Cisco Secure Access Control Server (ACS) FIPS Module (cryptolib) Security Policy

FIPS 140-2 Level 1

Version 0.6
May 2012

Table of Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

List of Tables

List of Figures

Overview

The Cisco Secure Access Control Server (ACS) FIPS Module (cryptolib) Versions 1.1, 1.2, and 1.3 referred to herein as "the module" is a software cryptographic library that provides cryptographic services to the Cisco Access Control Server (ACS) application. The module provides FIPS compliant cryptography supporting AAA for IEEE 802.11i security (WPA2) with EAP protocols like EAP-TLS, EAP-FAST, PEAP with RADIUS Key Wrap functionalities, Cisco TrustSec (CTS), and 802.1x-rev.

The module does not implement any of the protocols directly. Instead, it provides the cryptographic primitives and functions to allow a developer to implement various protocols. The software module contains implementations of the following Approved cryptographic algorithms:

- AES Key Wrap
- AES symmetric encryption
- RSA signature generation and verification
- SHA-1 and SHA-2 hash algorithms
- HMAC SHA-1 , HMAC SHA-2
- ANSI X9.31 random number generation

For FIPS 140-2 purposes the module is classified as a multi-chip standalone module. The logical cryptographic boundary of the module is the software library libCryptoLib.so. The physical cryptographic boundary of the module is the enclosure of the computer system on which the module is executing.

Note This document may be copied in its entirety and without modification. All copies must include the copyright notice and statements on the last page.

Security Requirements

Cryptographic Module Specification

The cryptographic module is the “Cisco Secure Access Control Server (ACS) FIPS Module (cryptolib), versions 1.1, 1.2, and 1.3” running in the operational environment of a standard Intel-based computer running Cisco CARS 1.2.0.182, a Linux based operating system. Per section 4.5 of FIPS PUB 140-2, the module is a multiple-chip standalone module. No custom integrated circuits are used by the module.

Figure 1 Cryptographic Boundary Functional Block Diagram

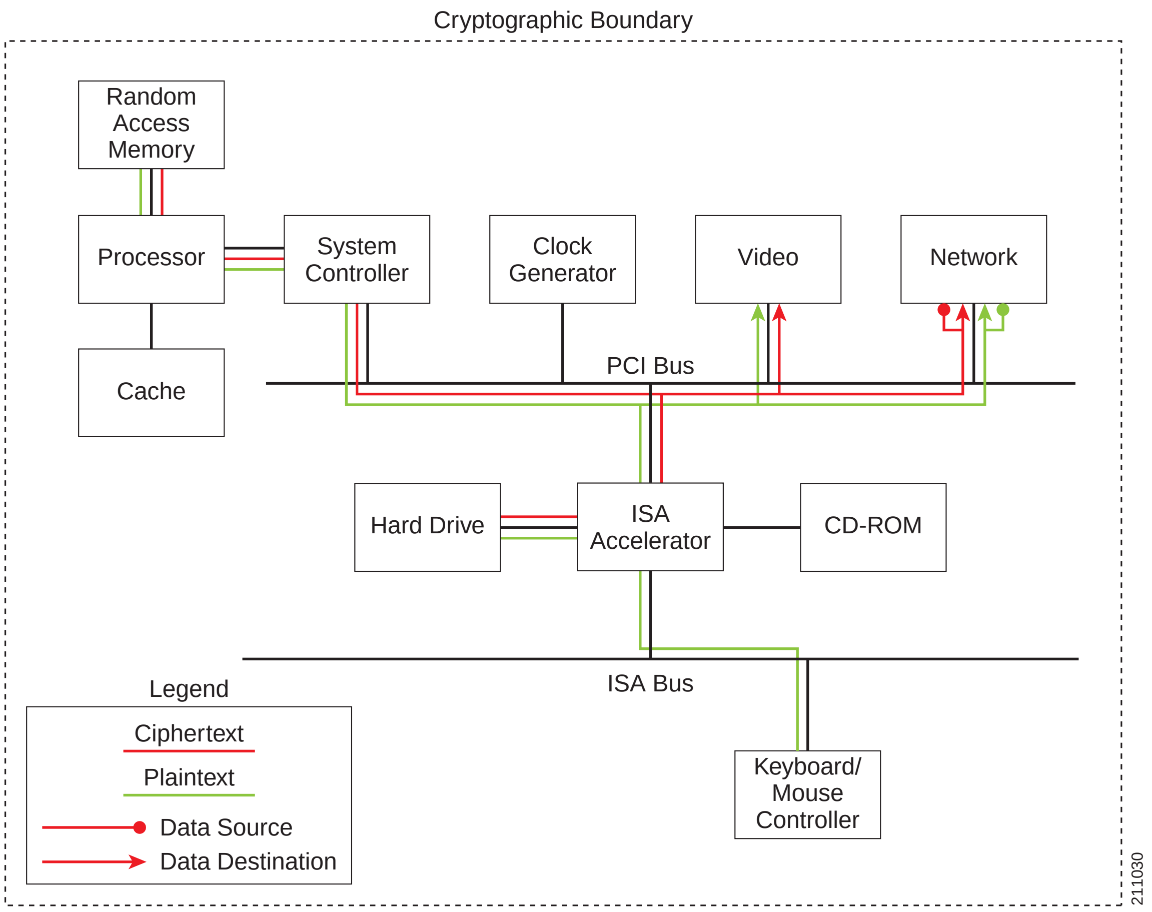

Figure 1 shows the functional block diagram for the computer on which the module runs. All components shown in the diagram are within the physical cryptographic boundary of the module, and the diagram shows interconnections among the major components of the module. Dashed lines represent connections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

The processor component shown in Figure 1 executes all software.

Figure 2 Software Architecture Functional Block Diagram

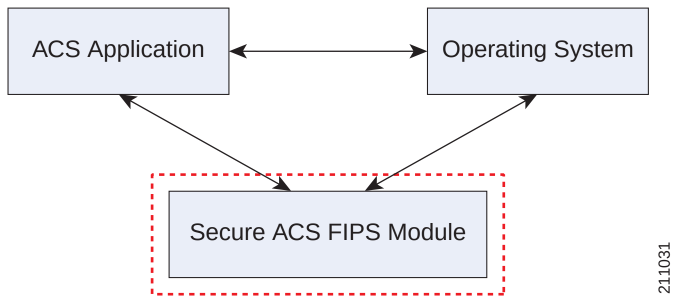

Figure 2 is a functional block diagram that demonstrates how the Cisco Secure ACS FIPS module fits into the overall operational environment. The crypto boundary is represented as a dotted red line which consists of the software library libCryptoLib.so. The functionality of the module is exercised by the Cisco ACS application.

The FIPS mode initialization is performed when the application invokes the CryptoLib software library with CryptoLibrary::Init() function.

When the module is in FIPS mode an operator can only use the FIPS approved or allowed algorithms. The use of non-approved algorithms (i.e. MD4) is not allowed in FIPS mode.

Security Functions

Table 1 lists the cryptographic algorithms implemented by the module.

Table 1 Security Functions

Algorithm	Approved	Algorithm Certificate Number
AES (Encrypt/Decrypt)	Y	1474
RSA (Sign/Verify)	Y	721
SHA-1, SHA-2	Y	1333
HMAC SHA-1, HMAC SHA-2	Y	867
ANSI X9.31 RNG	Y	805
AES Key Wrap	N (allowed in FIPS mode)	—
RSA (Encrypt/Decrypt)	N (1024-2048 bit key size;	—

	allowed in FIPS mode)	
Diffie Hellman	N (2048 bit key size; allowed in FIPS mode)	—
MD5	N	—
MD2	N	—
MD4	N	—
HMAC-MD5	N	—
RC4	N	—
RC2	N	—
DES	N	—

In the FIPS approved mode, the module supports AES key wrap, AES for encryption/decryption, RSA (X.509 certificates) for signature generation/verification, key transport and module integrity test, Diffie-Hellman for key agreement and HMAC-SHA-1 and HMAC-SHA-2 for message authentication.

Cryptographic Module Ports and Interfaces

This section will first detail the physical interfaces to the module, and then the logical interfaces.

Logical Interfaces

The logical interface to the module is the Application Programming Interface (API) of the Cisco Secure ACS FIPS module. Data output is a result of the API function calls, and status output is provided as return values from API function calls.

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. Table 2 describes the logical interfaces.

Table 2 Logical Interfaces

FIPS 140-2 Interface	**Logical Interface**
Data Input	Input parameters of API function calls
Data Output	Output parameters of API function calls
Control Input	API function calls
Status Output	Function calls that return status information and return codes provided by each API function call
Power Interface	None

When the module enters an error state, it no longer will send or receive data. If an error state is encountered, the module will reset and will not send or receive any data until the reset is completed.

The module performs its self tests during the module initialization process. Until the self tests are complete, 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 initialization will fail. When this occurs, any function calls to the module will result in an error, and thus data output will not be possible.

The output data path is physically and logically disconnected from the processes that perform key generation and zeroization. No key information is output through data output interfaces during key generation or zeroization.

Roles, Services & Authentication

The module supports two roles, a User and a Crypto Officer Role. The User Role and Crypto Officer Role are implicitly assumed by the calling application. Calling applications will implicitly choose a role based on the module service requested. Moreover, a calling application will assume only one role in a particular instance in time.

Crypto Officer Role: The Crypto officer can install and initialize the Cisco Secure ACS FIPS module. Other tasks performed by the Crypto Officer include key entry, initiate the power-on self-tests on demand and check the status of the cryptographic module.

User Role: The user role can access the module’s API functions to perform cryptographic operations. This role can also initiate self tests on demand and check the status of the module.

Table 3 lists the roles and the services that they can access.

Table 3 Roles and Services

Role	Services & Access	Interface	Keys & CSPs
Crypto Officer	Module initialization (r, w, x)	CryptoLibrary::Init()	—
	Key Entry (r, w, x)	CryptoLibrary::GetKeyWrapInterface() CryptoLibrary::GetCertManagerInterface()	AES key, RSA private keys, HMAC key
	Show status (r, x)	ICLAuditInterface()	—
	Self-test (x)	CryptoLibrary::GetFipsControlInterfac e()	—
User	AES Key Wrap (r, w, x)	CryptoLibrary::GetKeyWrapInterface()	AES Wrapping Key, AES key, HMAC key, PMK
	Create SSL Context[1]	CryptoLibrary::GetSslInterface()	RSA keys, DH keys, TLS Master secret, AES key, HMAC key
	Cryptographic Primitives	CryptoLibrary::GetCryptoPrimitivesInterface() CryptoLibrary::GetNonFIPSCompliantInterface()	General symmetric and asymmetric keys
	Self-Test (x)	CryptoLibrary::GetFipsControlInterface()	—
	Show Status (r, x)	ICLAuditInterface()	—

Physical Security

Because the module is validated as a FIPS 140-2 level 1 software module and provides no physical security protection mechanisms, this requirement is not applicable.

[1] Used to create primitives for external applications to create TLS sessions used by EAP-FAST, EAP-TLS, PEAP and Cisco TrustSec

Operational Environment

The module's operational environment is described above, and consists of a commercially available general-purpose hardware computing platform and Linux operating system in single-user mode. Both of these operating systems were used for validation testing.

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 by verifying an RSA with SHA-2 digital signature.

Cryptographic Key Management

The module supports AES RADIUS key wrap, AES for encryption/decryption, RSA for authentication and key transport, Diffie Hellman for key agreement and HMAC-SHA-1 and HMAC-SHA-2 for message authentication and module integrity.

Key Generation

All keys that are generated in the module are generated using the ANSI X9.31 FIPS Approved random number generator. The Cisco Secure ACS FIPS Module supports generation of the EAP-FAST PAC key, EAP-FAST Master Key, the random nonce, and the DH key pair.

Key Establishment

The module supports DH key sizes of 2048 bits and RSA key sizes of 1024-2048 bits. These key establishment mechanisms provide sufficient protection of the key being transported. The Diffie-Hellman key establishment method provides 112-bits of encryption strength and the RSA key transport method provides 80 and 112-bits of encryption strength.

The module also uses AES Key Wrap to protect symmetric keys input and output from the module. The AES Key Wrapping Key is a 128-bit AES key. This key wrapping methodology provides 128-bits of encryption strength.

Key Entry/Output

The RSA key pair and the AES Key wrap keys are input into the module in plain-text form by the Crypto Officer using the module API.

Key Storage

The module does not provide persistent storage for the keys used by the algorithms.

Key Zeroization

The keys exist only in volatile memory for the duration of their use and the relevant keys are automatically zeroized after use. All buffers containing key material are zeroized after use before deleting or freeing them.

Table 4 shows the public keys and Table 5 shows secret and private cryptographic keys and CSPs used by the module in the approved mode of operation.

Table 4 Public Keys

Key	Description/Usage	Generation	Storage
TLS RSA public key	RSA key used to perform signature verification Allowed size : 1024 - 2048 bits	Externally provided via module API	Volatile: Stored in plaintext in DRAM

Table 5 Secret & Private Keys

Key	Description/Usage	Generation	Storage	Entry/ Output	Destruction
TLS RSA Private Key	Used for signature generation and symmetric key decryption operations as part of TLS, EAP-TLS, EAP-FAST and PEAP. 1024-2048 bit modulus size.	Externally provided via module API	Outside the module	API	Ephemerally present. Zeroized by power cycling the module
Diffie-Hellman Key pair	Used for DH based key exchange. Note Only authenticated DH key exchange is supported. 2048 bit DH Key	Generated inside the Module using ANSI X9.31 RNG	RAM	API	Ephemerally present. Zeroized by power cycling the module
TLS master secret	TLS master secret as used by TLS, EAP-TLS and EAP FAST. Negotiated during the TLS handshake.	Shared secret derived by asymmetric cryptography (RSA and/or DH)	RAM	N/A	Ephemerally present. Zeroized by power cycling the module
AES key	128-bit AES key used to encrypt session data as part of AES key wrap, Secure RADIUS, TLS, EAP-TLS, EAP-FAST, CTS, 802.11i and 802.1x-REV sessions	Entered via module API	RAM	Entered via API	Ephemerally present. Zeroized by power cycling the module
HMAC key	HMAC-SHA-1 or HMAC-SHA-2 key used for integrity protection as part of RADIUS, Secure RADIUS, TLS, EAP-TLS, EAP-FAST, CTS, 802.11i and 802.1x-REV sessions	Established as part of session set-up	RAM	Entered via API	Ephemerally present. Zeroized by power cycling the module
AES Wrapping Key	128-bit AES key used to wrap and unwrap keys transported from the module	Generated using ANSI X9.31 RNG			
Seed	This is the seed for the X9.31 RNG.	System entropy daemon	RAM	N/A	Zeroized when the system restarts.
Seed Key	This is the seed key for the RNG.	System entropy daemon	RAM	N/A	Zeroized when the system restarts.

Self-Tests

The following self tests occur during module operations. Messages are logged according to the module logging settings — either to the internal event log or to the console. All of these tests are run without inputs or action from an operator.

Power on Self Tests

The power-on self tests consist of a software integrity test, and known answer tests for the cryptographic algorithm implementations.

Software Integrity Test

The Software Integrity check is performed when the Crypto Module is initialized. The module implements an integrity test for the module software by verifying its 1024-bit RSA signature. The software integrity test passes if and only if the signature verifies successfully using the RSA public key.

Cryptographic Algorithm Self-Tests

The module performs the following Self-Tests at Power-on.

Table 6 Known Answer Tests

Algorithm	Test Procedure

AES	encrypt/decrypt KAT
RSA	sign/verify test
RNG	KAT
SHA-1 , SHA-256	KAT
HMAC SHA-1, SHA-256	KAT

.

Conditional Self Tests

Table 7 lists the conditional self tests performed by the module.

Table 7 Functional Block Diagram

Algorithm	Test Procedure
Approved RNG	Continuous test
RSA	Pairwise consistency test

Security Requirements

Secure Installation

The cryptographic module is installed on the Cisco CARS 1.2.0.182 operational environment of the Cisco ACS Server Application.

The following steps must be performed to install and initialize the cryptographic module for operating in a FIPS 140-2 compliant manner:

- The OS must be configured to a single user mode of operation.

Invoking the Approved Mode of Operation

The following policy must always be followed in order to achieve a FIPS 140-2 mode of operation:

- Calling the function CryptoLibrary: Init() will place the module in FIPS mode of operations.
- Only FIPS approved or allowed algorithms may be used.

Design Assurance

All source code and documentation is stored in Cisco's version control system.

The structure of the module's components corresponds directly to the security policy's rules of operation. The security mechanisms provided by the software including access control and cryptographic functionality, are addressed in the Security Policy. The Security Policy contains explicit instructions about how the module is accessed.

The module is coded in C++.

Mitigation of Other Attacks

The module is not designed to mitigate any other attacks.

Obtaining Documentation, Support & Security Guidelines

For information on obtaining documentation, obtaining support, providing documentation feedback, security guidelines, and also recommended aliases and general Cisco documents, see the monthly What's New in Cisco Product Documentation, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html