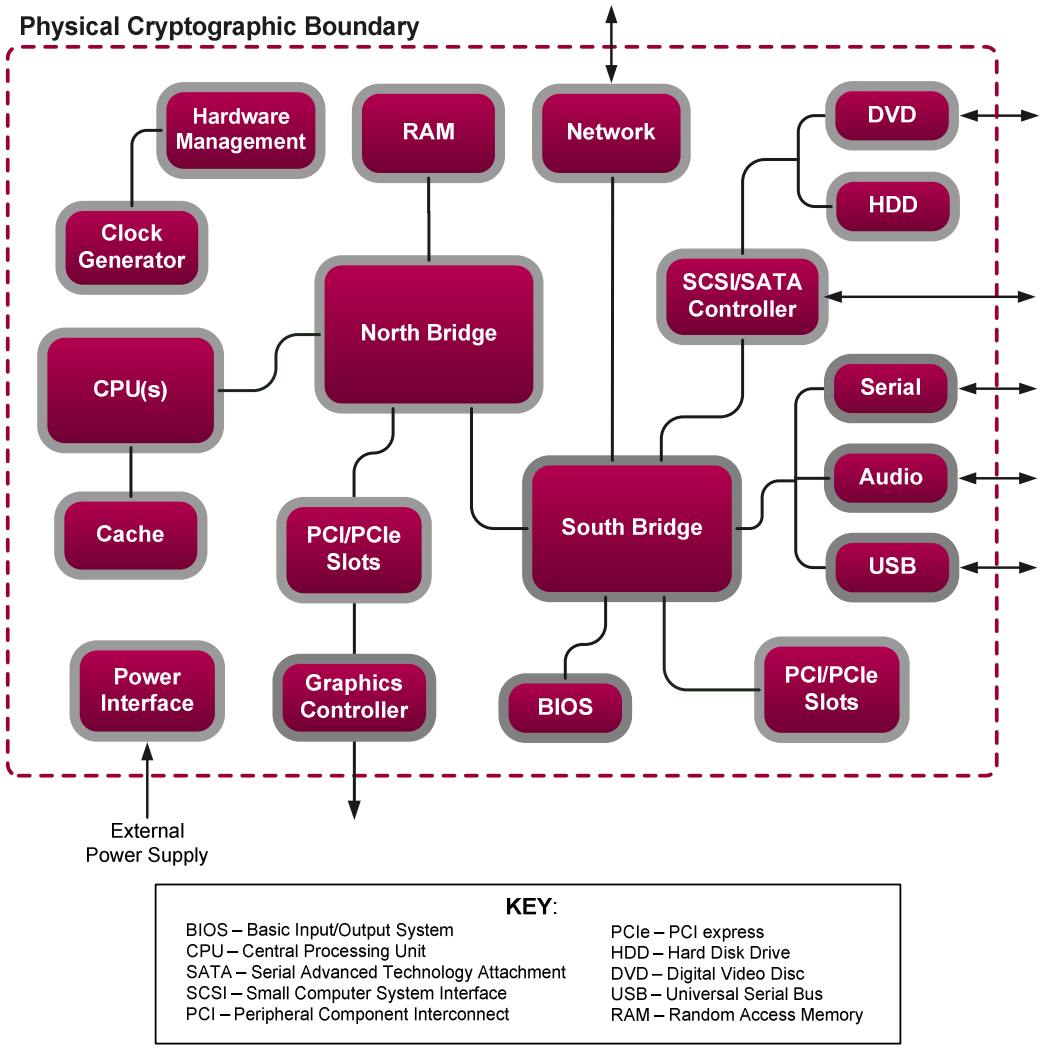

Figure 3 – Standard GPC Block Diagram

2.2.2 Logical Cryptographic Boundary

Figure 4 shows a logical block diagram of the module executing in memory and its interactions with surrounding software components, as well as the module's logical cryptographic boundary. The module's services are designed to be called by other Symantec software components. SymCPM v1.0 is composed of 3 files for the Windows and RHEL operating systems, and 5 files for the Solaris operating system.

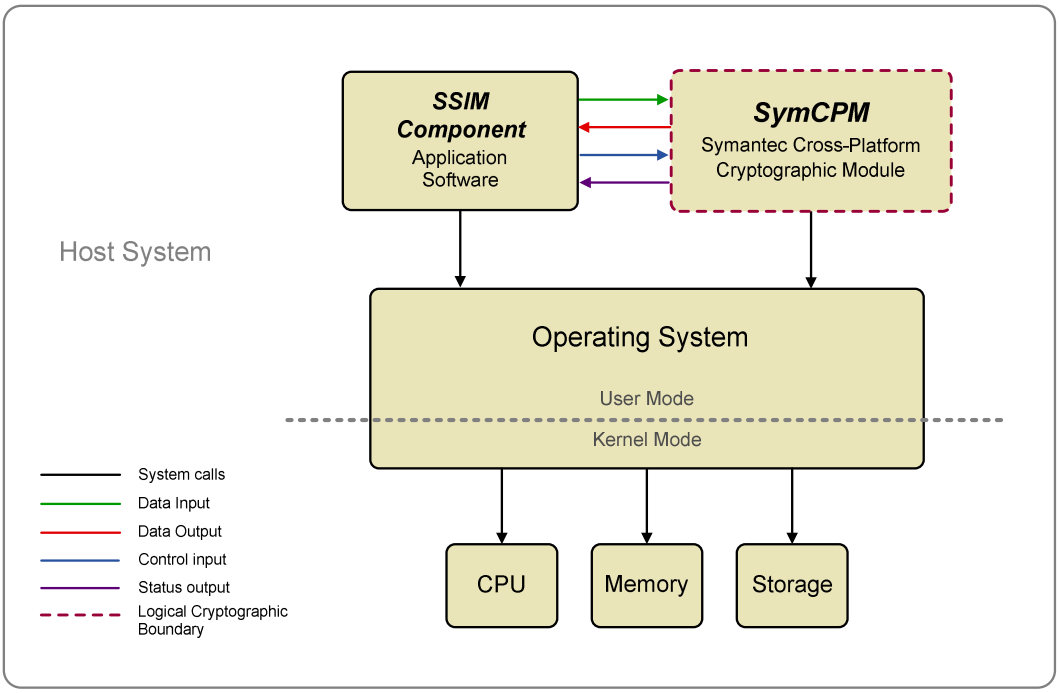

Figure 4 – Logical Block Diagram and Cryptographic Boundary

2.3 Module Interfaces

The module's logical interfaces exist at a low level in the software as an Application Programming Interface (API). Both the API and physical interfaces can be categorized into following interfaces defined by FIPS 140-2: Data Input, Data Output, Control Input, and Status Output. A mapping of the FIPS 140-2 logical interfaces, the physical interfaces, and the module interfaces can be found in Table 2 below.

Table 2 – FIPS 140-2 Interface Mappings

FIPS Interface	Physical Interface	Module Interface (API)
Data Input	USB ports (keyboard, mouse, data), network ports, serial ports, SCSI/SATA ports, DVD drive	Arguments for library functions that specify plaintext data, ciphertext, digital signatures, cryptographic keys (plaintext or encrypted), initialization vectors, and passwords that are to be input to and processed by the cryptographic module. Data examples: • pointers (to contexts, structures, variables, data locations, session handles, object handles, signatures, plaintext, ciphertext, message digests, object templates, etc.) • sizes of various data (encrypted data, recovered plaintext, RSA modulus ,signature, MAC, shared secret, digests, blocks) • type values (user, slot, mechanism) • slot IDs • flag values

FIPS Interface	Physical Interface	Module Interface (API)
Data Output	Monitor, USB ports, network ports, serial ports, SCSI/SATA ports, audio ports, DVD drive	Arguments for library functions that receive plaintext data, ciphertext data, digital signatures, cryptographic keys (plaintext or encrypted), and initialization vectors from the cryptographic module. Data examples: • pointers (to contexts, structures, variables, data locations, signatures, plaintext, ciphertext, message digests, etc.) • keys • random strings • token, object, slot, or mechanism information
Control Input	USB ports (keyboard, mouse), network ports, serial ports, power switch	Arguments for library functions that initiate and control the operation of the module, such as arguments that specify commands and control data (e.g., algorithms, algorithm modes, digest type, or module settings).
Status Output	Monitor, network ports, serial ports	Function return codes, error codes, or output arguments that receive status information used to indicate the status of the cryptographic module. Status information may be: • integer values that indicates success or failure • a NULL value that indicates success or failure • flag values
Power Input	Power Interface	N/A

2.4 Roles and Services

The Symantec Cross-Platform Cryptographic Module supports the following two roles for operators, as required by FIPS 140-2: Crypto-Officer (CO) role and User role. Both roles are implicitly assumed when services that do not require a login are executed. Although the module employs authentication mechanisms, it should be noted that the module only claims Security Level 1 for this section. SymCPM uses role-based authentication to control access to services that require access to sensitive CSPs. To perform these sensitive services, an operator must log into the module by authenticating with a password. The password is initialized by the CO as part of module initialization. When performing any service that requires login, an operator must explicitly request to assume the appropriate role by authenticating to the module using a password.

The logical interface of SymCPM consists of the PKCS #11API, also called *Cryptoki* or the "cryptographic token interface". The module implements three PKCS #11 tokens: two tokens for the non-FIPS Approved mode of operation, and one token for the FIPS Approved mode of operation. The FIPS token is designed specifically for FIPS 140-2, and allows applications using the module to operate using only FIPS Approved functions. The services listed in the following tables correspond to the functions available in PKCS #11.

Note 1: Table 3 through Table 8 use the following definitions for "CSP[7] and Type of Access".

R – Read: *The plaintext CSP is read by the service.*
W – Write: *The CSP is established, generated, modified, or zeroized by the service.*
X – Execute: *The CSP is used within an Approved (or allowed) security function or authentication mechanism.*
Z – Zerioze: *The CSP is zeroized with an Approved security function*

[7] CSP – Critical Security Parameter

Note 2: The "Input" and "Output" columns are with respect to the module's logical boundary.

2.4.1 Crypto Officer Role

The operator in the Crypto Officer role installs, uninstalls, and administers the module via the host platform's OS interfaces. An operator assumes the CO role by invoking one of the following services:

Table 3 – Crypto Officer Services

Service	Description	Input	Output	CSP Access
FC_GetFunctionList	Return a list of PKCS#11 function pointers for FIPS mode services	API call parameters	Status	None
C_InitToken	Initialize or reinitialize a token by clearing the key database and removing the password	API call parameters	Status	All – W
C_InitPIN	Set a user's initial password. Login required.	Password, API call parameters	Status	Password – WZ
C_Initialize	Load and initialize the module, perform power-up self-tests – enter FIPS mode	API call parameters	Status	None
C_Finalize	Shutdown and unload the module library – leave FIPS mode	API call parameters	Status	All – WZ
C_GetInfo	Obtain general information about module library	API call parameters	Data, status	None

2.4.2 User Role

The operator in the User role is a consumer of the module's security services. All cryptographic and general-purpose services are available to the User except those that perform an installation function. The role is assumed by invoking a service listed in the tables below.

Table 4 – User Services for Slot and Token Management

Service	Description	Input	Output	CSP Access
C_GetSlotList	Obtain a list of available slots	API call parameters	Data, status	None
C_GetSlotInfo	Obtain information about a particular slot	API call parameters	Data, status	None
C_GetTokenInfo	Obtain information (show status) about a token	API call parameters	Data, status	None
C_GetMechanismList	Obtain a list of crypto algorithms (mechanisms) supported by a token	API call parameters	Data, status	None
C_GetMechanismInfo	Obtain information about a mechanism	API call parameters	Data, status	None
C_SetPIN	Change a user's password. Login required.	Old password, API call parameters	New password, status	Password – WX

Table 5 – User Services for Session Management

Service	Description	Input	Output	CSP Access
C_OpenSession	Open a session between an application and a token	API call parameters	Status	None
C_CloseSession	Close a session	API call parameters	Status	All keys – WZ
C_CloseAllSessions	Close all sessions with a token	API call parameters	Status	All keys – WZ
C_GetSessionInfo	Obtain information (show status) about a session	API call parameters	Status	None
C_GetOperationState	Save the state of a session's digest operation	API call parameters	Data, status	None
C_SetOperationState	Restore the state of a session's digest operation	API call parameters	Status	None
C_Login	Log into a token	Password, API call parameters	Status	Password – X
C_Logout	Log out from a token. Login required.	API call parameters	Status	None

Table 6 – User Services for Object Management

Service	Description	Input	Output	CSP Access
C_CreateObject	Create an object. Login required.	API call parameters	Data, key, status	All keys – W
C_CopyObject	Create a copy of an object. Login required.	Key, API call parameters	Data, key, status	All keys – RW
C_DestroyObject	Destroy an object. Login required.	Key, API call parameters	Status	All keys – WZ
C_GetObjectSize	Obtain the size of an object in bytes. Login required.	Key, API call parameters	Data, status	All keys – R
C_GetAttributeValue	Obtain an attribute value of an object. Login required.	Key, API call parameters	Data, status	All keys – R
C_SetAttributeValue	Modify an object's attribute value. Login required.	Key, API call parameters	Data, key, status	All keys – W
C_FindObjectsInit	Initialize an object search operation. Login required.	API call parameters	Status	None
C_FindObjects	Continue an object search operation. Login required.	Key, API call parameters	Data, key, status	All keys – R
C_FindObjectsFinal	Finish an object search operation. Login required.	API call parameters	Status	None

Table 7 – User Services for Cryptographic Services

Service	Description	Input	Output	CSP Access
C_EncryptInit	Initialize an encryption operation. Login required.	Key, API call parameters	Status	AES key – R TDES key – R
C_Encrypt	Encrypt single-part data. Login required.	Plaintext, API call parameters	Status, ciphertext	AES key – X TDES key – X
C_EncryptUpdate	Continue a multi-part encryption. Login required.	Plaintext, API call parameters	Status, ciphertext	AES key – X TDES key – X
C_EncryptFinal	Finish a multi-part encryption. Login required.	Plaintext, API call parameters	Status, ciphertext	AES key – X TDES key – X
C_DecryptInit	Initialize a decryption operation. Login required.	Key, API call parameters	Status	AES key – R TDES key – R
C_Decrypt	Decrypt single-part data. Login required.	Ciphertext, API call parameters	Status, plaintext	AES key – X TDES key – X
C_DecryptUpdate	Continue a multi-part decryption. Login required.	Ciphertext, API call parameters	Status, plaintext	AES key – X TDES key – X
C_DecryptFinal	Finish a multi-part decryption. Login required.	Ciphertext, API call parameters	Status, plaintext	AES key – X TDES key – X
C_DigestInit	Initialize a message-digest operation using SHS[8]	API call parameters	Status	None
C_Digest	Digest single-part data	Message, API call parameters	Status	None
C_DigestUpdate	Continue a multi-part digest	Message, API call parameters	Status	None
C_DigestKey	Continue a multi-part digest by digesting the value of a secret key. Login required.	Key, API call parameters	Status	AES key – R TDES key – R HMAC key – R
C_DigestFinal	Finish a multi-part digest	API call parameters	Status, digest	None
C_SignInit	Initialize a signature or MAC operation. Login required.	Key, API call parameters	Status	RSA private – R DSA private – R HMAC key – R
C_Sign	Sign or MAC single-part data. Login required.	Message, API call parameters	Status	HMAC key – X
C_SignUpdate	Continue a multi-part MAC or signature. Login required.	Message, API call parameters	Status	HMAC key – X
C_SignFinal	Finish a multi-part MAC or signature. Login required.	API call parameters	Status, MAC, signature	RSA private – X DSA private – X HMAC key – X
C_SignRecoverInit	Initialize a signature operation with digest recovery. Login required.	Key, API call parameters	Status	RSA private – R DSA private – R

[8] SHS – Secure Hash Standard

Service	Description	Input	Output	CSP Access
C_SignRecover	Sign single-part data with digest recovery. Login required.	Message, API call parameters	Status, signature	RSA private – X DSA private – X
C_VerifyInit	Initialize a verification operation. Login required.	Key, API call parameters	Status	RSA public – R DSA public – R HMAC key – R
C_Verify	Verify a signature on single-part data. Login required.	MAC, API call parameters, signature	Status	RSA public – X DSA public – X HMAC key – X
C_VerifyUpdate	Continue a multi-part verification. Login required.	MAC, API call parameters, signature	Status	RSA public – X DSA public – X HMAC key – X
C_VerifyFinal	Finish a multi-part verification. Login required.	MAC, API call parameters, signature	Status	RSA public – X DSA public – X HMAC key – X
C_VerifyRecoverInit	Initialize a verification operation with digest recovery. Login required.	Key, API call parameters	Status	RSA public – R DSA public – R
C_VerifyRecover	Verify a signature on single-part data with digest recovery. Login required.	API call parameters	Status, digest	RSA public – X DSA public – X
C_DigestEncryptUpdate	Continue a multi-part digest and encryption. Login required.	Digest, API call parameters	Status, ciphertext	AES key – X TDES key – X
C_DecryptDigestUpdate	Continue a multi-part decryption and digest. Login required.	Ciphertext, API call parameters	Status, digest	AES key – X TDES key – X
C_SignEncryptUpdate	Continue a multi-part signature and encryption. Login required.	Message, API call parameters	Status, ciphertext, signature	AES key – X TDES key – X RSA public – X DSA public – X HMAC key – X
C_DecryptVerifyUpdate	Continue a multi-part decryption and verification. Login required.	Ciphertext, API call parameters, signature	Status, message	AES key – X TDES key – X RSA public – X DSA public – X HMAC key – X
C_GenerateKey	Generate a secret key. Login required.	API call parameters	Status, key	AES key – W TDES key – W HMAC key – W
C_GenerateKeyPair	Generate an asymmetric key pair. Login required.	API call parameters	Status, key pair	RSA key pair – W DSA key pair – W
C_WrapKey	Wrap (encrypt) a key. Login required.	Key, API call parameters	Status, ciphertext	AES key – R TDES key – R RSA public – X

Service	Description	Input	Output	CSP Access
C_UnwrapKey	Unwrap (decrypt) a key. Login required.	Ciphertext, API call parameters	Status, key	AES key – R TDES key – R RSA private – X
C_DeriveKey	Derive a key from a base key. Login required.	API call parameters	Status, key	DH[9] – W TLS master secret
C_SeedRandom	Mix in additional seed material to the random number generator	API call parameters	Status	RNG[10] seed
C_GenerateRandom	Generate random data	API call parameters	Status, random data	None

Table 8 – Legacy User Services

Service	Description	Input	Output	CSP Access
C_GetFunctionStatus	Legacy function, which simply returns the value 0x00000051	API call parameters	Status	None
C_CancelFunction	Legacy function, which simply returns the value 0x00000051	API call parameters	Status	None

2.5 Physical Security

The Symantec Cross-Platform Cryptographic Module is a software module and does not include physical security mechanisms. Thus, the FIPS 140-2 requirements for physical security are not applicable.

2.6 Operational Environment

The module was tested and found to be compliant with FIPS 140-2 requirements on the following platforms:

- GPC running Windows 2003 Server 32-bit
- GPC running RHEL 5 32-bit
- GPC running Solaris 10

Symantec affirms that the module also executes in its FIPS-Approved manner (as described in this Security Policy) on other operating systems that are binary-compatible to those on which the module was tested. All cryptographic keys and CSPs are under the control of the operating system, which protects the CSPs against unauthorized disclosure, modification, and substitution. The module only allows access to CSPs through its well-defined API.

2.7 Cryptographic Key Management

The module implements the following FIPS-Approved algorithms:

[9] DH – Diffie-Hellman
[10] RNG – Random Number Generator

Table 9 – FIPS-Approved Algorithm Implementations

Algorithm	Certificate Number
AES in ECB[11], CBC modes with 128, 192, and 256 bit keys	1614
Triple-DES in ECB, CBC modes with 112 and 168 bit keys	1055
RSA (PKCS[12] #1.5) sign/verify with 1024, 1536, 2048, 3072, 4096 bit keys	792
DSA sign/verify and key generation with 1024 bit keys	502
SHA-1, SHA-256, SHA-384, SHA-512	1423
HMAC-SHA-1, HMAC-SHA-256, HMAC- SHA-384, HMAC-SHA-512	946
SP 800-90 Hash_DRBG	83

The module utilizes the following non-Approved algorithms, which are allowed for use in a FIPS-Approved mode of operation:

- RSA key wrapping (1024- to 8192-bit keys)
  - Key establishment methodology provides between 80 and 192 bits of encryption strength
- Diffie-Hellman key agreement (1024- to 2236-bit keys)
  - Key establishment methodology provides between 80 and 112 bits of security
- Message Digest 5 (MD5)
  - Message authentication for use within the TLS Key Derivation Function (KDF)

Additionally, the module implements the following non-Approved algorithms, which are only available in a non-Approved mode of operation:

- DES
- Camellia
- SEED
- RC2 (Rivest Cipher 2)
- RC4 (Rivest Cipher 4)
- MD2

The CSPs supported by the module are shown in Table 10 below.

Note: The "Input" and "Output" columns in Table 10 are in reference to the module's logical boundary. Keys that enter and exit the module via an API call parameter can be in plaintext or ciphertext.

Table 10 – List of Cryptographic Keys, Key Components, and CSPs

CSP/Key	CSP/Key Type	Input	Output	Storage	Zeroization	Use
AES key	AES128-, 192-, 256-bit key	API call parameter or internally generated	API call parameter	Plaintext in volatile memory	By API call, power cycle	Encryption, Decryption
TDES key	TDES 112-, 168-bit key	API call parameter or internally generated	API call parameter	Plaintext in volatile memory	By API call, power cycle	Encryption, decryption

[11] ECB – Electronic Codebook
[12] PKCS – Public-Key Cryptography Standards

CSP/Key	CSP/Key Type	Input	Output	Storage	Zeroization	Use
HMAC key	HMAC key	API call parameter or internally generated	API call parameter	Plaintext in volatile memory	By API call, power cycle	Message Authentication with SHA-1 and SHA-2s
RSA private key	RSA 1024-, 1536-, 2048-, 3072-, 4096-bit key	API call parameter or internally generated	API call parameter	Plaintext in volatile memory	By API call, power cycle	Signature generation
RSA public key	RSA 1024-, 1536-, 2048-, 3072-, 4096-bit key	API call parameter or internally generated	API call parameter	Plaintext in volatile memory	By API call, power cycle	Signature verification
DSA private key	DSA 1024-bit key	API call parameter or internally generated	API call parameter	Plaintext in volatile memory	By API call, power cycle	Signature generation
DSA public key	DSA 1024-bit key	API call parameter or internally generated	API call parameter	Plaintext in volatile memory	By API call, power cycle	Signature verification
DH public key	DH 1024-bit key	API call parameter or internally generated	API call parameter	Plaintext in volatile memory	By API call, power cycle	Encryption, Decryption
DH private key	DH 1024-bit key	API call parameter or internally generated	API call parameter	Plaintext in volatile memory	By API call, power cycle	Encryption, Decryption
DRBG seed	880-bit random value	Internally generated	Never	Plaintext in volatile memory	By API call, power cycle	Seed input to SP 800-90 Hash_DRBG
Hash DRBG V value	Internal hash DRBG state value	Internally generated	Never	Plaintext in volatile memory	By API call, power cycle	Used for SP 800-90 Hash_DRBG
Hash DRBG C value	Internal hash DRBG state value	Internally generated	Never	Plaintext in volatile memory	By API call, power cycle	Used for SP 800-90 Hash_DRBG
User password	password	API call parameter	API call parameter	Plaintext in volatile memory	By API call, power cycle	User password is used to log into the token
Software Integrity key	DSA1024-bit public	Never	Never	Plaintext in .chk file or volatile memory	Memory zeroization only by API call	Used to verify module integrity

2.7.1 Key Generation

Symmetric cryptographic keys are generated using the SP 800-90 Hash_DRBG. Asymmetric key pairs are generated using the methods specified in the PKCS #1 RSA Encryption Standard and in FIPS 186-2 (Digital Signature Standard) with Change Notice 1.

2.7.2 Key Entry and Output

The cryptographic module itself does not support key entry or key output from its physical boundary. However, keys are passed to the module as parameters from the applications resident on the host platform via the exposed APIs. Similarly, keys and CSPs exit the module in plaintext or ciphertext via the well-defined exported APIs.

2.7.3 Key/CSP Storage and Zeroization

The module stores keys and CSPs in volatile memory, but does not persistently store any keys or CSPs on disk. Plaintext secret or private keys are zeroized when passed to the *C_DestroyObject, C_CloseSession,* or *C_CloseAllSessions* functions. All plaintext secret and private keys are automatically zeroized by the functions *C_Finalize* or *C_InitToken*. These services are the only services available to zeroize plaintext secret and private keys. All zeroization is performed by storing the value 0 into every byte of the memory region previously occupied by the plaintext secret key, private key, or password.

2.8 EMI/EMC

SymCPM is a software module. Therefore, the only electromagnetic interference produced is that of the host platform on which SymCPM resides and executes. FIPS 140-2 requires that the host systems on which FIPS 140-2 testing is performed meet the Federal Communications Commission (FCC) EMI and EMC requirements for business use as defined in Subpart B, Class A of FCC 47 Code of Federal Regulations Part 15. However, all systems sold in the United States must meet these applicable FCC requirements.

2.9 Self-Tests

The cryptographic module performs power-up and conditional self-tests listed in the sections below. If a self-test fails, the module will enter an error state. While in an error state, the module inhibits all data output and does not provide any cryptographic functionality until the error state is cleared.

2.9.1 Power-Up Self-Tests

The Symantec Cross-Platform Cryptographic Module performs the following self-tests at power-up:

- Software integrity test with DSA – A signature of the library is created using a DSA private key. The signature and public key are stored until the integrity test is executed. The public key is then used to create a new signature and verify it against the stored signature. The module will not initialize if any files have been modified.
- Known Answer Tests (KATs)
    - SHS
    - HMAC-SHA1
    - HMAC-SHA-256
    - HMAC-SHA-384
    - HMAC-SHA-512
    - Triple-DES-CBC encrypt/decrypt
    - Triple-DES-ECB encrypt/decrypt
    - AES-CBC encrypt/decrypt
    - AES-ECB encrypt/decrypt
    - Hash_DRBG
    - RSA for signature generation and verification
    - RSA for encrypt/decrypt
    - DSA for signature generation and verification

Upon successful completion of the power-up self-tests the module will return the `CKR_OK` status to the operator. Any error in the power-up self-tests will result in the `CKR_DEVICE_ERROR` status to be passed to the operator and the module will not be initialized.

2.9.2 Conditional Self-Tests

The Symantec Cross-Platform Cryptographic Module performs the following conditional self-tests:

- Continuous RNG test
- RSA pairwise consistency for sign/verify and encrypt/decrypt
- DSA pairwise consistency

Upon successful completion of the conditional self-tests the module will return the `CKR_OK` status to the operator. Any error in the conditional self-tests will result in the `CKR_DEVICE_ERROR` or `CKR_GENERAL_ERROR` status to be passed to the operator. Module operation will be inhibited.

2.10 Mitigation of Other Attacks

This section is not applicable. The module does not claim to mitigate any attacks beyond the FIPS 140-2 Level 1 requirements for this validation.

3 Secure Operation

The Symantec Cross-Platform Cryptographic Module 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 Initial Setup

The shared libraries and the associated *.chk* files should be installed in a directory on the shared library search path. SymCPM requires the Netscape Portable Runtime (NSPR) libraries which should also be installed in a directory on the shared library search path. The default mode of operation for the module is non-FIPS. To place the module in FIPS Approved mode, the SSIM software application must call *FC_GetFunctionList* to obtain the list of FIPS function pointers. The *C_Initialize* call initializes the module and performs power-up self-tests, including an integrity test using a DSA signature. If the module passes all self-tests, the module is in a FIPS-Approved mode of operation, and subsequent calls to the module's API using the obtained function pointer list will select the FIPS Approved mode of operation

3.2 Crypto–Officer Guidance

FIPS 140-2 mandates that a software cryptographic module at Security Level 1 be restricted to a single operator mode of operation. Prior to installing the module, the Crypto-Officer must ensure the host system OS is configured for single-user mode.

To configure the Windows OS for single-user mode, the Crypto-Officer must ensure that all remote guest accounts are disabled in order to ensure that only one operator can log into the Windows OS at a time. The services that need to be turned off for Windows are:

- Fast-user switching (irrelevant if server is a domain member)
- Terminal services
- Remote registry service
- Secondary logon service
- Telnet service
- Remote desktop and remote assistance service

The following explains how to configure a RHEL system for single-user mode.

- Remove all login accounts except "root" (the superuser).
- Disable NIS and other name services for users and groups.
- Turn off all remote login, remote command execution, and file transfer daemons.

In order to configure Solaris for single-user mode, the Crypto-Officer must type boot –s at the OK prompt or edit /boot/grub/menu.lst to set the single-user mode boot option.

Self-tests can be performed by the function calls *C_Finalize* followed by *C_Initialize*. This sequence executes the same power-up self-tests as when the module library for the FIPS Approved mode.

3.3 User Guidance

The SymCPM module is designed for use by Symantec software applications. SymCPM does not input, output, or persistently store CSPs with respect to the physical boundary. The User (Symantec software component, in this case) is responsible for providing persistent storage of the cryptographic keys and CSPs, and to ensure that keys are transmitted outside the physical cryptographic boundary in a secure manner only using FIPS-Approved algorithms like RSA.

4 Acronyms

Table 11 – Acronyms

Acronym	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
BIOS	Basic Input/Output System
CBC	Cipher Block Chaining
CMVP	Cryptographic Module Validation Program
CO	Crypto Officer
CPU	Central Processing Unit
CSEC	Communications Security Establishment Canada
CSP	Critical Security Parameter
DAS	Direct-Attached Storage
DES	Data Encryption Standard
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Algorithm
DVD	Digital Video Disc
ECB	Electronic Code Book
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FCC	Federal Communications Commission
FIPS	Federal Information Processing Standard
GIN	Global Information Network
GPC	General Purpose Computer
GUI	Graphical User Interface
HDD	Hard Disk Drive
HMAC	(Keyed-) Hash Message Authentication Code
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
KAT	Known Answer Test
KDF	Key Derivation Function
LDAP	Lightweight Directory Access Protocol
MAC	Message Authentication Code

Acronym	Definition
MD	Message Digest
NAS	Network-Attached Storage
NIST	National Institute of Standards and Technology
NSPR	Netscape Portable Runtime
NSS	Network Security Services
NVLAP	National Voluntary Laboratory Accreditation Program
OS	Operating System
PCI	Peripheral Component Interconnect
PCIe	Peripheral Component Interconnect Express
PKCS	Public-Key Cryptography Standards
PRNG	Pseudo Random Number Generator
RAM	Random Access Memory
RC	Rivest Cipher
RHEL	Red Hat Enterprise Linux
RNG	Random Number Generator
RSA	Rivest, Shamir, and Adleman
SAN	Storage Area Network
SATA	Serial Advanced Technology Attachment
SCSI	Small Computer System Interface
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SOC	Security Operations Center
SP	Special Publication
SQL	Structured Query Language
SSH	Secure Shell
SSIM	Symantec Security Information Manager
TDEA	Triple Data Encryption Algorithm
TLS	Transport Layer Security
USB	Universal Serial Bus

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