Leonovus Cryptographic Module

Version No: 3.4

by Leonovus Inc.

FIPS 140-2 Non-Proprietary Security Policy

FIPS Security Level: 1
Document Version: 1.1
Date: February 5, 2020

Prepared For:

Leonovus

Leonovus Inc.
2611 Queensview Drive, Suite 125
Ottawa, Ontario
Canada
K2B 8K2
https://www.leonovus.com

Prepared By:

intertek ewa canada

EWA-Canada, Ltd.
1223 Michael Street, Suite 200
Ottawa, Ontario
Canada
K1J 7T2
http://www.intertek.com/cybersecurity/ewa-canada/

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Purpose

This non-proprietary Security Policy for the Leonovus Cryptographic Module v3.4 by Leonovus Inc. describes how the module meets the security requirements of FIPS 140-2 and how to run the software only module in a secure FIPS 140-2 mode.

This document was prepared as part of the Level 1 FIPS 140-2 validation of the module.The following table lists the module's FIPS 140-2 security level for each 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	1
9	Self-Tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A

Table 1 - FIPS 140-2 Section Security Levels

1.2 Background

Federal Information Processing Standards Publication (FIPS PUB) 140-2 – *Security Requirements for Cryptographic Modules* details the requirements for cryptographic modules. More information on the National Institute of Standards and Technology (NIST) and the Communications Security Establishment (CSE) Cryptographic Module Validation Program (CMVP), the FIPS 140-2 validation process, and a list of validated cryptographic modules can be found on the CMVP website:

http://csrc.nist.gov/groups/STM/cmvp/index.html

1.3 Company and Products

Leonovus provides organizations the flexibility to store and migrate their data anywhere they choose — with security and compliance assured. As a result, they can readily benefit from the

savings, efficiencies and opportunities of contemporary, emerging and future storage and cloud innovations.

1.3.1 Vault

Leonovus Vault is a multi-cloud data controller software solution that distributes the organization's data across on-premises and public cloud storage. It delivers increased data security, reduced storage footprint, and enhanced data availability and durability. Vault empowers the IT organization with the flexibility to deal with the ever-evolving cloud storage landscape without losing control, regardless of where the data resides.

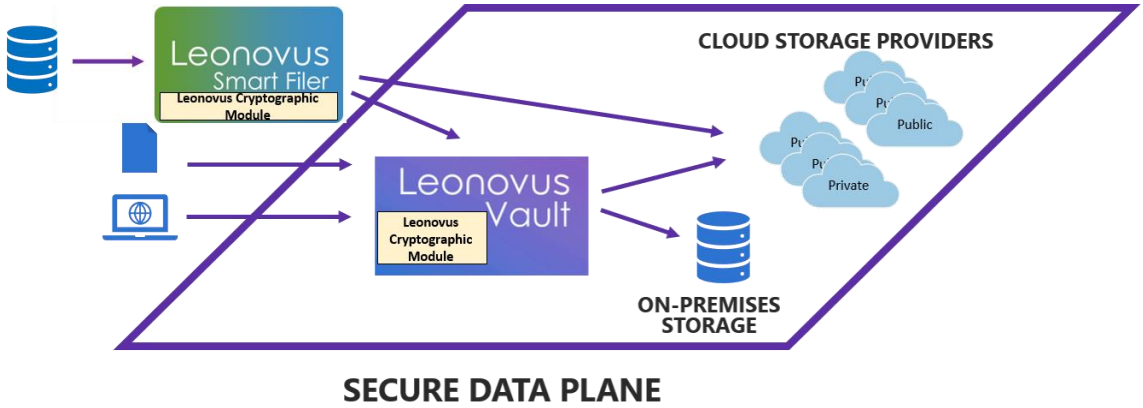

Figure 1 - Secure Data Plane

Vault instills a data-centric set of controls over the organization's storage resources, establishing a unified secure data plane decoupled from its underlying infrastructure. Applications access Leonovus Vault storage via an AWS S3-compatible, REST API interface while administrators configure security and compliance policies using an intuitive web interface that provides complete visibility into where and how data is stored.

Vault is a strategic solution, controlling and simplifying the role of data in the organization's digital transformation via cloud evolution.

1.3.2 Smart Filer

Of more immediate concern, Leonovus recognizes three key truths in the IT infrastructure environment:

- Data growth is an ever present and growing challenge,
- Storage budgets are not growing at the rate of data, often fixed or shrinking,
- Most unstructured data stored is infrequently accessed.

As a result, organizations are having to deal with continuously at-capacity or over-capacity file storage. They are striking a balance, trying to avoid buying additional hardware to keep up, without having the resources to evaluate how much of that storage is being consumed by less active but equally important data.

The Leonovus Smart Filer software solution eliminates this challenge by applying policy-based analytics and seamlessly offloading files from expensive primary storage to secondary storage in a less costly form, either on-premises, in the cloud or both. Users, applications and services reach their data in exactly the same way as they did when it resided solely on the organization's

primary storage devices, only now the data may actually be resident on less costly secondary resources such as lower cost commodity storage or private clouds on-premises or subscribed public cloud resources. The distinction is transparent to the end users and is fully controlled by IT policy instituted and executed in an ongoing basis by the Smart Filer.

1.3.3 Leonovus Solutions and Security

Whether data resources are being off-loaded by the applied data analytics of Smart Filer or they are being unified into a single data plane independent of its underlying infrastructure through Vault, simplicity, flexibility and security are the driving considerations behind Leonovus solutions. A key element of the data-centric security model which is intrinsic to each of the Leonovus solutions is IT controlled and managed cryptography. This is achieved through the common component: Leonovus Cryptographic Module (LCM).

Whether data is in-flight or at rest, whether it is whole and in one place or fragmented and distributed across the infrastructure, the LCM ensures the critical data elements are secured with state-of-the-art encryption and decryption algorithms. Because of the LCM, the IT organization retains control over the keys independent of the storage infrastructure and services. In this manner, IT regains its control over the data while greatly improving the flexibility on where it flows and resides.

1.3.4 Additional Information

More information about vendor, brand, product line, product, module can be found on the Leonovus Inc. website:

https://www.leonovus.com/

1.4 Document Organization

This non-proprietary Security Policy is part of the Leonovus Inc. Leonovus Cryptographic Module v3.4 FIPS 140-2 submission package. Other documentation in the submission package includes:

- Product documentation
- Vendor evidence documents
- Finite state model
- Additional supporting documents

The Leonovus Cryptographic Module v3.4 is also referred to in this document as the cryptographic module or the module.

2 Module Overview

2.1 Cryptographic Module Specification

The Leonovus Cryptographic Module is a standalone software module that is a cryptographic library providing cryptographic functionality for the Leonovus Vault and Smart Filer solutions. The physical boundary of the module is the boundary of the general purpose computer on which it resides. The logical cryptographic boundary is also depicted below and is defined as the API itself. The six files specified below in figure 3 are all considered within the boundary. The three .so files are compiled binaries and the corresponding .chk files contain the digital signatures used in verifying the integrity of these files. Depicted below are the module block diagrams, which indicates the direction and types of information flow between module components as well as highlighting the cryptographic boundary of the module.

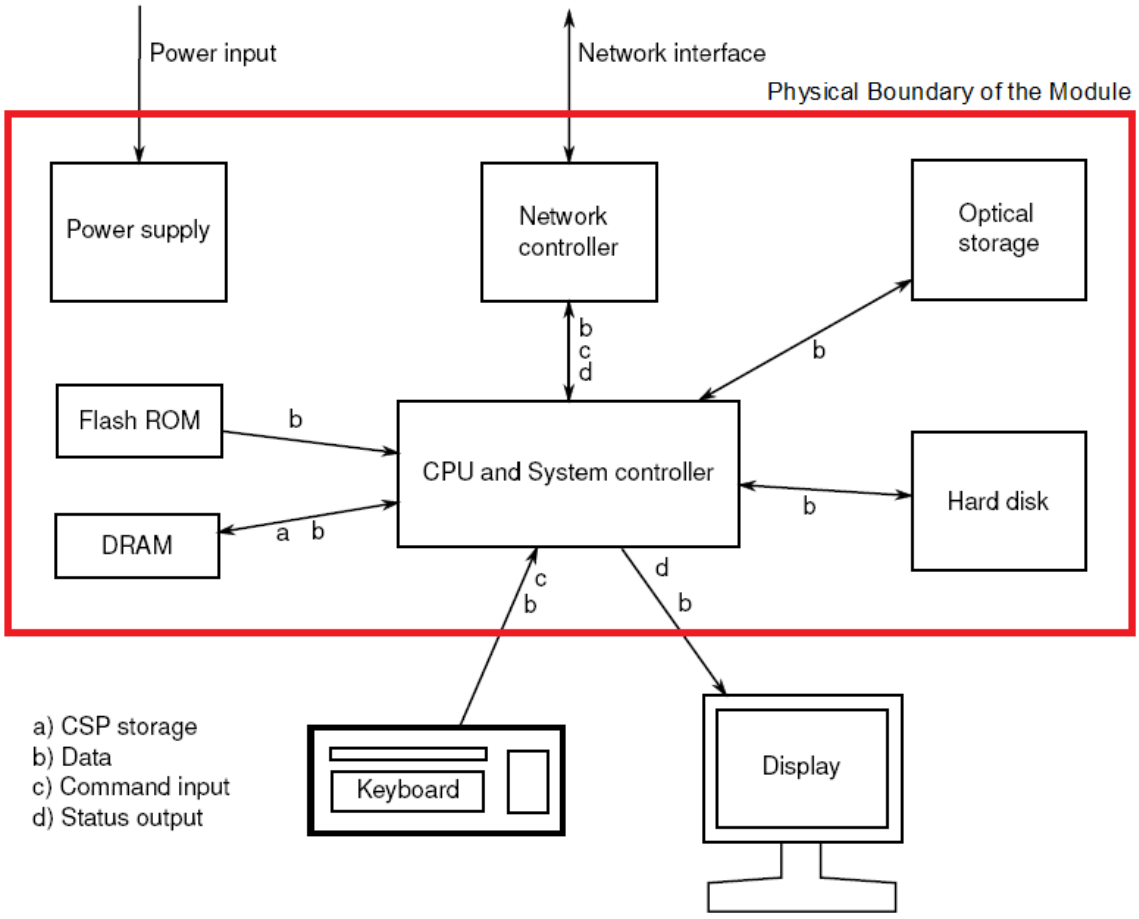

Figure 2 - Physical Diagram and Traffic Flows

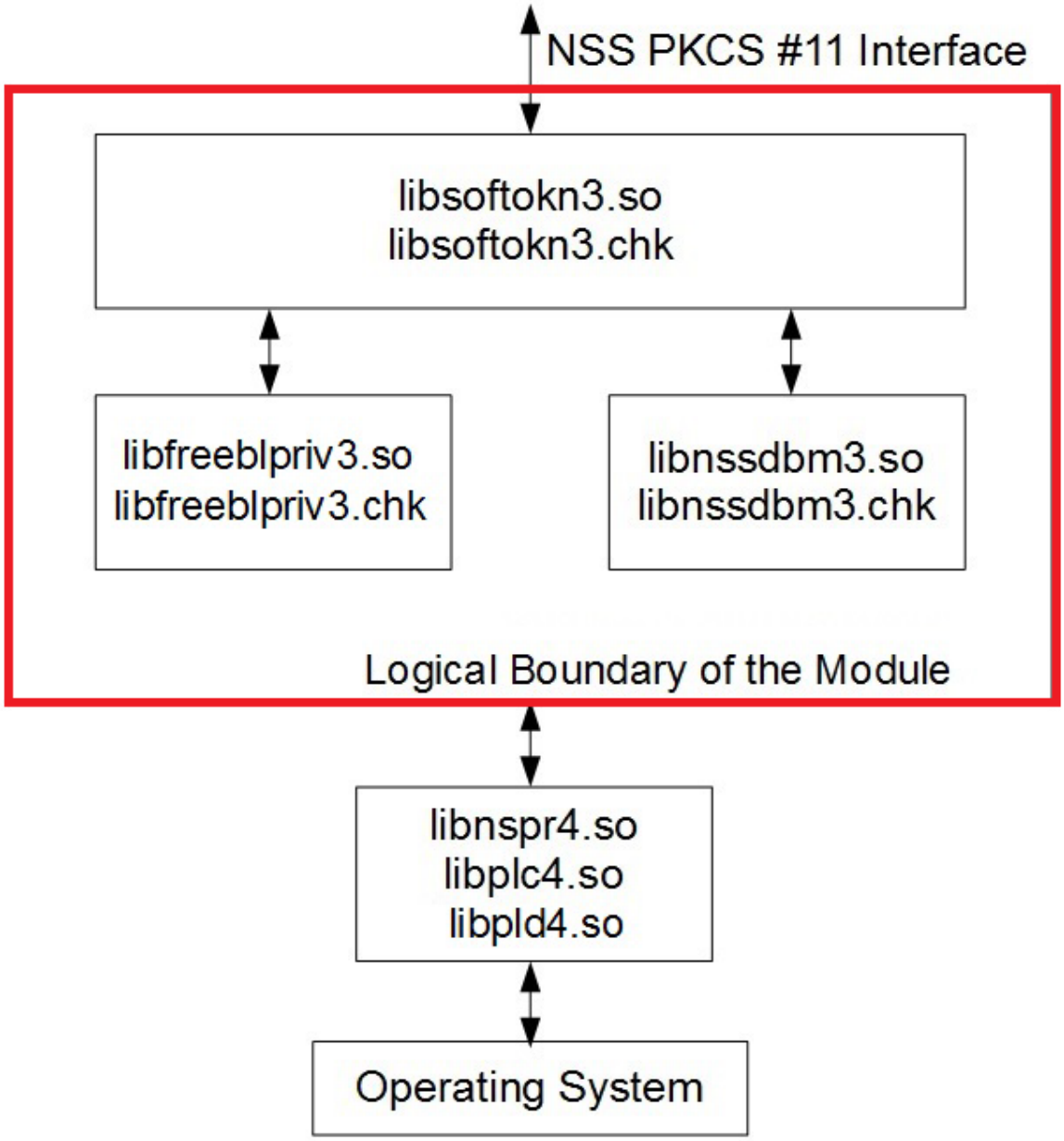

Figure 3 - Logical Boundary

2.2 Cryptographic Module Ports and Interfaces

The module's ports and interfaces that are supported when operating in FIPS mode are specified in Table 2.

FIPS 140-2 Interface	Logical Interface
Data Input	API input parameters
Data Output	API output parameters
Control Input	API function calls, environment variables and configuration files
Status Output	API return codes and status parameters
Power	Wall power source

Table 2 - Module Logical Interfaces

The module is a software module and as such, does not have its own hardware ports. It relies, rather, on the General Purpose Computer (GPC) to provide the hardware ports. Table 3 is a mapping of the module's logical interfaces to the GPC's physical ports.

FIPS 140-2 Interface	Physical Interface
Data Input	Ethernet, USB ports, keyboard
Data Output	Ethernet, networking ports
Control Input	Keyboard, ethernet
Status Output	Display screen
Power	Wall power source

Table 3 - Module Interface Mapping

The module uses different function arguments for input and output to distinguish between data input, control input, data output, and status output. This allows the module to distinguish between the logical paths followed by data/control input entering the module and data/status output exiting the module. The module doesn't use the same buffer for input and output. After the module is done with an input buffer that holds security-related information, it always zeroizes the buffer so that if the memory is later reused as an output buffer, no sensitive information can be inadvertently leaked.

2.3 Roles & Services

2.3.1 Roles

The module supports two roles, the Crypto Officer (CO) role and the User role. The module is security level one and does not not claim authentication. Both the CO and User roles are assumed implicitly based on the services provided to them and invoked by the operator.

The CO role is responsible for installation and initialization of the module.The CO controls the access to the module both before and after installation, including management of physical access to the computer, executing the module code and management of the security facilities provided by the operating system. The CO role can access other general-purpose services and status services of the module. The CO does not have access to any service that utilizes the secret keys of the module.

The User role has access to all cryptographic services in the module.

2.3.2 Services

Table 4 below specifies the services that are available to a module operator. In the Access column, Read (R) means that CSP(s) is used by the module to perform the service, while Zeroized (Z) is used to denote that the CSPs used have been zeroized.

Service	Role	Function	Description	Keys/CSPs	Access
General	CO	FC_Initialize	Initialize the module	none	-
	CO	FC_Finalize	Finalize (shut down) the module	AES Key	Z
Self-tests	CO	-	The self tests are performed automatically when loading or power-cycling the module	None	R
Status	CO, U	FC_GetFunctionList, FC_GetSessionInfo	Status of module	none	R
Encryption and Decryption	U	FC_EncryptInit	Initialize an encryption operation	AES key	R
	U	FC_Encrypt	Encrypt single-part data	AES key	R
	U	FC_EncryptUpdate	Continue a multiple-part encryption operation	AES key	R
	U	FC_EncryptFinal	Finish a multiple-part encryption operation	AES key	R
	U	FC_DecryptInit	Initialize a decryption operation	AES key	R
	U	FC_Decrypt	Decrypt single-part encrypted data	AES key	R
	U	FC_DecryptUpdate	Continue a multiple-part decryption operation	AES key	R
	U	FC_DecryptFinal	Finish a multiple-part decryption operation	AES key	R
	U	FC_DigestEncryptUpdate	Continue a multiple-part digesting and encryption operation	AES key	R
	U	FC_DecryptDigestUpdate	Continue a multiple-part decryption and digesting operation	AES key	R
Zeroization	U	FC_DestroyObject	All CSPs are automatically zeroized when freeing the cipher handle	AES Key	Z
	CO	FC_InitToken FC_Finalize FC_CloseSession FC_CloseAllSessions			

Table 4 - Services

2.4 Physical Security

The module is a security level 1 software FIPS module and does not claim physical security protections.

2.5 Operational Environment

The module executes on a General Purpose Computer (GPC). The module was tested on CentOS 7.6 running on VMware ESXi 6.0. The processor used to test the module is an Intel Xeon E5-2403 and the GPC used is a HP ProLiant DL360e Gen8.

The operational environment is restricted to a single user. Concurrent operators are not allowed. In operational mode, the ptrace system call, the debugger gdb, and strace shall be not used. In addition, other tracing mechanisms offered by the Linux environment, such as ftrace or systemtap, shall not be used. The module does not allow new firmware or software to be loaded.

2.6 Cryptographic Key Management

2.6.1 Algorithm Implementations

A list of FIPS-Approved algorithms implemented by the module can be found in Table 5.

CAVP Cert	Algorithm	Standard	Mode/ Method	Key Lengths, Curves or Moduli	Use
#C654	AES	FIPS 197, SP800-38A	CBC	256 bits	Data Encryption and Decryption
#C654	AES	FIPS 197, SP800-38A	CTR	256 bits	Data Encryption and Decryption
#C654	DSA	FIPS 186-4	SigVer	Mod 2048 with SHA-256	Digital Signature Verification (Used solely for Software Integrity)
#C654	SHS	FIPS 180-4	SHA-256	-	Message Digest (Used solely for Software Integrity)

Table 5 - FIPS-Approved Algorithm Implementations

2.6.2 Key Management Overview

CSP Name	Generation	Storage Method	Entry/Output	Storage	Zeroization
AES Key (256 bits)	Input through API	Plaintext	N/A	Application memory	Automatically zeroized by freeing the cipher handle

Table 6 - Cryptographic Keys, Key Components, and CSPs

2.6.3 Storage

The module does not perform persistent storage for any keys.

2.6.4 Zeroization

The application that uses the module is responsible for appropriate zeroization of the key material. The module provides zeroization methods to
clear the memory region previously occupied by a plaintext secret key. A plaintext secret key gets zeroized when it is passed to an FC_DestroyObject call. All plaintext secret keys must be zeroized when the module is shut down (with an FC_Finalize call), reinitialized (with an FC_InitToken call), or when the session is closed (with an FC_CloseSession or FC_CloseAllSessions call). All zeroization is to be performed by storing the value 0 into every byte of the memory region that is previously occupied by a plaintext secret key. Zeroization is performed in a time that is not sufficient to compromise plaintext secret keys.

During key zeroization, the Module may perform audit logging, but the audit records do not contain sensitive information. The Module does not return the function output arguments until the key zeroization is fnished. Therefore, the logical paths used by output data exiting the module are logically disconnected from the processes/threads performing key zeroization.

2.6.5 Key Output

The cryptographic module does not allow plaintext cryptographic key components or other unprotected CSPs to be output from the physical cryptographic boundary.

2.7 Electromagnetic Interference / Electromagnetic Compatibility

The module resides on a GPC. The GPC used during testing (an HP ProLiant DL360e Gen8) conforms to the FCC EMI/EMC requirements in 47 Code of Federal Regulation, Part 15, Subpart B, Unintentional Radiators, Digital Devices, Class A.

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user will be required to correct the interference at personal expense.

2.8 Self Tests

2.8.1 Power Up Self Tests

The modules perform the following tests automatically upon power up:

Algorithm	Type	Description
AES	KAT[1]	Encryption and decryption are tested separately, using CBC mode, 256-bit length
SHS	KAT	SHA-256 KAT
Module Integrity	KAT	DSA Signature Verification – key size 2048-bit with SHA-256

Table 7 - Power-On Self-Tests

All data output via the data output interface is inhibited when the module is performing self-tests or in the Error state. During self-tests, all data output via the data output interface is inhibited while the self-tests are being executed.

While in the “Error” state, the Boolean state variable sftk_fatalError tracks whether the module is in the Error state. All PKCS #11 functions that output data via the data output interface, check the sftk_fatalError state variable and, if it is true, return the CKR_DEVICE_ERROR error code immediately. Only the functions that shut down and restart the module, reinitialize the module, or output status information can be invoked in the Error state. These functions are FC_GetFunctionList, FC_Initialize and FC_Finalize.

When the module enters the Error state, it needs to be reinitialized to resume normal operation. Reinitialization is accomplished by calling FC_Finalize followed by FC_Initialize (power-cycling the module).

2.9 Design Assurance

Leonovus uses git for source code control. Code repositories are hosted internally using Atlassian Bitbucket Server (version v5.11.1).

The Leonovus CM (git) refers to each change made to the set of files under its management as a "commit". Each commit is checksummed and identified by that checksum when it is checked back out. See https://git-scm.com/about/info-assurance for more information on this.

Documentation version control is performed manually by updating the document date as well as the major and minor version numbers in order to uniquely identify each version of a document.

2.10 Mitigation of Other Attacks

The module does not mitigate attacks other than the requirements for FIPS 140-2.

[1] KAT: Known Answer Test

3 Secure Operation

The Leonovus Cryptographic Module always operates in a FIPS Approved mode of operation. The module is built and compiled to always operate in the Approved mode of operation and no specific operator actions or calls to the module are necessary in order to place the module into FIPS mode.

3.1 Crypto-Officer Guidance

There is no specific Crypto Officer Guidance. The module is distributed to the operator as part of a pre-configured virtual machine with other Leonovus software. Installation of the module is done by powering on the virtual machine and configuring it to an operational state.

3.2 Access to Audit Data

The module may use the Unix syslog function and the audit mechanism provided by the operating system to audit events. Auditing is turned off by default. Auditing capability must be turned on as part of the initialization procedures by setting the environment variable NSS_ENABLE_AUDIT to 1.

The Crypto-Officer must also configure the operating system's audit mechanism. The module uses the syslog function to audit events, so the audit data are stored in the system log. Only the root user can modify the system log. On some platforms, only the root user can read the system log; on other platforms, all users can read the system log. The system log is usually under the /var/log directory. The exact location of the system log is specified in the /etc/syslog.conf file. The module uses the default user facility and the info, warning, and err severity levels for its log messages.

The module can also be configured to use the audit mechanism provided by the operating system to audit events. The audit data would then be stored in the system audit log. Only the root user can read or modify the system audit log. To turn on this capability it is necessary to create a symbolic link from the library file /usr/lib64/libaudit.so.0 to /usr/lib64/libaudit.so.1.0.0 (on 64-bit platforms).

3.3 User Guidance

Secret keys and other security-relevant data items are maintained under the control of the cryptographic module.

All cryptographic keys used in the FIPS Approved mode of operation must be input by the calling Application while running in the FIPS Approved mode.

If the module enters the Error state, it needs to be reinitialized to resume normal operation. Reinitialization is accomplished by calling FC_Finalize followed by FC_Initialize.

4 Acronyms

Acronym	Definition
AES	Advanced Encryption Standard
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CMVP	Cryptographic Module Validation Program
CO	Crypto Officer
CSE	Communications Security Establishment
CTR	Counter Mode
CSP	Critical Security Parameter
DSA	Digital Signature Algorithm
EMI/EMC	Electromagnetic Interference / Electromagnetic Compatibility
FCC	Federal Communications Commission
FIPS	Federal Information Processing Standards
GPC	General Purpose Computer
KAT	Known Answer Test
NIST	National Institute of Standards and Technology
NSS	Network Security Services
NVM	Non-Volatile Memory
RAM	Random Access Memory

Table 8 – Acronyms

