Secure Network Communications

FIPS 140-2 Non-Proprietary Security Policy

21 June 2010

AccessData®

Table of Contents

Introduction

The AccessData Secure Network Communications FIPS 140-2 Module is a cryptographic module that operates as a multi-chip component library positioned between the OpenSSL API and a host application as illustrated in Figure 1. The Module provides to any AccessData application that utilizes it, electronic encryption designed to prevent unauthorized access to data transferred across a physical or wireless TCP/IP network. Instances of the Module operating concurrently on two general purpose computers encrypt data placed on, and decrypt data read from, the network; protecting the user of the application on the hardware platform.

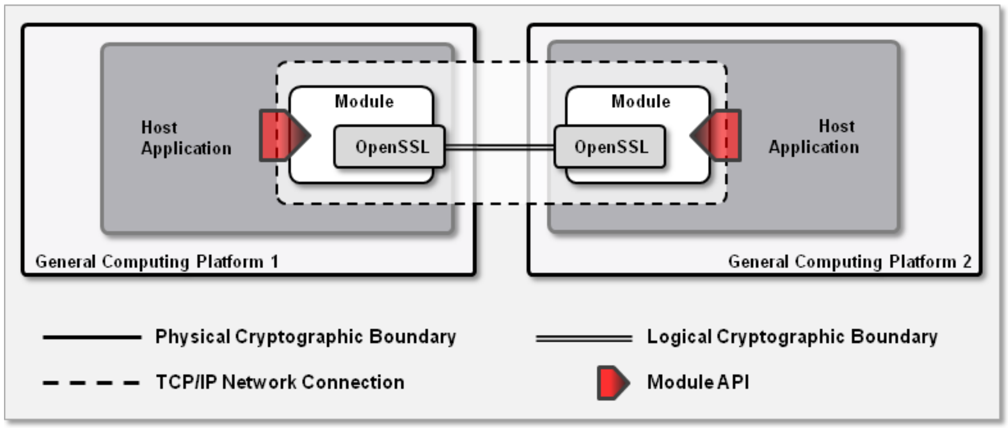
Host Application
Module
OpenSSL
Module
OpenSSL
Host Application
General Computing Platform 1
General Computing Platform 2
Physical Cryptographic Boundary
Logical Cryptographic Boundary
TCP/IP Network Connection
Module API

Figure 1 – Conceptual diagram of the AccessData Secure Network Communications Module

This document is the non proprietary FIPS 140-2 security policy for the AccessData Secure Network Communications software module to meet FIPS 140-2 level 1 requirements. This Security Policy details the secure operation of the AccessData Secure Network Communications FIPS 140-2 Module as required in Federal Information Processing Standards Publication 140 2 (FIPS 140-2) as published by the National Institute of Standards and Technology (NIST) of the United States Department of Commerce.

Module Specification

For the purposes of FIPS 140-2 validation the AccessData Secure Network Communications FIPS 140-2 Module v1.0 is defined as a dynamically-linked library (DLL). The term “Module” elsewhere in this document refers to this AccessData Secure Network Communications FIPS 140-2 Module.

The Module provides an API for invocation of FIPS approved cryptographic functions from calling applications. The Module is designed for use in AccessData applications that will be transmitting information over a general purpose network securely.

The Module was tested by the FIPS 140-2 Cryptographic and Security Testing (CST) laboratory for the specific platform identified in Table 1.

The Module provides confidentiality, integrity, and message digest services. It natively supports the following algorithms: AES, RSA (for digital signatures and key wrapping), SHA 1, SHA 224, SHA 256, SHA 384, SHA 512, and HMAC SHA 1, HMAC SHA 224, HMAC SHA 256, HMAC 384, HMAC SHA 512, ANSI X9.31 compliant deterministic random number generation.

Platform	Module File Name
Microsoft Windows XP SP3	FipsComm.dll

Table 1- FIPS Module by Platform

Ports and Interfaces

For the purposes of this FIPS 140-2 validation, the Module is considered a multi-chip standalone module. Although the Module is software the physical embodiment is a general purpose computer that consists of multiple components, considered to be a multi-chip standalone module by FIPS 140-2.

The logical cryptographic boundary for the Module is the dynamically-linked library, FipsComm.dll. The physical cryptographic boundary contains the general purpose computing hardware of the system executing the application. This system hardware includes the central processing unit(s), cache and main memory (RAM), system bus, and peripherals including disk drives and other permanent mass storage devices, network interface cards, and other internal system components.

The Module provides a logical interface via an Application Programming Interface (API). This logical interface exposes services that applications may utilize directly or extend to add support for new data sources or protocols. The API provides functions that may be called by the referencing application.

The API interface provided by the Module is mapped onto the FIPS 140-2 logical interfaces: data input, data output, control input, and status output. Each of the FIPS 140-2 logical interfaces relate to the Module's callable interface, as follows:

Data Input:	Input parameters to all functions that accept input from Crypto Officer or User entities
Data Output:	Output parameters from all functions that return data as arguments or return values from Crypto Officer or User entities
Control Input:	All API function input into the module by the Crypto Officer and User entities
Status Output:	information returned via exceptions (return/exit codes) to Crypto Officer or User entities

Approved Cryptographic Algorithms

The Module supports the following FIPS approved cryptographic algorithms:

Rivest Shamir Adleman (RSA) PKCS #1 digital signature
Advanced Encryption Standard (AES – FIPS 197)
Secure Hashing Algorithm (SHA 1, SHA 2 – FIPS 180-3)
Keyed Hash Message Authentication Code (HMAC – FIPS 198-1)

The module also supports the following non-approved algorithms that may be used in FIPS mode as part of the TLS protocol:

HMAC MD5
MD5
Rivest Shamir Adleman (RSA) – key wrapping; key establishment mechanism provides between 80 and 256 bits of encryption strength

The Module performs ANSI X9.31 random number generation.

Approved Algorithms				
Algorithm Type	Algorithm	Standard	FIPS Validation Certification Number	Usage

Asymmetric Keys	RSA	ALG[ANSIX9.31]; SIG(gen); SIG(ver);
		
		ALG[RSASSA-PKCS1_V1_5]; SIG(gen); SIG(ver)
		
		ALG[RSASSA-PSS]; SIG(gen); SIG(ver);
			626	Sign and Verify Operations

Symmetric Key	AES – CBC, CFB8, CFB128, ECB, OFB each with 128, 192, or	FIPS 197	1307	encrypt/decrypt operations

	256 bit keys			
HMAC	HMAC-SHA-1 HMAC-SHA-224 HMAC-SHA-256 HMAC-SHA-384 HMAC-SHA-512	FIPS 198	759	module integrity code integrity Message integrity
Hashing	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512	FIPS 180-2	1195	hashing
RNG	ANSI X9.31	ANSI X9.31	729	Random number generation

Table 2 – Approved Cryptographic Algorithms

Approved Mode of Operation

The module only has an approved mode of operation. In order to initialize the module and make use of the APIs, it is first necessary to perform the initialization call `FIPSVerify()`. The module's APIs will return an error and perform no operations until this call is made and successfully returns. Calling the `FIPSVerify()` function invokes the module's power-up self-tests and returns the status indicator of the test results. Calling the function after the initialization returns the state of the internal flag, acting as a FIPS mode indicator.

The DLL initialization is performed when the application invokes the `FIPSVerify()` call. Prior to using any Module functions `FIPSVerify`() must be called successfully. Execution of any function without having called `FIPSVerify()` first will fail as the DLL has not yet been initialized.

The `FIPSVerify()` function relies upon a composite integrity test (composed of both its own HMAC SHA-1 test and the HMAC SHA-1 integrity test used by the embedded OpenSSL FIPS Object module) to prevent code corruption and also performs algorithm power up self tests (Pairwise Consistency and Known Answer tests) to prevent cryptographic algorithm corruption. If any power up self test fails, the internal global error flag `FIPS_FAILED` is set to prevent subsequent invocation of any cryptographic function calls. If all components of the power up self test are successful then `FIPSVerify()`sets the internal flag to `TRUE`.

Test Environment

The Module was tested by the FIPS 140-2 CST laboratory on Windows XP SP3.

Roles, Services and Authentication

Roles and Services

The User and Crypto Officer roles are implicitly assumed by any entity that can access services implemented in the Module. In addition the Crypto Officer role can install and initialize the Module; this role is implicitly entered when installing the Module or performing system administration functions on the host operating system:

User Role	All services except installation
Crypto Officer Role	All services including installation

The Module meets the FIPS 140-2 level 1 requirements for Roles and Services for User and Crypto Officer Roles. As a library and as allowed by FIPS 140-2 the Module does not support user identification or authentication for those roles.

Authentication

The Module does not provide identification or authentication mechanisms that would distinguish between the two supported roles. These roles are implicitly assumed by the services that are accessed, and can be differentiated by assigning module installation and configuration services to the Crypto Officer. Only a single user in a specific role may access Module services at the same time.

Authorized Services

The services provided by the Module are listed in the following table. All services may be performed in both User and Crypto Officer Roles except for the Module installation service which may only be performed by in the Crypto Officer role:

Roles	Services	Ephemeral Critical Security Parameters	Algorithm	CSP Access
User, Cryptographic Officer	Symmetric Encryption / Decryption	Symmetric Key	AES, Triple-DES	Execute
Cryptographic Officer	Digital Signature Generation / Verification	Asymmetric Private Key	RSA	Execute
Cryptographic Officer	Installation	--	--	--

User, Cryptographic Officer	Module Initialization	--	--	--
User, Cryptographic Officer	Power-up Self Test	--	--	--
User, Cryptographic Officer	Random Number Generation	RNG Seed and RNG Seed Key	X9.31 RNG	Execute
User, Cryptographic Officer	Key Establishment	Asymmetric public and private keys	RSA	Read, Write, Execute
User, Cryptographic Officer	Show Status	--	--	--

Table 3 – Authorized Services

Finite State Model

The Module implements the finite state machine detailed in Appendix A.

Operational Environment

Applications referencing the Module run as processes under the control of the host system operating system.

Modern operating systems segregate running processes into virtual memory areas that are logically separated from all other processes by the operating system and CPU. The Module functions completely within the process space of the process which loads it. It does not communicate with any processes other than the one that loads it, and satisfies the FIPS 140-2 requirement for a single user mode of operation.

Rules of Operation

1. The Module API is accessible only after the DLL initialization using the `FIPSVerify()` function call. All APIs will return errors and perform no operations until this call is made.
2. The replacement or modification of the Module by unauthorized intruders is prohibited.
3. The Operating System enforces authentication method(s) to prevent unauthorized access to Module services.
4. The referencing application accessing the Module runs in a separate virtual address space with a separate copy of the executable code.

5. The unauthorized reading, writing, or modification of the address space of the Module is prohibited.
6. The writable memory areas of the Module (data and stack segments) are accessible only by a single application so that the Module is in "single user" mode, i.e. only the one application has access to that instance of the Module.
7. The operating system is responsible for multitasking operations so that other processes cannot access the address space of the process containing the Module.
8. Secret or private keys that are input to or output from an application must be input or output in encrypted form using a FIPS Approved algorithm. Note that keys exchanged between the application and the Module may not be encrypted.

Software Security

The module relies upon a composite integrity test using both its own HMAC SHA-1 test and the HMAC SHA-1 integrity test used by the embedded OpenSSL FIPS Object module. Both tests combine to ensure that the module's integrity has been maintained. This test is performed each time the module is initiated.

Critical Security Parameters

A Critical Security Parameter (CSP) is information, such as passwords, symmetric keys, asymmetric private keys, etc., that must be protected from unauthorized access. Since the Module is accessed via an API from a referencing application, the Module does not manage CSPs, except for the internal RNG seed and seed key used during random number generation. In fact, for most applications CSPs will be found in multiple locations external to the Module, such as in application buffers, primary (RAM) memory, secondary disk storage, CPU registers, and on the system bus. In the case of networked client server applications some CSPs will be found on both the client and server system and on the network infrastructure in between (Ethernet and WAN communication lines, routers, switches).

The application designer and the end user share a responsibility to ensure that CSPs are always protected from unauthorized access. This protection will generally make use of the security features of the host hardware and software which is outside of the cryptographic boundary defined for this Module. All of these ephemeral keys, including the module managed RNG Seed and RNG Seed Key, can be procedurally zeroized by cycling the power of the general purpose computer system thereby erasing all keys from memory, since they are only stored in RAM.

While not considered CSPs, the module does store the plaintext HMAC Integrity Keys within its cryptographic boundary. These keys are only used to perform the software integrity tests of the module, cannot be accessed by the operator, and are neither input nor output.

Self Tests

The Module performs a number of power up and conditional self tests to ensure proper operation of the Module. Power up tests include cryptographic algorithm known answer tests and integrity tests. The integrity tests are performed using HMAC SHA 1 digests calculated over the module's executable code. Power-up tests are run automatically when the Module is initialized. Additionally, software integrity tests may be executed at any time by calling the `FIPSVerify()`function and verifying it returns true. All self-tests can be executed by unloading the module and re-performing the initialization using the `FIPSVerify()` function. No cryptographic functionality will be available until after successful execution of all power up tests. No authentication is required to perform self tests either automatically or upon demand.

The failure of any power up self test or continuous test causes the Module to enter the Self Test Failure state (see Appendix A), and all cryptographic operations are disabled until the Module is reinitialized with a successful `FIPSVerify()`call. Note the most likely cause of a self test failure is memory or hardware errors. In practice a self test failure means the application must exit and be restarted.

Power up Tests

Known Answer Tests (KATs) are tests where a cryptographic value is calculated and compared with a stored previously determined answer. The power up self tests for the following algorithms use a KAT:

Algorithm	Known Answer
AES	encryption and decryption with 128 bit key
RSA	known answer test with 1024 bit key public encryption and private decryption with 1024 bit key; sign and verify test with 1024 bit key
HMAC	HMAC SHA-1 HMAC SHA-224 HMAC SHA-256 HMAC SHA-384 HMAC SHA-512
RNG	Generation of a known value using the ANSI X9.31 RNG

Table 4 – Power-Up Test Algorithms

Conditional Tests

In addition to the power up tests, the Module performs several conditional tests including pair wise consistency tests on newly generated public and private key pairs. Conditional tests are performed automatically as necessary and cannot be turned off. Currently, all conditional tests relate to services available only to users. Thus, conditional and critical function tests are not performed at any time in response to Crypto Officer actions.

Algorithm	Conditional Test
RSA	pairwise consistency test (public encryption and private decryption with the newly generated keypair)
RNG	Continuous Random Number Generation Test implemented for the approved RNG

Table 5 – Conditional Test Algorithms

Pair wise Consistency Test
A pairwise consistency test is performed when RSA key pairs are generated by applying a private key to the ciphertext and verifying that the result equals the original plaintext.

Software/Firmware Load Test
Not applicable; the Module does not utilize externally loaded cryptographic modules.

Manual Key Entry Test
Not applicable; keys are not manually entered into the Module.

Bypass Test
Not applicable; the Module does not implement a bypass capability.

Critical Function Tests
The Module does not mitigate against any specific attacks.

Design Assurance
The Module is managed in accordance with the established configuration management and source version control procedures of the AccessData Secure Network Communications project.

Application Management of Critical Security Parameters

Identifying CSPs
All CSPs must be created, stored, and destroyed in an approved manner as described by FIPS 140-2. CSPs are those items of information which must be protected from disclosure, such as symmetric keys, asymmetric private keys, etc. Note that the application designer and end user/system administrator/Crypto Officer share a responsibility for protection of CSPs; the former to include appropriate technical protections and the latter to install and configure the application correctly. Technical protections include checks to require that files storing CSPs have appropriate permissions (not group writable or world readable, for example). Administrative protections include installation of the runtime software (executables and configuration files) in protected locations. End users have a responsibility to refrain from comprising CSPs (as by sending a password in clear text or copying an encryption key to an unprotected location).

Storage of CSPs

The Module does not store any critical security parameters (CSPs) in persistent media; while the Module is initialized any CSPs reside temporarily in RAM and are destroyed at the end of the session. Any keys or other CSPs otherwise stored in persistent media must be protected in accordance with FIPS requirements in Reference 1, FIPS 140-2.

Destruction of CSPs

All keys and CSPs within the cryptographic module are stored within the RAM of the general purpose computer system. The values are procedurally zeroized when the general purpose computer system is powered down causing the RAM to be erased.

Appendix A - Finite State Model

This Appendix describes the Finite State Machine (FSM) model for an application utilizing the AccessData Secure Network Communications FIPS Module. Figure A.1 is a finite state diagram showing the states and transitions between states. At any point in time the Module is in one and only one state. Various software or operating system driven events can cause a transition to another state.

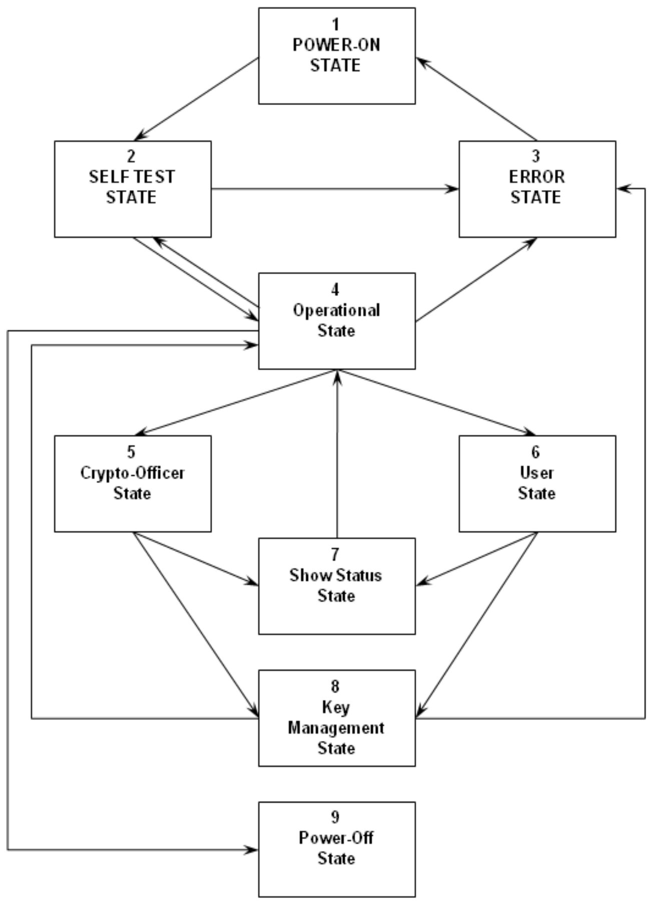

Diagram A.1 Finite State Machine Diagram

State Descriptions

1. Power-On State

The application making use of the Module has not been loaded into memory by the host operating system. The Module transitions to the Power On State when the application is invoked as a process by the host operating system and thereby loads the module into memory.

2. Self Test State

The application has been loaded into memory for a process created by the host operating system, but the power up self tests and DLL initialization (`FIPSVerify()` call) have not yet been performed. The `FIPSVerify()` call will transition to either the Error or Operational state. Any of the following errors can occur during the power up self test, all cause a transition to the Error state:

SUCCESS	Returned when a function successfully completes
RESEND	Returned when the data must be resent to the peer
CLOSED	Returned when a connection has been properly closed
FAILED	Indicates a general failure to properly execute
BUFFEROVERFLOW	Returned when the data returned is larger than the target buffer
FAILED_FIPS	Returned on self-test failure or incomplete DLL initialization
MEMORY_ERROR	Indicates a general memory allocation failure
CERT_ERROR	Returned when certificate file is not found or fails to load

3. Error State
The initial power up self test or subsequent optional self test has failed. The application and Module will typically terminate on detection of the power up self test error. While not likely in practice, a successful re invocation of the power up self test could transition to the Operational state.

4. Operational State
The power up self test has executed successfully. The cryptographic algorithms in the Module can now be accessed by the application. The Module will remain in the Operational state until the application is terminated and enters the Power Off state.

5. Crypto-Officer State
The application is in crypto officer state.

6. User State
The application is in user state.

7. Show Status State
The application is performing a show status operation.

8. Key Management State
The application is performing a key management operation.

9. Power-Off State
The host operating system has terminated the application process and released all memory.