Harris Corporation

Cryptographic Library (SECLIB)

Non-Proprietary Security Policy

R12

This document is published by Harris Corporation without any warranty. Improvements and changes to this document necessitated by typographical errors, inaccuracies of current information, or improvements to programs and/or equipment, may be made by Harris Corporation at any time and without notice.

Revision History

Revision	Date	Author	Changes
-	29-Jun-05	T. Hengeveld	Initial Draft for Atlan Labs Evaluation
R1	Feb-06	T. Hengeveld	Internal.
R2	March-06	T. Hengeveld	Revision corresponding to SECLIB API R2
R3	Sept-06	T. Hengeveld	Revision corresponding to SECLIB API R3
R4	March-07	A.Gandreti	Revisions to remove reference to static library and other updates for submission to Atlan Labs.
R5	July-07	A.Gandreti	Addressed Atlan’s comments
R6	Aug-07	A. Gandreti	Changed references from TDES MAC and TDES Key wrap to DESMAC and DES key wrap. Addressed Atlan’s comments.
R7	Sept-07	S. Shorter	Addressed another set of comments.
R8	Sept-07	S. Shorter	Addressed another set of comments.
R9	Jan-08	S. Shorter	Fixed Table of Contents
R10	Apr-10	D. Warheit	Rebranded to Harris Corporation. OS Versions updated. Corrected minor typographical errors.
R11	Jan-15	S. Bertrand	OS/Platforms validated as of Jan-15 included.
R12	Mar-16	S. Bertrand	Updates pursuant to RNG transition outlined in SP800-131A (see CMVP notice dated 11-24-2015)

TABLE OF CONTENTS

Page

1 Introduction

This is a non-proprietary security policy for the Harris Corporation Cryptographic Library. It describes how the module meets the security requirements of FIPS 140-2 (Ref. [1]) and how to securely operate the module in a FIPS compliant manner. This policy was prepared as part of the level 1 FIPS 140-2 validation of the cryptographic module.

1.1 References

[1] Security Requirements for Cryptographic Modules, FIPS 140-2, Information Technology Laboratory, NIST, Gaithersburg MD, May 25, 2001.

[2] *Advanced Encryption Standard*, Federal Information Processing Standards Publication 197 (FIPS PUB 197) November 26, 2001..

[3] *Data Encryption Standard*, Federal Information Processing Standards Publication 46-3 (Withdrawn) (FIPS PUB 46-3) October 25, 1999.

[4] *Secure Hash Standard*, Federal Information Processing Standards Publication 180-4 (FIPS PUB 180-4, http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf), August 2015.

[5] *TIA/EIA-102.AAAD*, Project 25 Block Encryption Protocol, Telecommunications Industry Association, July 2002.

[6] *Recommendation for Block Cipher Modes of Operation, Methods and Techniques*, Morris Dworkin, National Institute of Standards and Technology, NIST Special Publication 800-38A, 2001 Edition.

[7] *HMAC: Keyed Hashing for Message Authentication*, Internet RFC 2104, Krawczyk, et. al., February 1997.

[8] *The Keyed-Hash Message Authentication Code (HMAC)*, Federal Information Processing Standards Publication 198-1 (FIPS PUB 198-1, http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf), July 2008.

[9] *AES Key Wrap Specification*, 16-November-2001.

[10] *NIST-Recommended Random Number Generator Based on ANSI X9.31 Appendix A.2.4 Using 3-Key Triple DES and AES Algorithms*, Sharon S. Keller, NIST/ITL/CSD, January 2005.

1.2 Abbreviations

AES	Advanced Encryption Standard (Ref. [2])
API	Application Programmer’s Interface
CBC	Cipher Block Chaining Mode
CO	Crypto-Officer
CRNG	Continuous Random Number Generator (Test)
CSP	Cryptographic Security Parameter
CTR	Counter Mode
DES	Data Encryption Standard (Ref. [3])
DLL	Dynamic Link Library
ECB	Electronic Code Book Mode
HMAC-SHA1	Hash Message Authentication Code based on SHA1 (Ref. [8])
HMAC-SHA256	HMAC employing SHA-256
KAT	Known Answer Test
MAC	Message Authentication Code
OFB	Output Feedback Mode
OS	Operating System
P25	APCO Project 25
RNG	Random Number Generator
SECDLL	Security Library DLL
SECLIB	Security Library – The subject cryptographic module.
SHA1	Secure Hash Standard (Ref. [4])
SHA256	Secure Hash Standard (Ref. [4]) with 256-bit result.

2 Cryptographic Module Overview

2.1 Operating Environment & Module Interfaces

The SECLIB version R1A is a software cryptographic library and is validated as a multi-chip, standalone module. It runs in the operational environment of a standard Intel-based computer running the Windows or Windows Server operating system. The cryptographic module boundary is the case of the computer, containing the integrated circuits of the motherboard, the CPU, random access memory, keyboard, mouse, video interfaces, hard drive, and other hardware components. The module is packaged as a dynamically loaded library (DLL) which contains all of the module's executable code. The SECLIB DLL module performs all of its own FIPS required operations.

The Harris Corporation security library was tested on the following platforms:

- Microsoft Windows XP Professional Service Pack 2
- Microsoft Windows Server 2003 Service Pack 2
- Microsoft Windows 7 Enterprise Service Pack 1
- Microsoft Windows 8.1 Pro
- Android Kit Kat 4.4.2

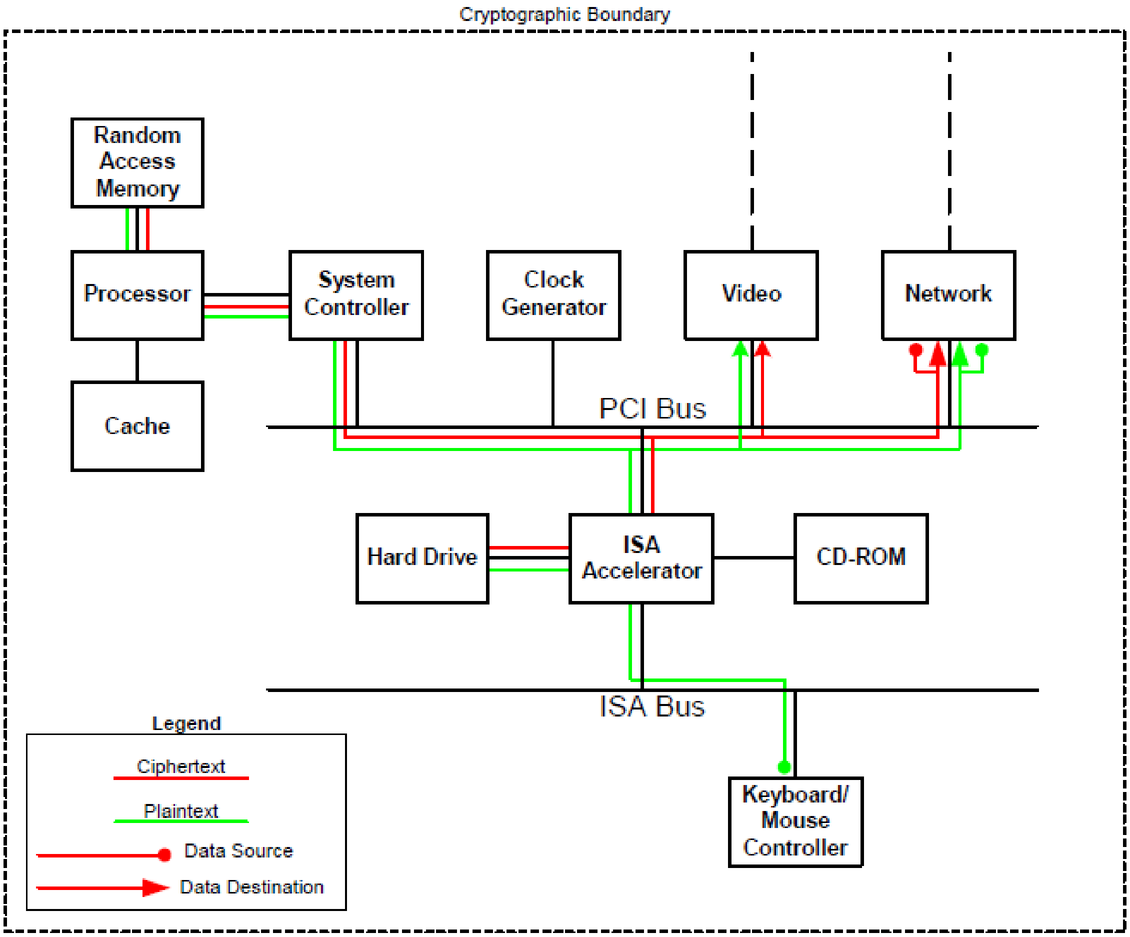

Figure 1: Generic Computer Hardware Functional Block Diagram

Figure 1 shows the functional block diagram for the computer on which the SECLIB module runs. All components shown in the diagram are within the physical cryptographic boundary of the module, and the diagram shows interconnections among the major components of the module. Dashed lines represent connections to equipment or components outside the cryptographic boundary.

Software is stored on the hard drive of the system, and loaded into random access memory for execution.

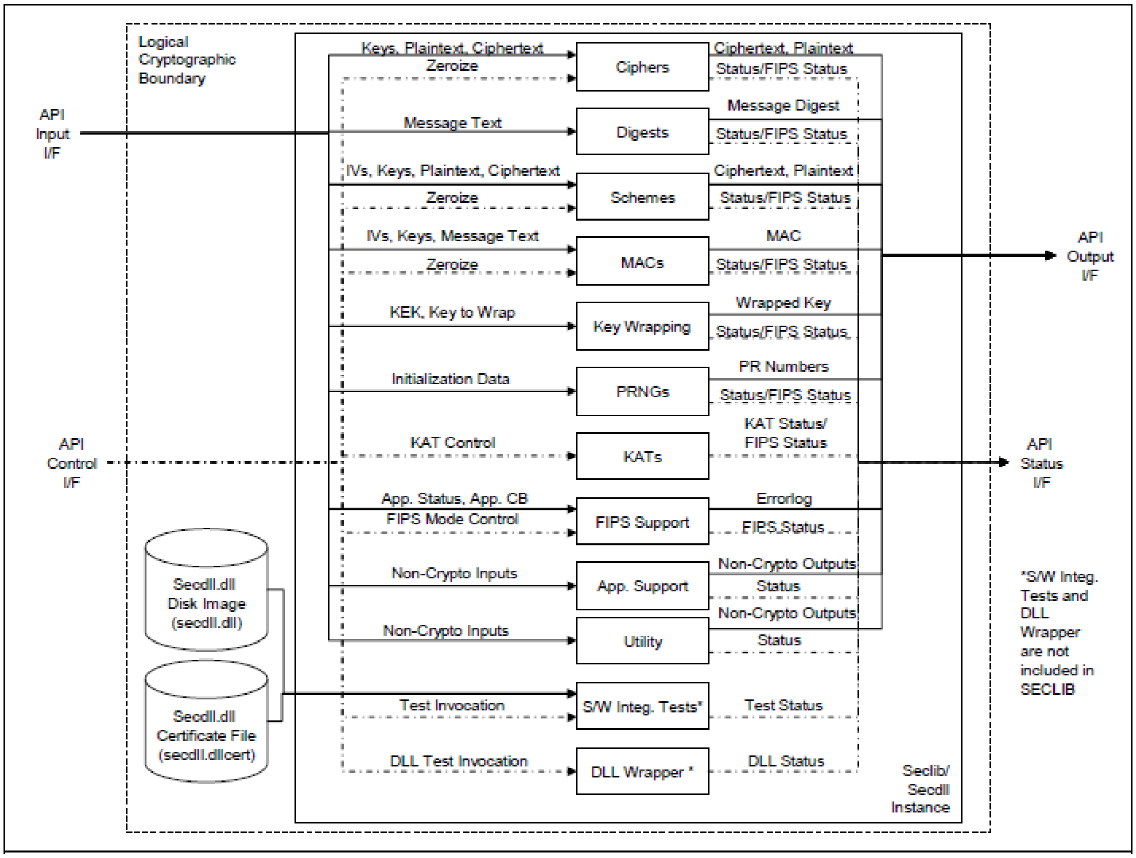

Figure 2: Diagram of the Module

Figure 2 illustrates the cryptographic module. As provided, the module consists of two disk images comprising an image of the SECLIB Dynamic Link Library, and a certificate file. Upon invocation of the DLL, the certificate file is employed by the DLL to perform a software integrity test. The DLL is then able to provide nine cryptographic service classes and two non-cryptographic service classes, as summarized in Table 1.

2.1.1 Module Validation Level

The module is intended to meet requirements of FIPS 140-2 security level 1 overall. The following table shows the security level for each of the eleven requirement areas of the validation.

FIPS 140-2 Security Requirement Area	Security Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles Services and Authentication	1
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	N/A

Table 1: Module Validation Level

2.1.2 Physical Interfaces

The physical interfaces are those of a standard Intel-based computer system, including the computer keyboard and mouse, network ports, CD-ROM drive, video monitor port, and power plug. All port connectors used in the module are standard. The system has a serial port which is not used.

2.1.3 Logical Interfaces

The logical interface to the module is the Application Programming Interface (API) of the software library. The module sends and receives data entirely through the API. The module provides for control input through the API calls. Data Input and Output are provided in variables passed with API calls, and Status Output is provided through function return values, exception function callbacks, and error codes that are documented for each provided service.

Each service class consists of a collection of API calls providing particular cryptographic or non-cryptographic functions.

The operating system controls separation of these logical interfaces when the module communicates over the same physical interfaces. Logical interfaces are separated by the structure of the API and the definition of the interfaces. Each input is directed to a particular API call and each output returned from a particular API call.

The operating environment obtains data from various sources, including network and keyboard interfaces, and prepares that data to become input to the software module. The data might be stored on the hard disk before being used as input data.

Service Class	Crypto	Description
Ciphers	Yes	Primitive functions implementing the AES, DES[1] and Triple-DES[2] ciphers.
Digests	Yes	Primitive functions providing the SHA1 and SHA-256 message digests.
Schemes	Yes	Functions providing encryption schemes (ECB, CTR, CBC, etc) based on the ciphers of the Cipher service class.
MACs	Yes	Functions providing message authentication codes (HMAC[3], CBC-MAC) based on the primitive Digest and Cipher service classes.
Key Wrap	Yes	Functions providing key wrap functions based on the Cipher service class.
RNGs	Yes	Functions providing two non-FIPS approved random number generators.
KATs	Yes	Functions providing known answer tests.
FIPS Support	Yes	Functions providing access to the FIPS status of the module.
S/W Integrity Tests	Yes	Functions providing the software integrity test for the module.
Application Support	No	Functions providing checksums and other arithmetic capabilities that are non-cryptographic.
Utility	No	Various non-cryptographic utility functions.

Table 2: Service Class Summary

[1] DES is used only in the Non-FIPS mode of operation.

[2] 2-key Triple-DES can only be used in the non-FIPS mode of operation.

[3] HMAC with key sizes less than 112 bits can only be used in the non-FIPS mode of operation.

Type	Logical Interface	Information/Purpose
Data Input	IVs	Initial Vectors for cryptographic schemes and MACs. Initial vectors are not CSPs.
	Keys	Plain-text keys provided to the module for encryption and authentication functions. Keys are CSPs.
	Plaintext	Plaintext to be encrypted.
	Ciphertext	Encrypted plaintext.
	Message Text	Data for which a MAC or message digest is to be computed.
	KEK	A Key encryption Key used to encrypt keys using keywrapping. KEKs are CSPs.
	Key to Wrap	A key to be wrapped using keywrapping.
	Initialization Data	Random initialization data used to initialize a RNG. Initialization data is a CSP.
	App. Status	The FIPS status of the application using SECLIB.
	App. CB (callback)	A point to a function that is notified of changes in the FIPS status of SECLIB.
	Non-Crypto Inputs	Various information of a non-cryptographic nature supplied as inputs to support and utility functions.
Control Input	Zeroize	A command to zeroize CSPs present in state variables.
	KAT Control	Function invocations for Known Answer tests.
	FIPS Mode Control	Function invocations and data commanding a particular FIPS operating mode.
	Test Invocation	Invocations of Software Integrity tests.
Data Output	Ciphertext	Encrypted plaintext.
	Plaintext	Decrypted ciphertext.
	Message Digest	The result of the computation of a message digest (such as SHA1) on message text.
	MAC	The result of the computation of a message authentication code (such as HMAC-SHA1) on message text.
	Wrapped Key	A key wrapped in accordance with ref. [9].
	PR Numbers	Pseudo-random numbers
	Errorlog	A recording of error conditions encountered by the module, reportable to an invoking application.
	Non-Crypto Outputs	Various non-cryptographic outputs of support and utility functions.
Status Output	Status/FIPS Status	FIP status indicators and function status indicators of cryptographic functions.
	KAT Status	Success or failure indicator of Known Answer Tests.
	FIPS Status	FIPS Status
	Status (Non-Crypto)	Non-cryptographic function status indicators.
	Test Status	The result of a software integrity test.
Power	None	N/A

Table 3: Interface Summary

2.2 Cryptographic Algorithms

The following table lists the cryptographic algorithms supported by SECLIB DLL:

Algorithm	Validation Certificate	Applicability
AES (ECB, CBC, OFB, CTR)	#637	FIPS and Non-FIPS modes
AES-MAC (vendor affirmed, non-compliant)	N/A	Non-FIPS mode
DES (ECB, CBC, OFB, CTR)	N/A	Non-FIPS mode
DES-MAC	N/A	Non-FIPS mode
Triple-DES (ECB, CBC, OFB, CTR)	#591	FIPS and Non-FIPS modes
SHA-1	#673	FIPS and Non-FIPS modes
HMAC-SHA1	#328	FIPS and Non-FIPS modes
SHA-256	#673	FIPS and Non-FIPS modes
HMAC-SHA256	#328	FIPS and Non-FIPS modes
Non-approved RNG (ANSI X9.31 (Ref.[10])	N/A	Non-approved. Allowed in FIPS mode for initial vector (IV) generation. Available for usage in Non-FIPS mode
NDRNG (SHA1 Based)	N/A	Non-approved. Allowed in FIPS mode for initial vector (IV) generation. Available for usage in Non-FIPS mode
AES Key Wrap (Ref. [9])	N/A	FIPS and Non-FIPS modes
DES Key Wrap	N/A	Non-FIPS modes

Table 4: Cryptographic Algorithms

2.3 Self-Tests

SECDLL automatically performs power-up and conditional self-tests on instantiation to ensure proper operation in the FIPS-140-2 compliant mode. Until the power-up self-tests are completed, no data can be processed by the module. Thus data output and input are inhibited during self testing. If self testing fails, the module will enter an error state and the module instantiation fails. When this occurs, any function calls to the SECLIB module will result in an error and thus data output will not be possible.

2.3.1 Power-up Self Tests

The following sections describe the power-up self-tests of the cryptographic module.

2.3.1.1 Software Integrity Test

SECLIB is delivered from the manufacturer in the form of a windows DLL file with an accompanying certificate file. Upon process instantiation, SECDLL performs a software integrity test as follows:

1. Upon instantiation, the operating system provides the DLL initialization routine with a “handle” that is used by SECLIB to fetch the full pathname of the DLL image.
2. SECLIB assumes that the certificate file resides on the same directory as the DLL image and has the same filename concatenated with the string “cert”.
3. SECLIB opens the certificate, retrieving the encrypted authentication key stored therein.
4. SECLIB decrypts the authentication key using a hard-coded key encryption key.
5. SECLIB then applies the HMAC-SHA1 message authentication code to the DLL image and compares it to the result stored in the certificate.
6. If the two results fail to match, the module enters the error state. Otherwise, it proceeds with the Known Answer Tests described below.

Any attempt to enter the approved FIPS mode following a failure of the software integrity test causes a transition into the FIPS_FAILED state.

2.3.1.2 Known Answer Tests

Upon process instantiation and upon entry into the FIPS approved mode, SECLIB performs Known Answer Tests (KATs) on the following cryptographic services:

- AES in ECB, CTR, CBC, and OFB modes with 128, 192 and 256 bit keys for both encryption and decryption
- Triple-DES in ECB, CTR, CBC and OFB modes for both encryption and decryption
- SHA-1
- HMAC-SHA-1
- SHA-256

- HMAC-SHA-256
- AES Key wrap KAT

KATs are also made available at the API of SECLIB module to allow the user to confirm the validity of the cryptographic algorithms at any time.

If the module is in the FIPS approved mode, then failure of any KAT produces a transition to the FIPS_FAILED state.

2.3.2 Conditional Self-Tests

SECLIB module performs the CRNG conditional self-test. Each time either of its random number generators is used to produce pseudo-random data, the module performs the continuous random number generator test. If the module is operating in the FIPS approved mode, then failure of the test produces a transition into the FIPS_FAILED state, followed by transition to the FIPS state after reseeding. In addition, the RNGs require seeding before being used to generate pseudo-random numbers. Failure of the user to seed the RNG prior to operation likewise causes a transition to the FIPS_FAILED state.

2.3.3 Critical Function Tests

SECLIB has no additional critical functions to test.

3 Security Policy

3.1 Identification and Authentication Policy

As allowed by FIPS 140-2 for Level 1 certification, SECLIB does not support user identification or authentication for any of its identified roles.

3.2 Access Control Policy

3.2.1 Supported Roles

SECLIB supports two roles, the Crypto-User (User) role and the Crypto-Officer (CO) role. These roles are implicitly assumed by the calling application. SECLIB does not support multithreading, so only one role may be active at a time.

An operator assuming the CO role can call any of the module's functions. An operator assuming the User role can call any of the module's functions except those identified in Section 3.2.2 as pertaining only to the CO.

Role	Type of Authentication	Authentication Data
Crypto-User (User)	None	None
Crypto-Officer (CO)	None	None

Table 5: Roles and Required Identification and Authentication

3.2.2 Complete List of Services

The table below provides a complete list of services provided by the DLL and identifies applicable roles.

Role	**Authorized Services**	**Key/CSP Access**	**Type(s) of Access**[4]
User and Crypto-Officer	Cipher Service Class	AES Key, Triple-DES Key	R,W
	Digest Service Class	None	
	Scheme Service Class	AES Key, Triple-DES Key	R,W
	MAC Service Class	HMAC Key, AES Key	R,W
	Key Wrap Service Class	AES Key	R,W
	RNG Service Class	Seed, Seed Key, AES Key	R,W
	FIPS Support Service Class (Including the Show Status Service)	None	
	Application Support Service Class	None	
	Utility Service Class	None	

[4] R=read, W=write

Crypto-Officer Only	Software Integrity Test Service Class (Self-test Service)	HMAC Integrity Key, 256-bit embedded key	R,W
	Known Answer Test Service Class (Self-test Service)	None	
	Installation	None	W
	Uninstallation	None	W (delete)

Table 6: Roles and Services

3.2.3 Cryptographic Security Parameters and Keys

SECDLL contains an embedded 256-bit plaintext key used to decrypt the authentication key used for the power-up software integrity test described in Section 2.3.1.1. All other keys employed by SECLIB are provided by the user through API parameters. Each cryptographic service provided by the module provides mechanisms for the zeroization of plaintext keys contained in its supporting data structures. Those services that provide atomic functions for which no key persistence is required zeroize local copies of keys and related material prior to returning to their calling function.

The user of SECLIB has responsibility for calling the appropriate zeroization functions under the following circumstances:

1. If a key containing SECLIB data structure is allocated on the process stack, zeroization functions must be called prior to returning from a user function.
2. If a key containing SECLIB data structure is allocated in static or global memory, zeroization functions must be called prior to de-instantiation of the process.

The following SECLIB data structures contain keys or derived key material that must be zeroized by the user under the above circumstances:

CSP	**Data Structure**	**Zeroization Service**
AES Key	aes_state	aes_key_expand
AES Key	aesmode_state_type	aesmode_keyset
AES Key	aesp25_state_type	aesp25_keyset
Triple-DES Key	des_context	des_keyset
Triple-DES Key	desmode_state_type	desmode_keyset
Triple-DES Key	desp25_state_type	aesp25_keytset
HMAC SHA-1 Key	hmac_sha1_state	hmac_sha1_start or hmac_sha1_finish
HMAC SHA-256 Key	hmac_sha256_state	hmac_sha256_start or
		hmac_sha256_finish
AES Key	aescbcmac_state	aescbcmac_start or aescbcmac_finish

Table 7: Data Structures Requiring User Zeroization

3.3 Physical Security Policy

Physical security is provided by the host PC on which SECLIB executes.

3.4 Operational Environment

The module's operational environment is described above, and consists of a commercially available general-purpose hardware computing platform and Windows Operating system configured for use in single-user mode.

While cryptographic processing is in use, keys and CSPs are protected by process separation. When the module starts up, it performs an integrity self-check using the HMAC-SHA1 algorithm.

3.5 Mitigation of Other Attacks

SECLIB provides no special mitigation against other attacks.

4 Secure Operation of the SECLIB Cryptographic Module

Operation of SECLIB in the secure mode is subject to the following rules and policies.

1. A user invokes the secure mode through a call to the secdll_enable_fips service. This service requires that the user previously provide a notification callback (via the secdll_install_fips_callback service) that allows SECDLL to inform the user of FIPS errors as they occur.
2. The user is responsible for using the zeroization capabilities for each data structure upon completion of the use of that structure, as specified Table 7 above. The user is also responsible for providing keys that meet the FIPS-140-2 criteria.
3. The non-approved RNG and NDRNG may be used in the secure mode, but only as specified in FIPS140-2 section 4.7.1, and for other non-cryptographic purposes.
4. A user must use only the cryptographic algorithms labeled as FIPS mode in Table 3.
5. Related to SP800-131A transitions in Table 1 of http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf, 2-key Triple-DES can only be used in the non-FIPS mode of operation.
6. Related to SP800-131A transitions in Table 10 of http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf, HMAC with key sizes less than 112 bits can only be used in the non-FIPS mode of operation.

5 Services Provided

Table 8: Services Provided

Service Class	Subclass	API Function	Purpose	Keys/CSPs	Role[5]	Applicability[6]
Ciphers	AES Encryption/Decryption	aes_key_expand	Set AES Key, Zeroize AES Key	AES Key	B	F
		aes_encrypt	Execute AES Cipher		B	F
		aes_decrypt	Execute AES Inverse Cipher		B	F
	Triple-DES Encryption/Decryption	des_keyset	Set Triple-DES Key, Zeroize Key	Triple-DES key	B	F
		des_encrypt	Execute Triple-DES Cipher		B	F
		des_decrypt	Execute Triple-DES Inverse Cipher		B	F
Digests	SHA-1 digest Computation	sha1_init	Initialize hash	None	B	F
		sha1_update	Hash additional message text		B	F
		sha1_finish	Finish hash		B	F
		sha1_hash	Hash a complete message		B	F
	SHA-256 digest computation	sha256_init	Initialize hash	None	B	F
		sha256_update	Hash additional message text		B	F
		sha256_finish	Finish hash		B	F
		sha256_hash	Hash a complete message		B	F
Schemes	AES mode (scheme) encryption/decryption	aesmode_set_mode	Choose AES mode (CBC, ECB, etc)	AES key	B	F
		aesmode_link_aes	Link cipher to mode		B	F
		aesmode_reset	Reset encryption mode		B	F
		aesmode_keyset	Set Key for mode, Zeroize Key		B	F
		aesmode_set_iv	Set Initial Vector for AES Mode		B	F
		aesmode_encrypt	Encrypt in selected AES mode		B	F
		aesmode_decrypt	Decrypt in selected AES mode		B	F
		aesp25_do_data_encryption	Select between voice and data encryption formats for P25 encryption		B	F
		aesp25_link_aes	Link cipher to mode		B	F
		aesp25_reset	Reset encryption mode		B	F

[5] U = User Only, CO = Crypto-Officer Only, B = Both, N/A = Not Available
[6] F = FIPS/Non-FIPS: Disabled on FIPS Error, N = Non-FIPS Only, NC = Non-Crypto: Always Available, N/A = Not Available

Service Class	Subclass	API Function	Purpose	Keys/CSPs	Role[5]	Applicability[6]
		aesp25_keyset	Set Key for mode, Zeroize Key		B	F
		aesp25_set_mi	Set P25 Message Indicator (IV)		B	F
		aesp25_get_mi	Get P25 Message Indicator Value		B	F
		aesp25_encrypt	Encrypt in selected P25 mode		B	F
		aesp25_decrypt	Decrypt in selected P25 mode		B	F
	Triple-DES mode (scheme) encryption/decryption	desmode_set_mode	Choose Triple-DES mode (CBC, ECB, etc)	Triple-DES Key	B	F
		desmode_link_des	Link cipher to mode		B	F
		desmode_reset	Reset encryption mode		B	F
		desmode_keyset	Set Triple-DES Key for mode, Zeroize Key		B	F
		desmode_set_iv	Set Initial Vector for Mode		B	F
		desmode_encrypt	Triple-DES Encrypt in selected mode		B	F
		desmode_decrypt	Triple-DES Decrypt in selected mode		B	F
		desp25_do_data_encryption	Select between voice and data encryption formats for P25 encryption		B	F
		desp25_link_des	Link cipher to mode		B	F
		desp25_reset	Reset encryption mode		B	F
		desp25_keyset	Set Key for mode, Zeroize Key		B	F
		desp25_set_mi	Set P25 Message Indicator (IV)		B	F
		desp25_get_mi	Get P25 Message Indicator Value		B	F
		desp25_encrypt	Triple-DES Encrypt in selected P25 mode		B	F
		desp25_decrypt	Triple-DES Decrypt in selected P25 mode		B	F
Key Wrap	AES key wrap [10]	aes_keywrap	Wrap a Key	AES key	B	F
		aes_keyunwrap	Unwrap a key		B	F
	DES key wrap	des_keywrap	Wrap a Key using DES		B	N
		des_keyunwrap	Unwrap a key using DES		B	N
MACs	HMAC-SHA-1 computation	hmac_sha1_start	Set key and initialize HMAC-SHA1 authentication code	HMAC Key	B	F

Service Class	Subclass	API Function	Purpose	Keys/CSPs	Role[5]	Applicability[6]
			computation			
		hmac_sha1_update	Include additional message text in the MAC		B	F
		hmac_sha1_finish	Complete the computation of the MAC		B	F
		hmac_sha1_authenticate	Compare/Validate HMAC		B	F
	HMAC-SHA-256 computation	hmac_sha256_start	Set key and initialize HMAC-SHA256 authentication code computation	HMAC Key	B	F
		hmac_sha256_update	Include additional message text in the MAC		B	F
		hmac_sha256_finish	Complete the computation of the MAC		B	F
		hmac_sha256_authenticate	Compare/Validate HMAC		B	F
	AES CBC MAC computation	aescbcmac_start	Set key and initialize AESCBC MAC computation	AES Key	B	N
		aescbcmac_update	Include additional message text in the MAC		B	N
		aescbcmac_finish	Complete the computation of the MAC		B	N
		aescbcmac_authenticate	Compare/validate CBC MAC		B	N
	DES CBC MAC computation	descbcmac_start	Set key and initialize DESCBC MAC computation	DES Key	B	N
		descbcmac_update	Include additional message text in the MAC		B	N
		descbcmac_finish	Complete the computation of the MAC		B	N
		descbcmac_authenticate	Compare/validate CBC MAC		B	N
RNGs	NDRNG Generation	prng_init prng_init_r	Initialize NDRNG	Seed, Seed Key	B	F
		prng_update_seed prng_update_seed_r	Update Seed of NDRNG		B	F
		prng_generate_bytes prng_generate_bytes_r	Generate Random Bytes		B	F
		prng_generate_bytes_fips prng_generate_bytes_rfips	Intermediate NDRNG processing			
	Non-approved RNG	fprng_init fprng_init_r	Initialize non-approved RNG	Seed, Seed	B	F

Service Class	**Subclass**	**API Function**	**Purpose**	**Keys/CSPs**	**Role**[5]	**Applicability**[6]
	Generation	fprng_update_seed fprng_update_seed_r	Update seed of non-approved RNG	Key	B	F
		fprng_generate_bytes fprng_generate_bytes_r	Generate pseudo-random bytes		B	F
		fprng_generate_bytes_fips fprng_generate_bytes_rfips	Intermediate non-approved RNG processing			
Known Answer Tests	AES KAT computation	KAT_aes	AES Known-Answer Test	AES key	CO	F
	SHA-1 KAT computation	KAT_sha1	SHA1 Known-Answer Test	None	CO	F
	HMAC-SHA-1 KAT computation	KAT_hmacsha1	HMAC-SHA1 Known-Answer Test	HMAC Key	CO	F
	SHA-256 KAT computation	KAT_sha256	SHA256 Known-Answer Test	None	CO	F
	HMAC-SHA-256 KAT computation	KAT_hmacsha256	HMAC-SHA256 Known-Answer Test	HMAC Key	CO	F
	AES Key wrap KAT computation	KAT_aeskeywrap	AES Key-wrap Known Answer Test	AES KeyWrap key	CO	F
	Triple-DES KAT	KAT_des	Triple-DES Known Answer Test	Triple-DES key	CO	F
	AES CBC MAC KAT	KAT_aescbcmac	AES-CBCMAC Known Answer Test	AES Key	CO	N
FIPS Support	Other FIPS support functions	DLL_Main	DLL Initialization Function	None	N/A	N/A5[7]
		seclib_enable_fips	Attempt to enter FIPS approved mode of operation		B	F
		seclib_fips_status (Show Status Service)	Return FIPS Status		B	NC
		seclib_user_fips_error	User Initiated FIPS Error		B	NC
		seclib_enumerate_fips_errors	Enumerates the content of a historical FIPS error log		B	NC
		seclib_reset_fips_errorlog	Resets error log content, but not error status	None	B	NC

[7] Invoked by Operation System on instantiation, de-instantiation

Service Class	**Subclass**	**API Function**	**Purpose**	**Keys/CSPs**	**Role**[5]	**Applicability**[6]
		seclib_install_fips_callback	Installs a FIPS notification callback		B	NC
		seclib_cancel_fips_callback	Cancels a previously installed callback		B	NF
Application Support	Application Support functions	otarcs_start	Initialize P25 OTAR Checksum	None	B	NC
		otarcs_update	Include additional message text in OTAR checksum.		B	NC
		otarcs_finish	Complete computation of OTAR checksum		B	NC
		otarcs_authenticate	Compare/Validate OTAR Checksum		B	NC
		p25lfsr_update	Run P25 Linear Feedback Shift Register		B	NC
		p25lfsr_load	Initialize P25 Linear Feedback Shift Register		B	NC
Utility	Other Utility functions	seclib_rval_text	Utility to translate return codes to text	None	B	NC
		seclib_fstat_text	Utility to translate FIPS state to text		B	NC
		seclib_module_text	Utility to translate module identifiers to text		B	NC
		secrval_rval_text	Utility to support the translation codes to text		B	NC
		secutil_8_2_16	Pack unsigned characters into 16-bit words		B	NC
		secutil_16_2_8	Unpack 16-bit words into characters		B	NC
Software Integrity Test	Software Integrity Test functions	secdll_check_certificate		HMAC Integrity Key, embedded 256-bit plaintext key	CO	F
		dllcert_check_certificate	Validates a power-on self-test certificate.		CO	F
		dllcert_rval_text	Translates dllcert error codes into human readable text		CO	NC
		dllcert_print_certtype			CO	

THIS PAGE INTENTIONALLY LEFT BLANK