SenSage, Inc.

CryptoCore Module

Software Version: 1.0

FIPS 140-2 Non-Proprietary Security Policy

FIPS Security Level: 1
Document Version: 0.4

Prepared for:

SenSage, Inc.
55 Hawthorne Street
San Francisco, CA 94105
United States of America

Phone: +1(415) 808-5900
Email: info@sensage.com
http://www.sensage.com

Prepared by:

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

Phone:+1 (703) 267-6050
Email: info@corsec.com
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 CryptoCore Module from SenSage, Inc. This Security Policy describes how the CryptoCore 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 CryptoCore Module is referred to in this document as the SenSage CryptoCore Module, 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 SenSage website (www.sensage.com) contains information on the full line of solutions from SenSage.
- 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 document
- Other supporting documentation as additional references

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

2 CryptoCore Module

2.1 Overview

SenSage offers Event Data[1] Warehouse solutions that handle massive amounts of log and event data. Event data contains evidence directly pertaining to and resulting from the execution of a business process or system function. Below are several examples of systems or devices that generate event data:

- Network and security devices
- Physical access systems
- Identity management systems
- Workstations, servers, and operating systems
- Enterprise applications – 3rd party and in-house

In addition to the systems and devices mentioned above, event data that can be collected and stored from these systems and devices are:

- Database activity
- Email, Windows, network and other systems management activity events
- Banking transactions such as online, ATM[2] and debit card use
- Historical prices of stocks and other financial instruments
- Telephone Call Detail Records (CDRs[3])
- Internet Protocol Detail Records (IPDRs) of web-based access and transactions

When properly configured, the Event Data Warehouse contains the records of all system activities including users logging in and logging out, users accessing confidential files, activities on the firewall, emails being sent and received, information on processed transactions, and the web sites being accessed.

Given the massive daily volumes of audit logs and the variety of sources across the network in a typical enterprise computing environment, efficiently collecting and aggregating all relevant events in a structured way for analysis can be a challenging task. SenSage solutions enable the user to easily collect and store large volumes of event data. They also provide the user an ability to query and perform analysis on the event data that are available.

The SenSage solution consists of several separate executable components. Figure 1 below shows a view of the deployed solution with the SenSage solution components shown in blue. There are four acronyms that are used in the diagram below that have not been defined in this document yet:

- LAN/WAN – Local Area Network/Wide Area Network
- CLI – Command Line Interface
- OS – Operating System
- SLS – Scalable Log Server

[1] Event Data – also referred to as an "audit trail" or "system of record"; this is a set of chronologically sequenced data records that capture information about an event.
[2] ATM – Automatic Teller Machine
[3] CDR – A Call Detail Record (CDR) is the computer record produced by a telephone exchange containing details of a call. It is the automated equivalent of the paper toll tickets that were written and timed by operators for long distance calls in a manual telephone exchange.

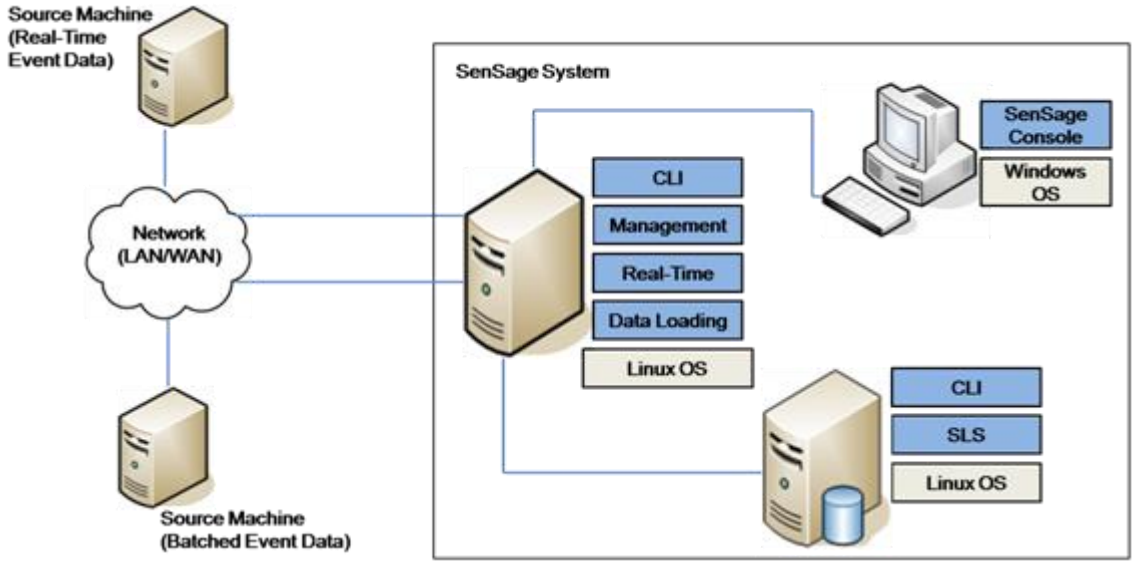


Figure 1 - SenSage Deployment

The following paragraphs provide a brief description of the components of the SenSage solution.

2.1.1 Scalable Log Server Component

The SenSage solution is composed of several components, the most important of these being the Scalable Log Server. The SLS is a high-performance, read-only columnar database with clustering ability. The SLS is coded in the C++ programming language, and is compiled and run on Red Hat Enterprise Linux 5.1 or 5.5. It uses the application level clustering technique to perform all load and query tasks in parallel, across any number of SLS database instances. This architecture allows users to load and query massive volumes of data in a single, logical database instance without partitioning. The SLS stores user data (*i.e.* the event data from source machines) in a special file system directly on disk. It also generates its own audit logs.

2.1.2 Management Component

The Management Component of the SenSage solution is responsible for processing the user requests. It acts as a communication agent between the SenSage Console and the SLS. The Management Component performs the following tasks:

- Authenticates, authorizes, and maintains SenSage Console sessions.
- Manages definitions for reports, libraries[4], dashboards, namespaces, schedules, users, roles and permissions.
- Stores rules for parsing the event data, rules for analyzing the event data, and conditions for triggering alerts.
- Processes queries against the SLS.
- Sends reports, alerts, and email with scheduled reports to SenSage Console.

2.1.3 Data Loading Component

Before the user is able to analyze event data, original event data must be collected into the SenSage system. The event data needs to be gathered and loaded into the SLS Component.

There are two ways in which the SenSage solution collects the event data:

[4] Library – A group of shared code. SenSage 4.6 enables users to create libraries, which allow common SQL fragments and Perl codes to be shared across queries.

- Batches – events are collected from log files and other event repositories maintained by network devices and software applications. The Data Loading Component is responsible for loading the batches of event data into the SLS Component.

- Streams – events flow into the SenSage system as a real-time stream of event-log entries from network devices and software applications that generate the events. The Real Time Component is responsible for loading the streams of real-time event data into the SLS Component.

As stated above, it is the Data Loading Component that gathers the batches of log data from the source, parses and transforms the log file, and loads the data into a specified SLS instance, namespace and table.

2.1.4 Real Time Component

The Real Time Component of the SenSage solution is responsible for collecting the real-time event data that is sent to the SenSage system by the source machines. The Real Time Component receives the event data streams by listening on designated network ports. It accepts event data from various systems, reformats them into a standard format, and then parses them into a normalized data structure so they can be ready for loading into the database in the SLS Component.

In addition to performing these tasks, the Real Time Component performs real-time correlation analysis on parsed normalized event data. If the analysis detects a predefined condition for raising alerts, the Real Time Component relays the information to the Management Component, which in turn sends the security alerts to the SenSage Console Component.

2.1.5 SenSage Console Component

The SenSage Console Component is a Java application installed on the user's Windows workstation. Since the SenSage Console Component is written in Java, the SenSage Console Component provides the end-users a graphical user interface for monitoring, analyzing, resolving, and reporting real-time and historical event data.

2.1.6 CLI Component

The CLI (Command Line Interface) Component of the solution is responsible for providing a CLI through which the administrative users of the solution and the solution system processes can execute SenSage-associated commands and scripts. The CLI is used for administering the SLS, Management, Real-Time, and Data Loading Components which run on Linux OS machines.

2.1.7 Deployment

These solution components can be deployed on a single Linux machine or they can be deployed across a large number of Linux platform machines. When deployed over multiple machines, depending on the volume of event data being stored and processed, the deployment can have multiple instances of the Scalable Log Server Component, which is a proprietary columnar database on a Linux machine that serves as an event data repository. It should be noted that the actual deployment is determined by the customer's network architecture and performance requirements; therefore, the configuration varies by each customer's computing environment.

The SenSage solution includes the ability to enable the use of Transport Layer Security (TLS) to secure network connections. The SenSage solution also has the ability to enable protection of data-at-rest within the system using symmetric key cryptography; the data-at-rest solution is called the SenSage Private Encrypted File System (PEFS).

The scope of this validation is limited to the SenSage CryptoCore Module, which provides the cryptographic functionality for all SenSage solution components, such as the SLS. The CryptoCore

Module provides the solution components with the cryptographic services necessary to protect network communication through TLS and also data-at-rest through PEFS.

The CryptoCore Module is validated at the following FIPS 140-2 Section levels shown in Table 1.

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[5]
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
14	Cryptographic Module Security Policy	1

2.2 Module Specification

The CryptoCore Module is a software-only module that operates within a multi-chip standalone embodiment, such as a General Purpose Computer (GPC). The overall security level of the module is 1. The physical cryptographic boundary is defined as the physical perimeter of the GPC on which the module is installed; where as the logical cryptographic boundary of the CryptoCore Module includes the following components as depicted in

Figure 2:

- SenSage CryptoCore Library (libCryptoCore.so, libCryptoCore.so.hmac)
- Cryptographic Library (libcrypto.so, libcrypto.so.hmac)
- TLS Library (libssl.so, libssl.so.hmac)

The cryptographic components of the CryptoCore Module allow for safe and secure data collection throughout the SenSage solution applications (components listed in Section 2.1 (Overview)). The communication between the SenSage solution applications and SenSage CryptoCore Module can be seen in

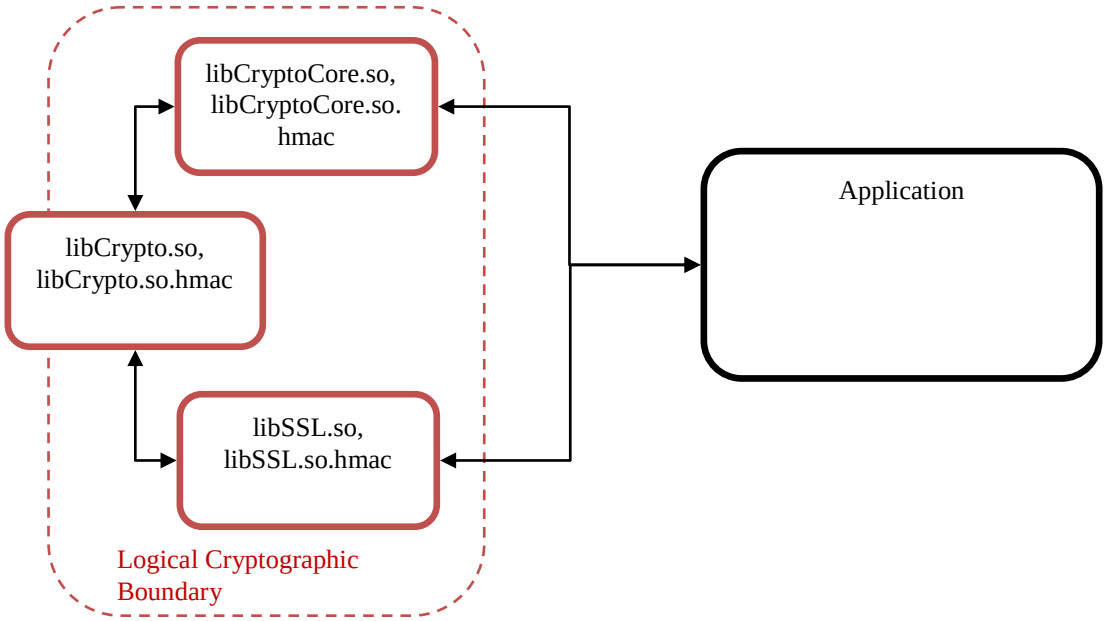
Figure 2 below. In

Figure 2, the various SenSage solution applications call SenSage CryptoCore Module for cryptographic services.

[5] N/A – Not Applicable
[6] EMI/EMC – Electromagnetic Interference / Electromagnetic Compatibility

The module supports both a FIPS-Approved mode and non-FIPS-Approved mode of operation. In order to determine the mode of operation, the operator may invoke CryptoGlobal::gCryptoGlobal->isFIPSMode, which will indicate whether FIPS-Approved mode has been enabled. The operator may also examine the configuration file (crypto.cfg) to verify that "gFIPSTag" has been set to a non-zero value or that the "FIPS" value has been set to 1, but a call to CryptoGlobal::gCryptoGlobal->isFIPSMode provides the most assurance.

Figure 2 – FIPS 140-2 Logical Block Diagram



The cryptographic module was tested and found compliant on the following platforms:

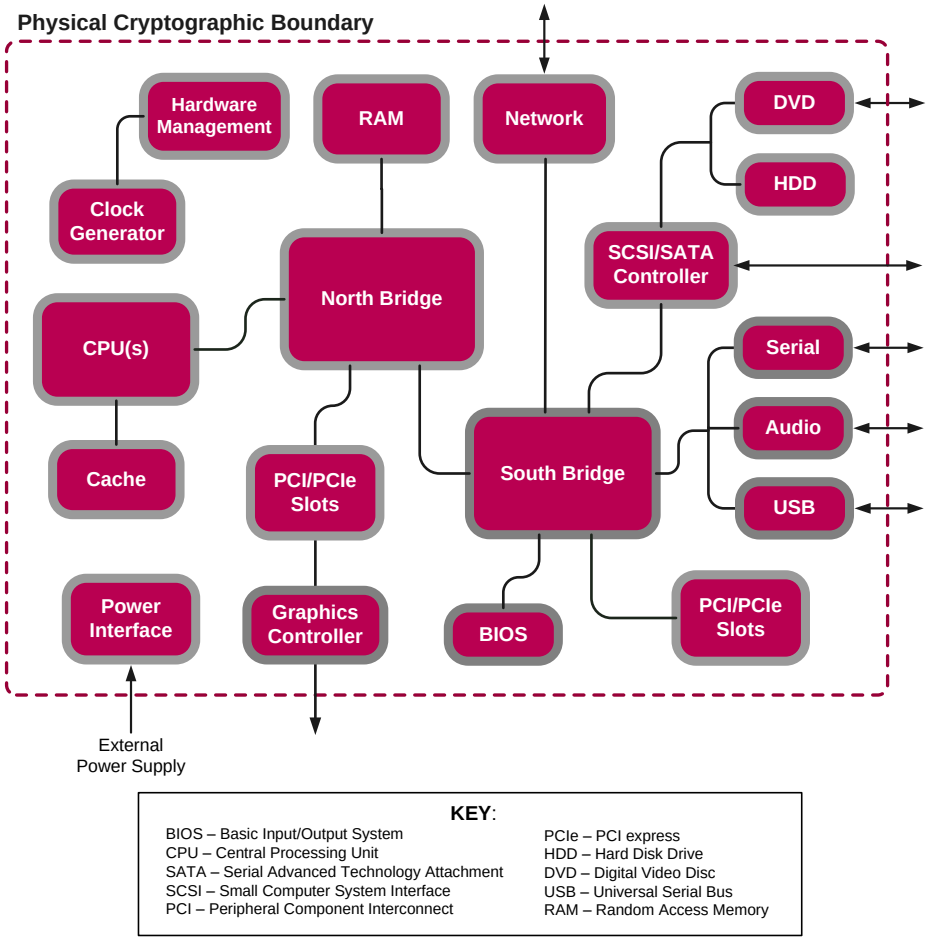
- Red Hat Enterprise Linux (RHEL) 5.1
- RHEL 5.5

2.3 Module Interfaces

The module's interfaces are provided by the logical application programming interface (API), which provides the data input, data output, control input, and status output logical interfaces defined by FIPS 140-2. The module is installed on a GPC with physical ports consistent with that of a GPC as depicted in Figure 3.

.

Figure 3 – FIPS 140-2 GPC Block Diagram



The mapping of logical interfaces to the physical ports of the GPC is provided in Table 2 below.

Table 2 – FIPS 140-2 Logical Interface Mappings

FIPS 140-2 Logical Interface	Port/Interface Name	Module Interface
Data Input	Network, DVD, SCSI/SATA Controller, Serial, USB	API
Data Output	Network, SCSI/SATA Controller, Serial, USB, Graphics Controller	API
Control Input	Network, DVD, Serial, USB	API
Status Output	Audio, Graphics Controller	API
Power	Power Interface	N/A

.

2.4 Roles and Services

The CryptoCore Library is a software only module that provides an API for applications to implement the PEFS or support TLS. As such, authentication is not provided by the library, though it may be enforced by the calling applications. The module has been designed to comply with FIPS 140-2 Level 1 requirements only, which does not require authentication. The module may be used to support two roles that operators may assume when authenticating to a Secure Key Vault (SKV): a Crypto-Officer role and a User role, which are implicitly assumed.

2.4.1 Crypto-Officer Role

The Crypto-Officer role has the ability to manage Users, the Master Keys, and the SKV, which is a secure container for encrypted cryptographic keys and user information. Descriptions of the services available to the Crypto-Officer role are provided in the table below. The Crypto-Officer may also perform any service available to the User role. Please note that the keys and critical security parameters (CSPs) listed in the table indicate the type of access required using the following notation:

- R – Read: The CSP is read.
- W – Write: The CSP is established, generated, modified, or zeroized.
- X – Execute: The CSP is used within a FIPS-Approved or Allowed security function or authentication mechanism.

Table 3 – Crypto-Officer Services

Service	Description	CSP and Type of Access
Create SKV	Creates a new SKV and initializes the User Lock Box (contains the User Key and optional Management Key), Management Block (contains User Key, optional Management Key, and Directory Private Key), and Master Key Block (contains the Master Keys, HMAC[7] Key, and Directory Public Key).	W Passphrase R/W/X Personal Access Key W/X User Key W/X Management Key R/W/X Directory Keys R/W/X HMAC Key R/W/X Master Key
List Users	Returns a list of valid users for a given SKV.	R Management Key X Directory Key R/X Personal Access Key
Add User	Adds a new user to the SKV and invokes a SKV rewrite, which re-generates the encrypted Master Key Block, Management Block, and the associated HMAC.	R/W/X Management Key R/W/X User Key R/X Directory Key W Personal Access Key R/X HMAC Key

[7] HMAC - Hash-based Message Authentication Code

.

Service	Description	CSP and Type of Access
Delete User	Deletes a user from the SKV and invokes a SKV rewrite.	R/X Management Key X Directory Key R/X User Key R/X HMAC Key
Change User Privilege	Changes the privilege flag for a user and invokes a SKV rewrite.	R Management Key X Directory Key X Personal Access Key R/W/X Management Key R/X User Key R/X HMAC Key
Change User Passphrase	Changes the passphrase for a user and invokes a SKV rewrite.	R/X Management Key W/X Personal Access Key R/W/X Passphrase X Directory Key R/X User Key R/X HMAC Key
Rotate Master Key	Creates a new master key and makes it the 'current' master key and invokes a SKV rewrite.	R/X Management Key W Master Key R/X User Key R/X HMAC Key
Rekey	Regenerates internal keys and invokes a SKV rewrite.	R/W/X Directory Key R/W/X Management Key R/W/X User Key R/W HMAC Key R/X Personal Access Key
Change Master Key Spec	Changes the master key spec, and invokes a SKV rewrite.	R/X Management Key R/X User Key R/X HMAC Key
Reset Master Key	Resets a Master Key so that it cannot be used and invokes a SKV rewrite.	R/X Management Key R/X User Key R/X HMAC Key
Zeroization	Zeroization of temporary keys through system reboot or unloading module from memory	W TLS Session Encryption Key W TLS Session Integrity Key W TLS Pre-Master Secret W TLS Master Secret W TLS DH Private Parameter W TLS DH Public Parameter W TLS RSA Public/Private Key W TLS Peer RSA Public Key W TLS DSA Public/Private Key W TLS DSA Peer Public Key W DRNG Seed Value/Key

2.4.2 User Role

The User role has the ability to perform the basic SKV and PEFS operations. Descriptions of the services available to the User role are provided in Table 4 below.

.

Table 4 – User Services

Service	Description	CSP and Type of Access
Encrypt Payload Header	Encrypts the indicated Payload Header.	X Master Key
Decrypt Payload Header	Decrypts the indicated Payload Header.	X Master Key
Open SKV	Open an existing SKV and verify its integrity.	R/X Passphrase W/X Personal Access Key R/X User Key R/X Management Key R/X HMAC Key
Change My Passphrase	Change user's own passphrase and rewrite the SKV.	W/X Personal Access Key R/X/W Passphrase R/X Directory Key R/X Management Key R/X User Key R/X HMAC Key
File Reader	Create and use a File Reader for plaintext or encrypted files, respectively.	R/X Master Key R/X Private Data Key R/X HMAC Key
File Writer	Create and use a File Writer for plaintext or encrypted files, respectively.	R/X Master Key R/W/X Private Data Key R/W/X HMAC Key
Blob Encryptor	Encrypt an arbitrary blob of information using a key derived from a passphrase using PKCS#5.	R Passphrase W/X Personal Access Key
Blob Decryptor	Decrypt an arbitrary blob of information using a key derived from a passphrase using PKCS#5.	R Passphrase W/X Personal Access Key
Show Status	Provide status information through return codes and exceptions.	N/A
Self-Tests	Invokes the suite of FIPS 140-2 self-tests by restarting the module.	N/A
TLS	Maintain a secure communication channel.	R/W/X TLS Session Encryption Key, TLS Session Integrity Key, TLS RSA Private Key, TLS DSA Private Key, DRNG Seed Value, DRNG Seed Key
TLS Key Agreement	Establish the TLS Session Keys for secure communication.	R/W/X TLS Session Encryption Key, TLS Session Integrity Key, TLS Pre-Master Secret, TLS Master Secret, TLS DH Private Parameter, TLS RSA Private Key, TLS DSA Private Key, DRNG Seed Value, DRNG Seed Key

2.5 Physical Security

The CryptoCore Module is a software only module, which operates on a multi-chip standalone device, such as a GPC. As such, it does not include physical security mechanisms and the FIPS 140-2 requirements for physical security are not applicable.

2.6 Operational Environment

The module meets the FIPS 140-2 Operational Environment requirements for a Level 1 device. The module can be run on Red Hat Enterprise Linux 5.1 and 5.5. The operating system must be configured for single user mode for FIPS 140-2 compliance (see Section 3.2 for guidance).

All cryptographic keys and CSPs are under the control of OS, which protects the CSPs against unauthorized disclosure, modification, and substitution. The module only allows access to CSPs through its well-defined APIs.

2.7 Cryptographic Key Management

The module implements the FIPS-Approved algorithms listed in Table 5 below.

Table 5 – FIPS-Approved Algorithm Implementations

Algorithm	Certificate Number
AES[8] 128/192/256 in ECB[9]/CBC[10]/CFB[11]/OFB[12] modes	#1761
Triple-DES[13] (TDES) 128/192 in CBC/CFB/OFB modes	#1140
RSA[14] (X9.31, PSS, PKCS#1) key generation, signature generation/verification – 1024 , 1536 , 2048 , 3072 , 4096	#877
DSA[15] (FIPS 186-2) signature generation/verification - 1024	#551
SHA[16]-1, SHA-224, SHA-256, SHA-384, SHA-512	#1545
HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC- SHA-384, HMAC-SHA-512	#1032
NIST-Recommended ANSI X9.31 Deterministic Random Number Generator (DRNG) using AES	#938

Additionally, the module utilizes the following non FIPS-Approved algorithm implementation:

- Diffie-Hellman (key agreement, key establishment methodology provides between 80 and 219 bits of encryption strength)
- HMAC MD5[17] and MD5 within TLS only or in non FIPS-Approved mode

[8] AES - Advanced Encryption Standard
[9] ECB - Electronic Code Book
[10] CBC - Cipher-Block Chaining
[11] CFB - Cipher Feedback
[12] OFB - Output Feedback
[13] DES - Data Encryption Standard
[14] RSA - Rivest, Shamir and Adleman
[15] DSA - Digital Signature Algorithm
[16] SHA - Secure Hash Algorithm

.

- DES, CAST5[18], Blowfish in non FIPS-Approved mode
- RSA (key wrapping; key establishment methodology provides between 80 and 256 bits of encryption strength)

[17] MD5 - Message-Digest algorithm 5
[18] CAST5 – Carlisle Adamas and Stafford, alternatively called CAST-128

.

The module supports the critical security parameters (CSPs) listed below in Table 6.

Table 6 – List of Cryptographic Keys, Cryptographic Key Components, and CSPs

CSP	Generation / Input	Output	Storage	Zeroization	Use
Passphrase	Not Applicable (N/A).	N/A	Not persistently stored.	N/A. Not persistently stored.	Used to derive the Personal Access Key through PKCS#5.
Personal Access Keys (AES or TDES)	Internally derived using PKCS#5. (See SP 800-132 for more information)	N/A. Never output beyond the physical boundary of the GPC by the module.	SKV, wrapped by the Directory Public Key in the User Info Block.	N/A. Encrypted.	Encrypts the User Lock Boxes within the SKV. Each User Lock Box contains a User Key and potentially a Management Key.
User Key (AES or TDES)	Internally, using the FIPS-Approved ANSI X9.31 DRNG.	N/A. Never output beyond the physical boundary of the GPC by the module.	SKV, encrypted by the Personal Access Key in the User Lock Box and by the Management Key in the Management Block.	N/A. Encrypted.	Each User Key encrypts a Master Key Block, which contains Master Keys, an HMAC Key, and a Directory Public Key.
Management Key (AES or TDES)	Internally, using the FIPS-Approved ANSI X9.31 DRNG.	N/A. Never output beyond the physical boundary of the GPC by the module.	SKV, encrypted by the Personal Access Key and by the Management Key in the Management Block.	N/A. Encrypted.	Encrypts the Directory Private Key and protected configuration information.

.

CSP	Generation / Input	Output	Storage	Zeroization	Use
Master Keys (AES or TDES)	Internally, using the FIPS-Approved ANSI X9.31 DRNG.	N/A. Never output beyond the physical boundary of the GPC by the module.	SKV, encrypted by the User Key within the Master Key Block.	N/A. Encrypted.	Encrypts the payload header of an encrypted file, which contains the HMAC Key and Private Data Key.
HMAC Keys (HMAC)	Internally, using the FIPS-Approved ANSI X9.31 DRNG.	N/A. Never output beyond the physical boundary of the GPC by the module.	Encrypted by the Master Key within the header of an encrypted file. HMAC Keys are also stored in the SKV, encrypted by the User Key within the Master Key Block.	N/A. Encrypted.	Provides data integrity.
Private Data Keys (AES or TDES)	Internally, using the FIPS-Approved ANSI X9.31 DRNG.	N/A. Never output beyond the physical boundary of the GPC by the module.	Encrypted by the Master Key within the header of an encrypted file.	N/A. Encrypted.	Encrypts the data payload of an individual encrypted file.
Directory Public Key (RSA)	Internally, using the FIPS-Approved ANSI X9.31 DRNG.	N/A. Never output beyond the physical boundary of the GPC by the module.	SKV, encrypted by the User Key within the Master Key Block.	N/A. Encrypted.	Wraps the User Info Block, which contains the Personal Access Keys.

.

CSP	Generation / Input	Output	Storage	Zeroization	Use
Directory Private Key (RSA)	Internally, using the FIPS-Approved ANSI X9.31 DRNG.	N/A. Never output beyond the physical boundary of the GPC by the module.	SKV, encrypted by the Management Key in the Management Key Block.	N/A. Encrypted.	Unwraps the Personal Access Keys stored in the User Info Block within the SKV.
TLS Session Encryption Key	Established during the TLS handshake.	N/A. Never output beyond the physical boundary of the GPC by the module.	Volatile memory only.	N/A. Not persistently stored, but zeroized upon session termination.	Provides confidentiality of session data.
TLS Session Integrity Key	Established during the TLS handshake.	N/A. Never output beyond the physical boundary of the GPC by the module.	Volatile memory only.	N/A. Not persistently stored, but zeroized upon session termination.	Provides data integrity over session data.
TLS Pre-Master Secret	Established during the TLS handshake. The pre-master secret is either agreed upon using Diffie-Hellman or it is established using RSA Key Transport.	N/A. Never output beyond the physical boundary of the GPC by the module.	Volatile memory only.	N/A. Not persistently stored, but zeroized upon session termination.	Used to establish the TLS Master Secret.
TLS Master Secret	Established during the TLS handshake based on the pre-master secret.	N/A. Never output beyond the physical boundary of the GPC by the module.	Volatile memory only.	N/A. Not persistently stored, but zeroized upon session termination.	Used to establish the TLS Session Encryption Key and TLS Session Integrity Key.

.

CSP	Generation / Input	Output	Storage	Zeroization	Use
TLS DH Private Parameter	Internally, using the FIPS-Approved ANSI X9.31 DRNG.	N/A. Never output beyond the physical boundary of the GPC by the module.	Volatile memory only.	N/A. Not persistently stored, but zeroized upon session termination.	Used to establish the TLS Pre-Master Secret
TLS DH Public Parameter	Internally, using the FIPS-Approved ANSI X9.31 DRNG. If not internally generated, then it is provided by the calling application as the TLS peer's TLS DH Public Parameter.	N/A. Never output beyond the physical boundary of the GPC by the module. The calling application is responsible for providing this value to the TLS peer during the TLS handshake.	Volatile memory only.	N/A. Not persistently stored, but zeroized upon session termination.	Used by the peer to establish the TLS Pre-Master Secret when Diffie-Hellman is the selected cipher suite. If received, then it is the TLS peer's TLS DH Public Parameter, which is used internally to establish the TLS Pre-Master Secret.
TLS RSA Private Key	Imported.	N/A. Never output beyond the physical boundary of the GPC by the module.	Volatile memory only.	N/A. Not persistently stored, but zeroized upon session termination.	Depending on the cipher suite selected, it may be used to either create digital signatures or to decrypt the TLS Pre-Master Secret if the module is used to implement the TLS Server.

.

CSP	Generation / Input	Output	Storage	Zeroization	Use
TLS RSA Public Key	Imported.	N/A. Never output beyond the physical boundary of the GPC by the module. The calling application or operator is responsible for ensuring the TLS peer receives this public key.	Volatile memory only.	N/A. Not persistently stored, but zeroized upon session termination.	Depending on the cipher suite selected, it may be used by the TLS peer to verify digital signatures generated by the module or it may be used by the TLS peer to encrypt the TLS Pre-Master Secret if the module is used to implement the TLS Server.
TLS Peer RSA Public Key	Imported.	N/A. Never output beyond the physical boundary of the GPC by the module.	Volatile memory only.	N/A. Not persistently stored, but zeroized upon session termination.	Depending on the cipher suite selected, it may be used to verify digital signatures generated by the TLS Peer or it may be used to encrypt the TLS Pre-Master Secret if the module is used to implement the TLS Client.
TLS DSA Private Key	Imported.	N/A. Never output beyond the physical boundary of the GPC by the module.	Volatile memory only.	N/A. Not persistently stored, but zeroized upon session termination.	Used to create digital signatures during the TLS session.

.

CSP	Generation / Input	Output	Storage	Zeroization	Use
TLS DSA Public Key	Imported.	N/A. Never output beyond the physical boundary of the GPC by the module. The calling application or operator is responsible for ensuring the TLS peer receives this public key.	Volatile memory only.	N/A. Not persistently stored, but zeroized upon session termination.	Used by the TLS peer to verify digital signatures created by the module.
TLS DSA Peer Public Key	Imported.	N/A. Never output beyond the physical boundary of the GPC by the module.	Volatile memory only.	N/A. Not persistently stored, but zeroized upon session termination.	Used to verify digital signatures created by the TLS peer.
DRNG Seed Value	Imported from dev/urandom.	N/A.	Volatile memory only.	N/A. Not persistently stored.	Used to generate random values.
DRNG Seed Key	Imported from dev/urandom.	N/A.	Volatile memory only.	N/A. Not persistently stored.	Used to generate random values.
Software Integrity Key (HMAC)	N/A.	N/A.	Embedded within the library.	N/A.	Used to perform the software integrity test at power-on.

.

2.8 EMI/EMC

The test platform used to perform operational testing complies with the EMI/EMC requirements of the FIPS 140-2 standard.

2.9 Self-Tests

This section describes the power-up and conditional self-tests performed by the module.

2.9.1 Power-Up Self-Tests

The CryptoCore Module performs the following self-tests at power-up when the FIPS-Approved mode of operation has been invoked:

- Software integrity checks (HMAC SHA-256) over each component
- Known Answer Tests (KATs)
  - AES KAT
  - Triple-DES KAT
  - RSA KAT
  - DSA Sign/Verify Pairwise Consistency Test
  - HMAC SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 KATs
  - ANSI X9.31 DRNG KAT

2.9.2 Conditional Self-Tests

The CryptoCore Module performs the following conditional self-tests:

- Continuous DRNG Test
- RSA Pairwise Consistency Check

2.10 Design Assurance

Source code and documentation are both managed and stored within Perforce, an automated configuration management system, and its associated server.

2.11 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 CryptoCore 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.

The Crypto-Officer role is responsible for installing the module as part of its host application. The cryptographic module is installed and can operate in a FIPS-Approved or non-FIPS-Approved mode of operation. The cryptographic module implements a software integrity test that consists of an HMAC computed over each of the libraries. During the power-up self-tests phase, the signatures are verified over the stored CryptoCore Module instance. If the stored signatures are verified, then the test is passed. Otherwise, the test is failed and the module enters an error state where no cryptographic functionality is allowed.

3.1 Initial Setup

The module components must be installed within the same directory.

3.2 Secure Management

This section provides guidance which ensures that the module is always operated in a secure configuration.

3.2.1 Initialization

It is the Crypto-Officer's responsibility to configure the module into the FIPS-Approved mode. In order to invoke the FIPS-Approved mode of operation, the Crypto-Officer must perform the following:

1. The dynamic libraries and HMAC signature files must be installed in the same directory.

2. The calling application must invoke "CryptoGlobal::setFIPSMode".

3. The crypto.cfg configuration file must have "gFIPSTag" set to a non-zero value or the "FIPS" value set to 1 before loading the module.

4. The operator must invoke "CryptoGlobal::gCryptoGlobal->isFIPSMode" to determine whether FIPS-Approved mode has been successfully enabled.

FIPS 140-2 mandates that a software cryptographic module at Security Level 1 shall be restricted to a single operator mode of operation. Prior to installing the module, the Crypto-Officer must ensure the server running RHEL v5.1 or RHEL v5.5 is in single-user mode.

3.2.2 Management

The Crypto-Officer should monitor the module's status regularly and make sure only the services listed in Table 3 and Table 4 are being used. If any irregular activity is noticed or the module is consistently reporting errors, then SenSage customer support should be contacted.

3.2.3 Zeroization

All persistently stored keys are encrypted within the SKV or are encrypted and then stored within the header of an encrypted file. As a result, zeroization is not required. If the operator desires to destroy the

encrypted keys, despite their protected storage, the operator may erase the SKV or the encrypted files containing the keys.

Zeroization of temporary keys normally occurs during the termination of a secure network session. The Cryptographic Officer has the option to zeroize the temporary keys on-demand by performing a system reboot or by unloading the module from memory. Please refer to Table 3 for the list of temporary keys that can be zeroized by this method.

3.3 User Guidance

The User is responsible for ensuring the FIPS-Approved mode has been assumed prior to invoking any cryptographic functions. This may be done by verifying a call to "*CryptoGlobal::gCryptoGlobal->isFIPSMode" returns TRUE*. The user must only invoke services through the libCryptoCore or libSSL libraries.

In addition, it is the User's responsibility to select a passphrase that is sufficiently strong. The User should report to the Crypto-Officer if any irregular activity is noticed.

4 Acronyms

This section describes the acronyms.

Table 7 – Acronyms

Acronym	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
ANSI	American National Standards Institute
ATM	Automated Teller Machine
BIOS	Basic Input/Output System
CAST	Carlisle Adamas and Stafford, alternatively called CAST-128
CBC	Cipher Block Chaining
CDR	Call Detail Record
CFB	Cipher Feedback
CLI	Command Line Interface
CMVP	Cryptographic Module Validation Program
CPU	Central Processing Unit
CSEC	Communications Security Establishment Canada
CSP	Critical Security Parameter
DES	Data Encryption Standard
DRNG	Deterministic Random Number Generator
DSA	Digital Signature Algorithm
DVD	Digital Video Disc
ECB	Electronic Codebook
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FIPS	Federal Information Processing Standard
HDD	Hard Disk Drive
HMAC	(Keyed-) Hash Message Authentication Code
IPDR	Internet Protocol Detail Records
KAT	Known Answer Test
LAN	Local Area Network
MD5	Message-Digest algorithm 5
N/A	Not Applicable
NIST	National Institute of Standards and Technology

.

Acronym	Definition
OFB	Output Feedback
OS	Operating System
PCI	Peripheral Component Interconnect
PCIe	Peripheral Component Interconnect Express
PEFS	Private Encryption File System
PKCS	Public-Key Cryptography Standards
PRNG	Pseudo Random Number Generator
PSS	Probabilistic Signature Scheme
RAM	Random Access Memory
RHEL	Red Hat Enterprise Linux
RNG	Random Number Generator
RSA	Rivest Shamir and Adleman
SATA	Serial Advanced Technology Attachment
SCSI	Small Computer System Interface
SQL	Structured Query Language
SHA	Secure Hash Algorithm
SKV	Secure Key Vault
SLS	Scalable Log Server
TDES	Triple Data Encryption Standard
TLS	Transport Layer Security
USB	Universal Serial Bus
WAN	Wide Area Network

Prepared by:
Corsec Security, Inc.

13135 Lee Jackson Memorial Highway, Suite 220
Fairfax, VA 22033
United States of America

Phone: +1 (703) 267-6050
Email: info@corsec.com
http://www.corsec.com