EFJohnson Encryption Module

(Software Version 1.0.0.5)

FIPS 140-2 Non-Proprietary Security Policy

Level 1 Validation
Version 0.13

December, 2004

Table of Contents

Introduction

Purpose

This is a non-proprietary Cryptographic Module Security Policy for the EFJohnson Encryption Module from EFJohnson, Inc. This security policy describes how the EFJohnson Encryption Module meets the security requirements of FIPS 140-2 and how to run the module in a secure FIPS 140-2 mode. This policy was prepared as part of the Level 1 FIPS 140-2 validation of the module.

FIPS 140-2 (Federal Information Processing Standards Publication 140-2 — *Security Requirements for Cryptographic Modules*) details the U.S. 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) Cryptographic Module Validation Program (CMVP) website at http://csrc.nist.gov/cryptval/.

The EFJohnson Encryption Module is referred to in this document as the Encryption Module, the EM, or the module.

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 EFJohnson, Inc. website (http://www.efjohnson.com) contains information on the full line of products from EFJohnson.

- The CMVP website (http://csrc.nist.gov/cryptval/) contains contact information for answers to technical or sales-related questions for the module.

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 Machine
- Other supporting documentation as additional references

With the exception of this Non-Proprietary Security Policy, the FIPS 140-2 Validation Documentation is proprietary to EFJohnson, Inc. and is releasable only under appropriate non-disclosure agreements. For access to these documents, please contact EFJohnson.

EFJOHNSON ENCRYPTION MODULE

Overview

The EFJohnson Encryption Module is a software cryptographic module that serves both as a key store and a cryptographic service provider for integration into an OTAR system as specified in the TIA/EIA Telecommunications Systems Bulletin, APCO Project 25, Over-The-Air-Rekeying (OTAR) Protocol, New Technology Standards Project, Digital Radio Technical Standards, TSB102.AACA, January, 1996, Telecommunications Industry Association and a 3DES/AES OTAR system as specified in ANSI/TIA-102.AACA-1-2002 titled, Project 25- Digital Radio Over-the-Air-Rekeying (OTAR) Protocol Addendum 1 – Key Management Security Requirements for Type 3 Block Encryption Algorithms. The module is accessible through an intuitive API, and provides an easy-to-use yet secure means of storing sensitive cryptographic keys. The Encryption Module meets level 1 FIPS 140-2 requirements and achieves level 2 in the "Roles, Services, and Authentication" and "Operation Environment" sections of FIPS 140-2. The following table details the level met in each individual FIPS 140-2 section.

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	2
4	Finite State Model	1
5	Physical Security	N/A
6	Operational Environment	2
7	Cryptographic Key Management	1
8	EMI/EMC	1
9	Self-tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A

The Encryption Module is composed of two binaries, the Cryptography and Key Storage Module (CKSM) and the Root Key Provider (RKP) service. The CKSM is a dynamic link library (CCEFFIPS.DLL) that serves as an interface into the module and provides cryptographic services, and the RKP service is an executable file (EFJRKP.EXE) that authorizes operators of the module.

In FIPS 140-2 terminology, the EFJohnson Encryption Module is defined as a multi-chip standalone cryptographic module. The module is capable of running on any standard Personal Computer (PC) supported by Windows 2000, and, for the purposes of the FIPS 140-2 validation, it was tested on the Common Criteria (CC) certified EAL 4 augmented Microsoft Windows 2000 compliant with Controlled Access Protection Profile (CAPP) running on Dell OptiPlex GX400. The Windows 2000 CC evaluation report can be found at http://niap.nist.gov/cc-scheme/st/ST_VID4002.html. The Controlled Access Protection Profile (CAPP)

to which Windows 2000 conforms to can be found at http://niap.nist.gov/cc-scheme/pp/PP_CAPP_V1.d.pdf.

The physical cryptographic boundary of the Encryption Module is the case of the PC, which completely encloses the module's components. The following diagram

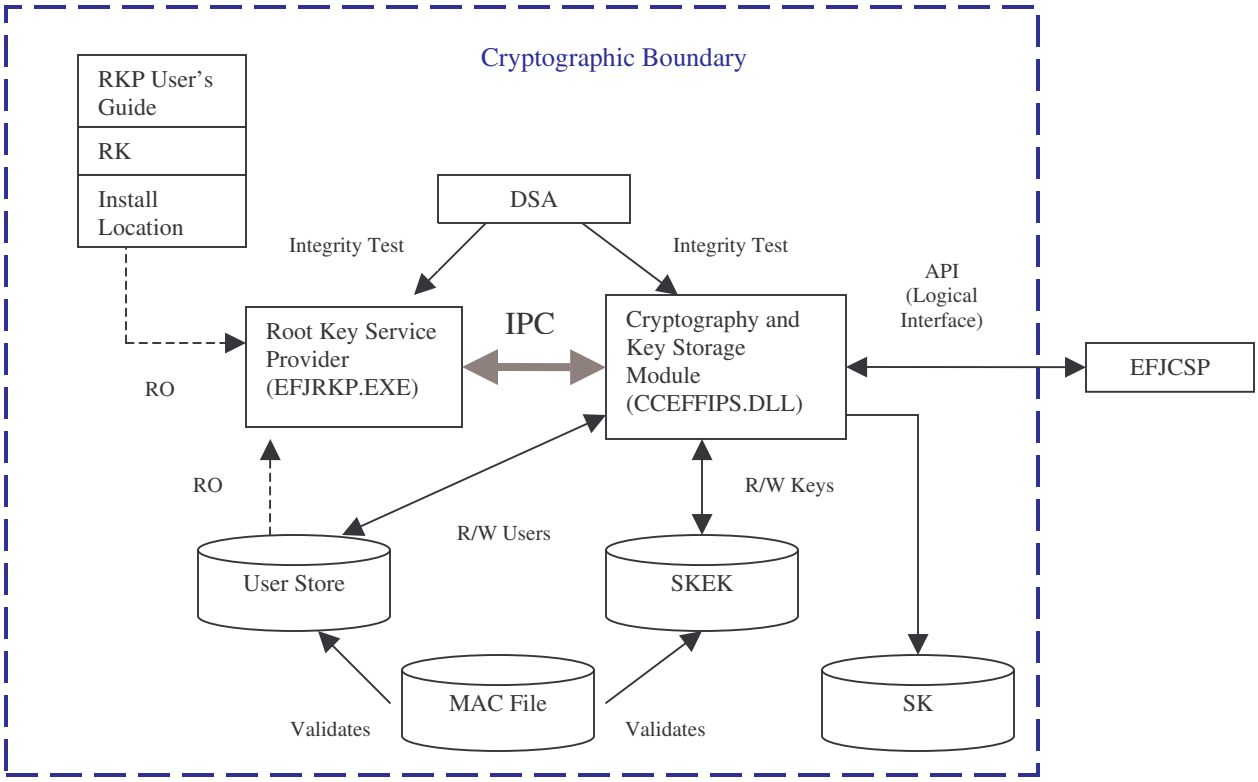

Figure 1 - Cryptographic Boundary

specifies the logical cryptographic boundary.

Module Interfaces

The Encryption Module runs on any of the hardware platforms utilized in the Windows 2000 CC certification. These evaluated configurations include the following hardware.

- Compaq Proliant ML570
- Compaq Proliant ML330 (both 2-processor and 4-processor version)
- Compaq Professional Workstation AP550
- Dell Optiplex GX400
- Dell PE 2500
- Dell PE 6450

- Dell PE 2550
- Dell PE 1550

The Encryption Module's physical ports are composed of the physical ports provided by the hardware platforms listed above. These standard PC ports include the monitor, keyboard, mouse, serial ports, parallel ports, USB ports, floppy disk drive, CD-ROM drive, and network ports.

The Encryption Module's logical interface is the API calls of the CKSM, which provide the only means of accessing the module's services. Data inputs are certain function calls, including specific arguments to a function. These specific arguments are pointers to input data. Control inputs are also certain function calls, including specific arguments. However, these arguments control how the function is executed. Operational state of RKP service also acts as a Control Input for the module. The result of a function is data output. Some functions include an argument that specifies where the result (data output) should be stored. The return values of the functions are status outputs.

The logical interfaces and their module and physical interface mappings are described in the following table.

Table 2 - FIPS 140-2 Logical, Physical, and Module Interface Mapping

PC Physical Port	Module Interface	FIPS 140-2 Logical Interface
Keyboard, mouse, CD-ROM, floppy disk, and serial/USB/parallel/network ports	Function calls that accept, as their arguments, data to be used or processed by the module	Data Input Interface
Hard Disk, monitor, and serial/USB/parallel/network ports	Arguments for a function that specify where the result of a function is stored	Data Output Interface
Keyboard, CD-ROM, floppy disk, mouse, and serial/USB/parallel/network port	Function calls utilized to initiate the module and the function calls used to control the operation of the module	Control Input Interface
Hard Disk, monitor, and serial/USB/parallel/network ports	Return values	Status Output Interface
Power Interface	Not Applicable	PC Power Interface

Roles and Services

The module supports role-based authentication and contains three roles: RKP User role, Crypto Officer role, and User role. The Users, Crypto Officers, and the RKP User are all defined by accounts on the Windows 2000 Operating System, and these accounts then have permission mapped into the module (by the Crypto-Officer). Operators authenticate to the CC certified Windows 2000 Operating System using a username and password, and the Encryption Module provides authorization to the operators upon loading of the CKSM. The module supports multiple concurrent operators.

RKP User Role

The RKP User is only used interactively (physical operator) to install the Encryption Module on the system. During installation, the RKP User creates an empty key store and establishes an initial Crypto-Officer. After installation completes, this role is no longer used by a physical operator. Since the RKP User is a member of the Windows Administrator group, all Windows Administrators can start or stop the RKP service. RsaCcefBootKeyStore function should not be executed upon an installed module. This is because, if this function is executed on an installed module, a new empty key store and a user store with one user will be created thus deleting all the previous keys and users.

Table 3 - RKP User Services

Functions	Service	Input	Output	Accessed CSPs	Permission
RsaCcefBootKeyStore	Creates a new empty key store and a user store with a single user, the initial CO from the installation program during initial system installation and configuration.	MAC Key information, CO user information	Status output	Root Key, BootMAC Key	Read/Write

Crypto Officer Role

The Crypto Officer role has the responsibility of managing both the Key Store and the User Store. Managing the Stores comprises the ability to add or delete an individual key to or from the Key Store and delete all keys from the Key Store. The Crypto Officer is able to access all the system management functions as well as all the cryptographic functionalities of the module. The services available only to the Crypto Officer are provided in Table 4.

Table 4 - Crypto Officer Services

Functions	Service	Input	Output	Accessed CSPs	Permission
RsaCcefAddUser	Adds a single file to the User Store. The file's name is derived from the new user's (CO or End User) name, and their permissions.	CKSM session handle, CO user information	Status output	SKEK File MAC Key	Read
RsaCcefModifyUser	Modifies the permissions of a CO or regular user in the user store.	CKSM session handle, user information	Status output	SKEK File MAC Key	Read
RsaCcefDeleteUser	Removes a CO or an End User from the EFJEM User Store.	CKSM session handle, user name (CO or End User)	Status output	--	
RsaCcefGenerateNewSkek	Generates a new SKEK and either installs it into the EFJEM or returns it to the caller.	CKSM session handle, new SKEK information	Status output	SKEK KEKs, SKEK KeyInfo MAC Key, SKEK File MAC Key	Read/Write

Functions	Service	Input	Output	Accessed CSPs	Permission
RsaCcefInstallNewSkek	Installs a previously generated SKEK into the EFJEM.	CKSM session handle, SKEK information	Status output	SKEK KEKs, SKEK KeyInfo MAC Key, SKEK File MAC Key	Read/Write
RsaCcefDeleteSkek	Deletes a SKEK file from the disk.	CKSM session handle, current SKEK number	Status output	SKEK KEKs, SKEK KeyInfo MAC Key, SKEK File MAC Key	Delete
RsaCcefDeleteKeyFile	Deletes TEMP keys file, PERM keys file, current SKEK file, the Root Key, or any combination of these.	CKSM session handle, bit-field flag parameter	Status output	Root Key, SKEK KEKs, SKEK KeyInfo MAC Key, SKEK File MAC Key, KS Keys	Delete
RsaCcefEscrowKeyStore	Escrows all of the keys in the PERM keys file to a key escrow structure.	CKSM session handle, key handle	Status output	SKEK KEKs, SKEK KeyInfo MAC Key, SKEK File MAC Key, KS Keys	Read
RsaCcefRestoreKeyStore	Restores escrowed keys to a PERM keys file.	CKSM session handle, key handle	Status output	a) SKEK KEKs, SKEK KeyInfo MAC Key, SKEK File MAC Key b)KS Keys	a)Read b)Write

User Role

Users are allowed only to use the module to perform cryptographic functions. The Encryption module authenticates the Users using Windows 2000 SID and password. Users may use a key from the Key Store or a key that exists in memory to perform cryptographic functions. All the services provided for Users, are also available to Crypto Officers. Service descriptions and inputs/outputs available for Users are listed in the following table:

Table 5 - User Services

Functions	Services	Input	Output	Accessed CSPs	Permission
RsaCcefOpenFipsLibrary	Initialize an EFJEM session.	CKSM session handle	Status output	Root Key, SKEK KEKs, SKEK KeyInfo MAC Key, SKEK File MAC Key	Read
RsaCcefCloseFipsLibrary	Uninitialize the EFJEM session.	CKSM session handle	Status output	--	
RsaCcefCreateKeyHandleGenerate	Generate a new key handle	CKSM session handle, key info, key handle.	Status output	a) KS key b) SKEK KEKs, SKEK KeyInfo MAC Key, SKEK File MAC Key	a) Read b) Write

Functions	Services	Input	Output	Accessed CSPs	Permission
RsaCcefCreateKeyHandleFromInfo	Create a key handle from key data	CKSM session handle, key info, key handle.	Status output	a) KS key b) SKEK KEKs, SKEK KeyInfo MAC Key, SKEK File MAC Key	a)Read/Write b) Read
RsaCcefWrapKey	Wrap a key	CKSM session handle, key info, two key handles	Status output	a) KS key b) SKEK KEKs, SKEK KeyInfo MAC Key, SKEK File MAC Key	a)Read/Write b) Read
RsaCcefDestroyKeyHandle	Destroy a key handle	CKSM session handle, key handle	Status output	--	
RsaCcefCreateBlockCipherMac	Create a MAC using block cipher	CKSM session handle, key handle, data to MAC	MAC, status output	KS Key	Read
RsaCcefAddSeed	Add seed	CKSM session handle, seed value length	Seed, status output		
E_SKEY_new	Create a key object	CKSM session handle, key information	Key object, status output	a) KS key b) SKEK KEKs, SKEK KeyInfo MAC Key, SKEK File MAC Key	a)Read/Write b) Read
E_SKEY_free	Destroy a key object	Key object	Status output	KS Key	Delete
E_SKEY_set_info	Set key data to a key object	Key object, key object ID, key data	Status output	KS Key	Write
E_SKEY_get_info	Retrieve key data from a key object.	Key object, key object ID	Key data, status output	KS Key	Read
E_CR_new	Create a crypto object	CKSM session key, algorithm ID, algorithm type	Crypto object, status output	--	
E_CR_free	Destroy a crypto object	Crypto object	Status output	--	
E_CR_set_info	Add information to a cryptographic object	Crypto object, object ID, object information	Status output	KS Key	Read
E_CR_encrypt_init	Initialize a crypto object for encryption	Crypto object, key object, IV	Status output	KS Key	Read
E_CR_encrypt_update	Encrypt bulk input data	Crypto object, data, data length	Cipher, cipher length, status output	KS Key	Read
E_CR_encrypt_final	Finish the encryption process	Crypto object	Cipher, cipher length, status output	KS Key	Read
E_CR_encrypt	Encrypt the input data	Crypto object, data, data length	Cipher, cipher length, status output	KS Key	Read
E_CR_decrypt_init	Initialize a crypto object for decryption	Crypto object, key object, IV	Status output	KS Key	Read

Functions	Services	Input	Output	Accessed CSPs	Permission
E_CR_decrypt_update	Decrypts bulk input data	Crypto object, data, data length	Plaintext, plaintext length, status output	KS Key	Read
E_CR_decrypt_final	Finish the decryption process	Crypto object	Plaintext, plaintext length, status output	KS Key	Read
E_CR_decrypt	Decrypt the input data	Crypto object, data, data length	Plaintext, plaintext length, status output	KS Key	Read

Authentication Mechanisms

The module depends upon the Windows 2000 OS to provide a username and password based authentication mechanism. Each valid username has an associated role, and the username can only be used to gain access to the module with the associated password.

The implementation is the one that is used by the CC EAL 4 certified Windows 2000 Operating system of the module. The minimum length of the password is 8 and the module requires the password to be alpha numeric. Also, the complexity requirement for password is it cannot be all numeric or all alphabets (26 lowercase and 26 uppercase letters). Assuming at least 90 characters with repetition, the chance of a random attempt falsely succeeding is 1 in 4,251,212,271,468,544.

Physical Security

EFJohnson Encryption Module is a software cryptographic module running on the Common Criteria (CC) certified Windows 2000 operating system, and the module's software is entirely encapsulated by the cryptographic boundary shown in Figure 1. The physical cryptographic boundary is the case of the PC.

While purely a software module, the FIPS 140-2 evaluated platform must be a standard PC that has been tested for and meets applicable FCC EMI and EMC requirements for business use as defined by 47 Code of Federal Regulations, Part15, Subpart B.

Cryptographic Key Management

The Encryption Module contains numerous functions for performing cryptographic operations. RNGs, MAC functions, and symmetric algorithms use keys and/or CSPs to perform their respective operations.

The module implements the following FIPS-approved algorithms (all statically linked from the FIPS validated RSA Crypto-C ME library version 1.7.2):

- AES (FIPS 197)– ECB, CBC, CFB (128 bits), and OFB modes; 128/256 bit key sizes (certificate #26)
- Triple DES (FIPS 46-3) - ECB, CBC, CFB (64 bits), and OFB modes; 1 keying option (certificate #135)
- DES (FIPS 46-3 – for legacy use only) - ECB, CBC, CFB (64 bits), and OFB modes (certificate #186)
- SHA-1 (FIPS 180-2) (certificate #121)
- DSA (FIPS 186-2) (certificate #72)
- FIPS 186-2 Deterministic RNG (FIPS 186-2 Appendix 3.1 Change Notice 1 with G function built from SHA-1) (certificate #14)

The module implements the following non-FIPS-approved algorithms which must not be used while in a FIPS mode:

- AES-MAC (Certification #26, P25 AES OTAR, vendor affirmed)

The module supports the following critical security parameters:

Table 6 - List of CSPs for the Encryption Module

Key	Key type	Generation	Storage	Use
Root Key	256-bit AES key	Internally generated during installation using the FIPS 186-2 RNG	In non-volatile memory (hard drive – Windows registry) plaintext	Encrypts the SKEK
BootMAC key	256-bit AES key	Internally generated during installation using the FIPS 186-2 RNG	In non-volatile memory (hard drive) encrypted by the RK	MACs the initial CO user file.
TDES SKEK KEK	192-bit key	Internally generated by the FIPS 186-2 RNG , or externally generated and loaded into the module	In SKEK folder in non-volatile memory (hard drive) encrypted by the RK	Encrypts Key Store Keys
AES SKEK KEK	256-bit key	Internally generated by the FIPS 186-2 RNG , or externally generated and loaded into the module	In SKEK folder in non-volatile memory (hard drive) encrypted by the RK	Encrypts Key Store Keys.
AES SKEK KeyInfo MAC key	256-bit key	Internally generated by the FIPS 186-2 RNG , or externally generated and loaded into the module	In SKEK folder in non-volatile memory (hard drive) encrypted by the RK	MACs Key Store keys
AES SKEK File MAC key	256-bit key	Internally generated by the FIPS 186-2 RNG , or externally generated and loaded into the module	In SKEK folder in non-volatile memory (hard drive) encrypted by the RK	MACs Key Store files
User password	Windows 2000 SID password	Entered electronically	Windows registry in plaintext	Authenticate users to Windows OS
Key Store Keys	DES, TDES, AES-128, or AES-256 keys	Internally generated by the FIPS 186-2 RNG , or externally generated and loaded into the module	In Key Store folder in non-volatile memory (hard drive) encrypted by SKEK	Encrypts data or other keys
EFJRKP Service Module Key	DSA Public Key size 512	Externally generated by RSA and installed with the module	Encrypted in hard drive and plaintext in volatile memory	Used to verify the integrity of EFJRKP.EXE
Key Storage	DSA Public	Externally generated by RSA	Encrypted in hard drive	Used to verify the

Service Module Key	Keys size 512	and installed with the module	and plaintext in volatile memory	integrity of CCEFFIPS.DLL
AES Decrypt Key	AES-128 key	Hard coded created on-site at RSA	Plaintext in hard drive	Decrypts EFJRKP and Key Storage Service Module Keys (DSA Public Keys)

Key generation

Root key and BootMAC key are generated during installation of the module. The symmetric keys placed in the module's key store are either generated internally by the module or imported into the module through its API. These symmetric keys can be imported into the module either in plaintext or encrypted form. DSA public keys are generated externally by the manufacturer and are installed with the module. AES Decrypt Key's raw data is hard coded in the module.

Key storage

The Root Key is stored in plaintext in the Windows Registry on the module's hard drive. The EFJRKP Service Module Key (public key) and Key Storage Service Module Key (public key) are stored encrypted on the module's hard drive and plaintext in the volatile memory. All other CSPs, other than User password and AES Decrypt Key, reside encrypted on the module's hard drive. AES Decrypt Key is stored in plaintext on hard drive.

Key entry and output

All symmetric keys are entered into or output from the module electronically. The symmetric keys can either be encrypted or in plaintext while crossing the cryptographic boundary through the API functions. Password is entered into the module electronically in plaintext. The Root Key and BootMAC Key generated by the configuration file, they do not exit the module. DSA Public Keys and AES Decrypt Key are hard coded and the module does not output them outside the module.

Key zerorization

The Crypto Officer is able to zerorize all the keys (Root Key, BootMAC Key, Symmetric keys) with RsaCcefDeleteSkek or RsaCcefDeleteKeyFile functions. Deletion of the Root Key will include a zerorization of the Root Key registry key (Win32 registry APIs is used), and deletion of this key will disable the module. When the context of a symmetric crypto object is no longer needed, a free function is called to free the context and to zerorize any key-sensitive material in the context. Hard-coded keys are zeroized when the module is deleted from the system.

Self-Tests

The Encryption Module runs power-up and conditional self-tests to verify that it is functioning properly. Power-up self-tests are performed during startup of the

module, and conditional self-tests are executed whenever specific conditions are met.

The module implements the following Power-up self-tests:

Software Integrity Test: The module employs a software integrity test in the form of DSA signature verification with SHA-1.

Cryptographic Algorithm Tests: Known Answer Tests (KATs) are run at power-up for the following algorithms:

- AES 128 KAT (ECB)
- Triple-DES KAT (CBC)
- DES KAT (CBC)
- SHA-1 KAT
- DSA sign/verify test at power up
- PRNG KAT

The module implements the following Conditional self-tests:

- Continuous random number generator test

If any of these self-tests fail, the module will output an error indicator and enter an error state.

Design Assurance

All the source code and associated documentation files are managed and recorded using the Concurrent Versions System (CVS).

Additionally, Microsoft Visual Source Safe (VSS) version 6.0 was used to provide configuration management for the module's FIPS documentation. The VSS Administrator established a "/EFJEM" project in VSS and assigned read/write permissions to the engineers working on the project

Mitigation of Other Attacks

The Encryption Module does not employ security mechanisms to mitigate specific attacks.

SECURE OPERATION

The EFJohnson Encryption Module meets overall Level 1 requirements for FIPS 140-2. The sections below describe how to place and keep the module in FIPS-approved mode of operation.

Initial Setup

The software module will be provided either preinstalled or on physical media by EFJohnson Company for exclusive use in their products.

Before installing EFJohnson Encryption Module, the operator needs to ensure that the system runs EAL 4 augmented CC certificated Microsoft Windows 2000 Professional with Service Pack 3 operating system compliant with the Controlled Access Protection Profile. The module is also compatible to Windows XP Professional SP1, however they are not part of the validated configuration.

The system must be installed, configured, and started before operators may utilize its features. The Encryption Module system installation must be performed by a properly privileged user (a Windows Administrator), who is deemed the RKP User.

Additionally, to maintain the FIPS mode of operation, the operator of the module can only execute FIPS approved services. The AES MAC service cannot be used while in a FIPS approved mode.

Crypto Officer Guidance

Only Crypto officer can add another Crypto Officer or User to the CKSM User Group using RsaCcefAddUser function. The function has the following signature:

```
int RsaCcefAddUser (
  RsaCcefLibCtx libCtx,
  RsaCcefUserInfo *info
);
```

The permissions field of the RsaCcefUserInfo structure is a bit field. The CO must create this argument by OR'ing together one or more of the permission flags, which outline what the user will be allowed to do. The following are the permission flags.

RSA_CCEF_PERMISSION_DATA_ENCRYPTION

RSA_CCEF_PERMISSION_DATA_DECRYPTION

RSA_CCEF_PERMISSION_KEY_ENCRYPTION

RSA_CCEF_PERMISSION_KEY_DECRYPTION

RSA_CCEF_PERMISSION_ADD_KEK

RSA_CCEF_PERMISSION_DELETE_KEK

RSA_CCEF_PERMISSION_ADD_TKEK

RSA_CCEF_PERMISSION_DELETE_TKEK

RSA_CCEF_PERMISSION_IMPORT_SKEK

RSA_CCEF_PERMISSION_EXPORT_SKEK

RSA_CCEF_PERMISSION_RESET_SKEK

RSA_CCEF_PERMISSION_RESET_SIG_KEY

RSA_CCEF_PERMISSION_RESET_ENC_KEY

RSA_CCEF_PERMISSION_SECURITY_MANAGER

The new user will be a CO if and only if the RSA_CCEF_PERMISSION_SECURITY_MANAGER flag is set. The Crypto Officer should be very careful to set the permission flags while creating a new User account.

The User guidance described below also applies to the Crypto Officer.

User Guidance

Two functions are used by all Crypto Officers and regular Users to establish a session with the CKSM and to end that session, RsaCcefOpenFipsLibrary and RsaCcefCloseFipsLibrary. RsaCcefCloseFipsLibrary function must be called once for each call to RsaCcefOpenFipsLibrary. This function finalizes the CKSM session. Allocated memory associated with the library context is freed. The RKP server is contacted to manage the count of active CKSM sessions.

Users may allocate different cryptographic object during the session. It is Users responsibility to destroy the objects, freeing up the memory before closing the session.

ACRONYMS

AES	Advanced Encryption Standard
API	Application Programming Interface
CAPP	Controlled Access PP
CC	Common Criteria
CKSM	Cryptography and Key Storage Module
CMVP	Cryptographic Module Validation Program
CO	Crypto Officer
CSP	Critical Security Parameter
DES	Data Encryption Standard
DSA	Digital Signature Algorithm
EAL	Evaluation Assurance Level
EM	Encryption Module
EFJEM	EFJohnson Encryption Module
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FCC	Federal Communication Commission
FIPS	Federal Information Processing Standard
KAT	Known Answer Test
KS	Key Store
MAC	Message Authentication Code
NIST	National Institute of Standards and Technology
OS	Operating System
PC	Personal Computer
PERM	Permanent Key
PP	Protection Profile
RKP	Root Key Provider
RNG	Random Number Generator
RSA	Rivest Shamir and Adleman
SHA	Secure Hash Algorithm
SKEK	Storage Key Encrypting Key
TEMP	Temporary Key
USB	Universal Serial Bus

