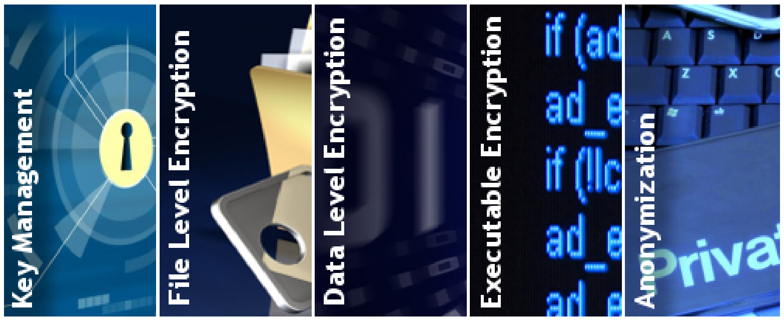
ERUCES, Inc.

Tricryption Cryptographic Module

Software Version: 7.0



FIPS 140-2 Non-Proprietary Security Policy

Level 1 Validation

Document Version 0.9

Prepared for:

ERUCES, Inc.

11142 Thompson Avenue
Lenexa, KS 66219
Phone: (913) 310-0888
Fax: (913) 859-9797
http://www.eruces.com

Prepared by:

Corsec Security, Inc.

10340 Democracy Lane, Suite 201
Fairfax, VA 22030-2518
Phone: (703) 267-6050
Fax: (703) 267-6810
http://www.corsec.com

Revision History

Version	Modification Date	Modified By	Description of Changes
0.1	2008-05-27	Xiaoyu Ruan	Release version
0.2	2008-06-25	Xiaoyu Ruan	Processors in Table 1
0.3	2008-07-16	Xiaoyu Ruan	Minor changes in Section 2.7
0.4	2008-07-18	Xiaoyu Ruan	Added classes ECKey, ECPoint, and RSAkey. Changes in Sections 2.1 and 2.2, Tables 6 and 7.
0.5	2008-08-20	Darryl Johnson	Addressed lab comments post-functional testing
0.6	2008-09-29	Darryl Johnson	Addressed final lab comments for CMVP submission
0.7	2008-10-02	Darryl Johnson	Added cleanup items to final document
0.8	2009-01-08	Darryl Johnson	Addressed CMVP comments
0.9	2009-02-09	Darryl Johnson	Addressed CMVP comments

Table of Contents

Table of Figures

List of Tables

1 Introduction

1.1 Purpose

This is a non-proprietary Cryptographic Module Security Policy for the Tricryption Cryptographic Module from ERUCES, Inc. This Security Policy describes how the Tricryption Cryptographic Module meets the security requirements of Federal Information Processing Standards (FIPS) Publication 140-2, which 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) and Communications Security Establishment Canada (CSEC) Cryptographic Module Validation Program (CMVP) website at http://csrc.nist.gov/groups/STM/index.html.

This document also describes 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. The Tricryption Cryptographic Module is referred to in this document as "the module".

1.2 References

This document deals only with the 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 ERUCES website (http://www.eruces.com) contains information on the full line of products from ERUCES.
- The CMVP website (http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140val-all.htm) contains contact information for answers to technical-related or sales-related questions for the module.

1.3 Document Organization

The Security Policy document is one document in a FIPS 140-2 submission package. In addition to this document, the submission package contains:

- Vendor Evidence
- Finite State Machine
- Other supporting documentation as additional references

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

2 Tricryption Cryptographic Module

2.1 Overview

ERUCES is committed to providing innovative data security solutions that enable users without impeding their work procedures and workflows. ERUCES mission is to offer the most effective, secure, and operationally flexible software tools and products for the implementation of encryption and key management.

ERUCES' Tricryption products and software technologies provide enforceable security at the file and data level, while maintaining secure transparent implementation and operations for:

- Users: collaborative operations without work flow disruption
- IT (Information Technology) administrators: implementation & administrative ease
- IT Security: policy enforcement and accessible monitoring

ERUCES solutions efficiently and effectively integrate with existing security infrastructures and employ current state-of-the-art encryption algorithms and key control mechanisms. Application of our technologies is extremely cost-efficient, both initially and in terms of total cost of ownership.

The Tricryption system is a key management and storage system developed by ERUCES. See Figure 1 – Tricryption Software Component Architecture. The Tricryption system includes a centralized symmetric key management server, Tricryption KeyServer, which manages automatic key generation, key storage and retrieval, key integrity checks and authorization and auditing of key usage. Tricryption KeyServer utilizes symmetric keys (AES[1] or Triple DES[2]) to provide maximum performance and reliability. The Tricryption KeyServer offers a wide range of cryptography, authentication, and authorization options to allow customers to support virtually any global security standard. Using this software, customers can request encryption or decryption of data elements using the other ERUCES software components. The Tricryption KeyServer can be asked for a key, or can be asked to encrypt or decrypt information. The Tricryption KeyServer communicates with the Tricryption Desktop via Transport Layer Security (TLS) sessions.

Figure 1 below depicts the functional components of the Tricryption KeyServer software. Note that only the CSP component of the software comprises the module, and not the entire Tricryption KeyServer software; thus, only the CSP component is within the module's FIPS 140-2 logical cryptographic boundary.

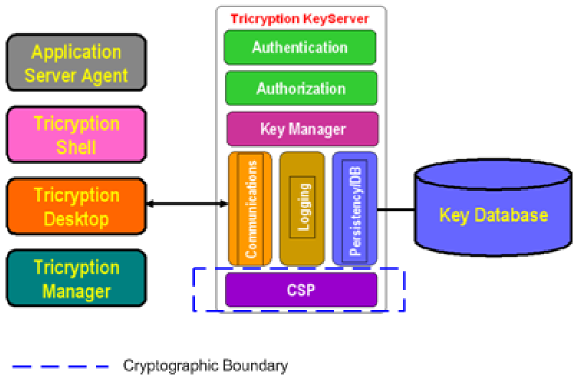


Figure 1 – Tricryption Software Component Architecture[3]

[1] AES – Advanced Encryption Standard
[2] DES – Data Encryption Standard
[3] "CSP" in Figure 1 stands for Cryptographic Server Provider. In the remainder of this document, CSP refers to Critical Security Parameter.

The logical cryptographic boundary surrounds just the library that provides cryptographic services to the software. The library is a dynamic link library (DLL) on Windows Server 2003 R2 or a user-space shared object (SO) on Red Hat Enterprise Linux 5. Figure 2 provides a logical block diagram of the module executing in memory, as well its interactions with surrounding components. Note that the logical cryptographic boundary surrounds only the Tricryption cryptographic module (this is represented in Figure 1 as the CSP).

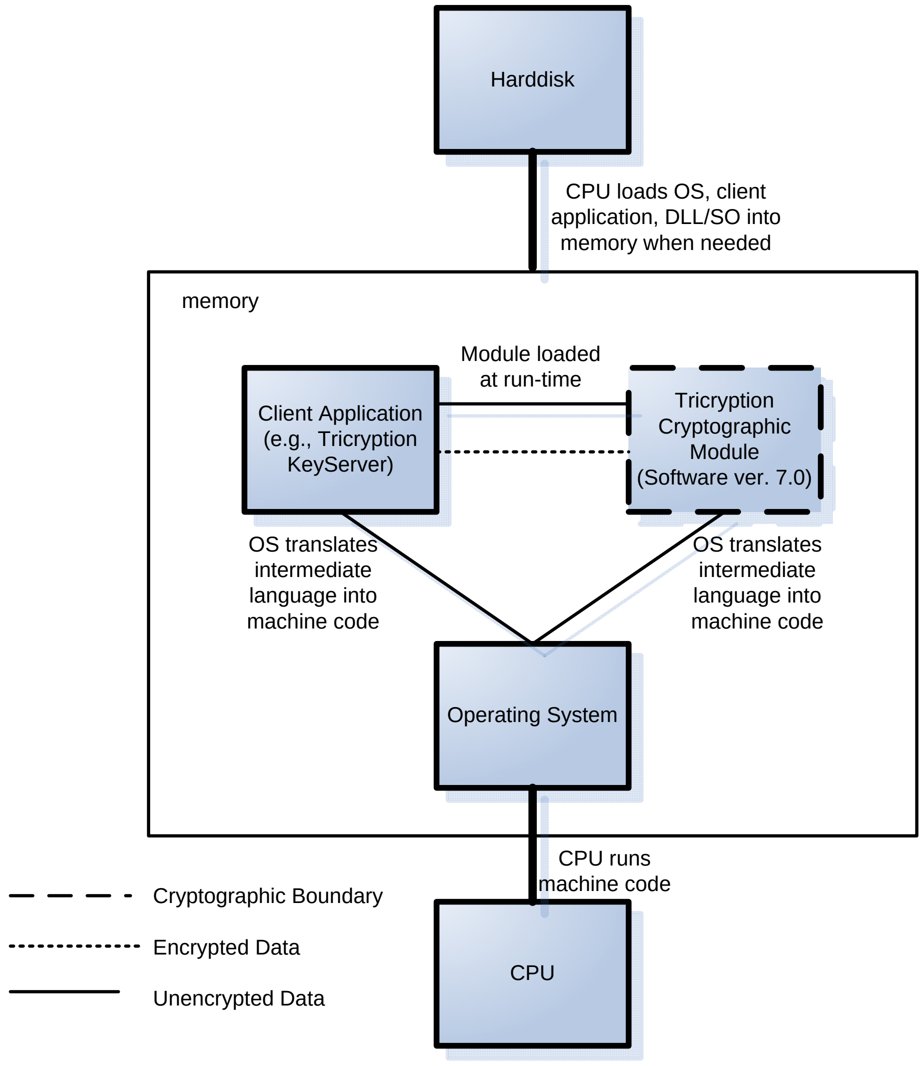

Harddisk
CPU loads OS, client application, DLL/SO into memory when needed
memory
Module loaded at run-time
Client Application (e.g., Tricryption KeyServer)
Tricryption Cryptographic Module (Software ver. 7.0)
OS translates intermediate language into machine code
OS translates intermediate language into machine code
Operating System
CPU runs machine code
CPU
Cryptographic Boundary
Encrypted Data
Unencrypted Data

Figure 2 – Logical Block Diagram

The module implements all cryptographic functions for the Tricryption KeyServer. Table 1 shows the operating systems, processor architectures, and file names of the module. The signature files store 20-byte keyed-Hash Message Authentication Code (HMAC) fingerprints of the binary files. The fingerprints are used in the software integrity test.

Table 1 – Binary Form of the Module

Operating System	Processor	Binary File Name	Signature File Name
Windows Server 2003 R2	Intel Pentium 4	tcm-fips.dll	tcm-fips.dll.hmac

Operating System	Processor	Binary File Name	Signature File Name
Red Hat Enterprise Linux 5	AMD Opteron	libtcm-fips.so	libtcm-fips.so.hmac

The module features only FIPS-approved and FIPS-allowed algorithms and always operates in an Approved mode of operation. Following is a list of FIPS-approved algorithms supported by the module when operating in a FIPS-approved mode of operation:

- AES encryption/decryption: 128, 192, and 256 bits, in ECB[4] and CBC[5] modes (certificate #796)
- Triple DES encryption/decryption: 112 and 168 bits, in ECB and CBC modes (certificate #684)
- SHA[6]-1, SHA-256, and SHA-384 (certificate #795)
- HMAC-SHA-1, HMAC- SHA-256, and HMAC-SHA-384 (certificate #437)
- ECDSA[7] key generation, signature generation, and signature verification: all P, K, and B curves (certificate #88)
- RSA[8] PKCS[9]#1 signature generation and verification: 1024, 2048, and 3072 bits (certificate #380)
- RNG[10]: ANSI[11] X9.31 Appendix A.2.4 RNG with 112-bit Triple DES (certificate #457)

The following functions implemented by the module are not FIPS-approved, but FIPS-allowed:

- Elliptic Curve Diffie-Hellman (ECDH) key agreement scheme: 1024 and 2048 bits for symmetric key establishment in TLS[12] sessions. The ECDH key agreement implementations support elliptic curve sizes up to 571 bits that provide 256 bits of encryption strength.
- RSA PKCS#1 v1.5 key wrap: 1024, 2048, and 3072 bits for symmetric key transport in TLS sessions. They provide 80, 112, and 128 bits of encryption strength, respectively.

The module is validated at FIPS 140-2 section levels shown in Table 2. Note that in Table 2, EMI and EMC mean Electromagnetic Interference and Electromagnetic Compatibility, respectively, and N/A indicates "Not Applicable".

Table 2 – Security Level per FIPS 140-2 Section

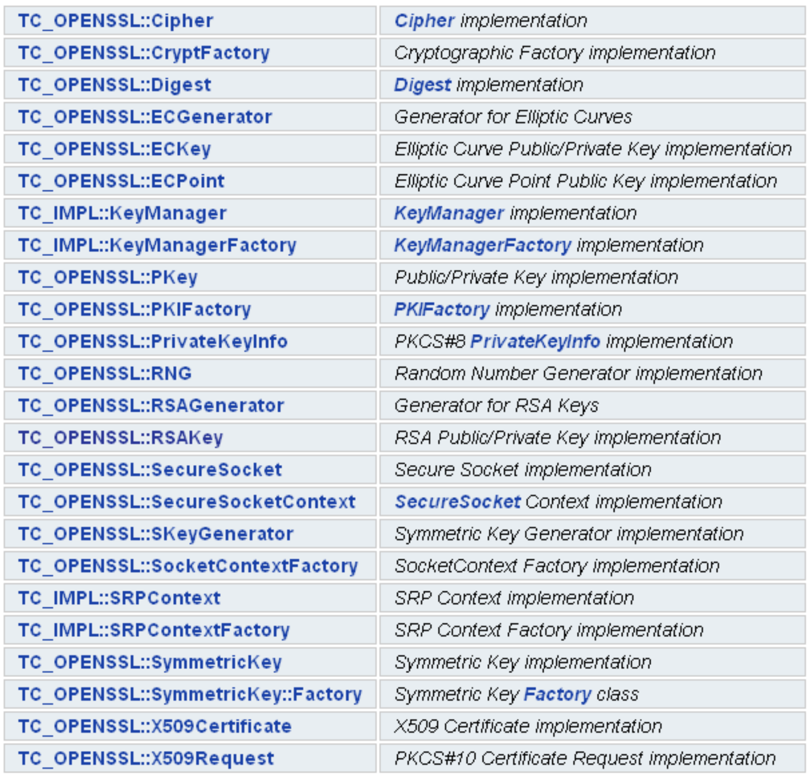
Section	Section Title	Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1
3	Roles, Services, and Authentication	1
4	Finite State Model	1
5	Physical Security	N/A
6	Operational Environment	1
7	Cryptographic Key Management	1
8	EMI/EMC	1

[4] Electronic Codebook
[5] Cipher Block Chaining
[6] Secure Hash Algorithm
[7] Elliptic Curve Digital Signature Algorithm
[8] Rivest, Shamir, and Adleman
[9] Public Key Cryptography Standard
[10] Random Number Generator
[11] American National Standards Institute
[12] Transport Layer Security

Section	Section Title	Level
9	Self-Tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A

2.2 Module Interfaces

See Figure 3 for a list of classes implemented by the Tricryption Cryptographic Module and their descriptions. The library module consists of 24 classes. Together, they provide a set of cryptographic services to ERUCES' client applications, such as the Tricryption KeyServer. SRP stands for the Secure Remote Password protocol developed by Stanford University. See http://srp.stanford.edu/ for details.



TC_OPENSSL::Cipher	*Cipher implementation*
TC_OPENSSL::CryptFactory	*Cryptographic Factory implementation*
TC_OPENSSL::Digest	*Digest implementation*
TC_OPENSSL::ECGenerator	*Generator for Elliptic Curves*
TC_OPENSSL::ECKey	*Elliptic Curve Public/Private Key implementation*
TC_OPENSSL::ECPoint	*Elliptic Curve Point Public Key implementation*
TC_IMPL::KeyManager	*KeyManager implementation*
TC_IMPL::KeyManagerFactory	*KeyManagerFactory implementation*
TC_OPENSSL::PKey	*Public/Private Key implementation*
TC_OPENSSL::PKIFactory	*PKIFactory implementation*
TC_OPENSSL::PrivateKeyInfo	*PKCS#8 PrivateKeyInfo implementation*
TC_OPENSSL::RNG	*Random Number Generator implementation*
TC_OPENSSL::RSAGenerator	*Generator for RSA Keys*
TC_OPENSSL::RSAKey	*RSA Public/Private Key implementation*
TC_OPENSSL::SecureSocket	*Secure Socket implementation*
TC_OPENSSL::SecureSocketContext	*SecureSocket Context implementation*
TC_OPENSSL::SKeyGenerator	*Symmetric Key Generator implementation*
TC_OPENSSL::SocketContextFactory	*SocketContext Factory implementation*
TC_IMPL::SRPContext	*SRP Context implementation*
TC_IMPL::SRPContextFactory	*SRP Context Factory implementation*
TC_OPENSSL::SymmetricKey	*Symmetric Key implementation*
TC_OPENSSL::SymmetricKey::Factory	*Symmetric Key Factory class*
TC_OPENSSL::X509Certificate	*X509 Certificate implementation*
TC_OPENSSL::X509Request	*PKCS#10 Certificate Request implementation*

Figure 3 – Classes in the Module

Figure 2 surrounds the classes listed above. The module's interactions with surrounding components, including the CPU, hard-disk, memory, application, and the Operating System (OS) are also demonstrated in Figure 2.

The module is validated for use on the platforms listed in the first column of Table 1. In addition to the binary, the physical device consists of the integrated circuits of the motherboard, the CPU, Random Access Memory (RAM), Read-Only Memory (ROM), computer case, keyboard, mouse, video interfaces, expansion cards, and other hardware components included in the computer such as hard disk, floppy disk, compact disc (CD) ROM drive, power supply, and fans. The physical cryptographic boundary of the module is the hard opaque metal and plastic enclosure of the server[13] running the module. The block diagram for a server is shown in Figure 4, with the physical cryptographic boundary denoted by the dashed line. Note that in this figure, I/O means Input/Output, BIOS stands

[13] The "server" here refers to device architecture and does not refer to the "Tricryption KeyServer".

for Basic Input/Output System, PCI stands for Peripheral Component Interconnect, ISA stands for Instruction Set Architecture, UART stands for Universal Asynchronous Receiver Transmitter, and IDE represents Integrated Drive Electronics.

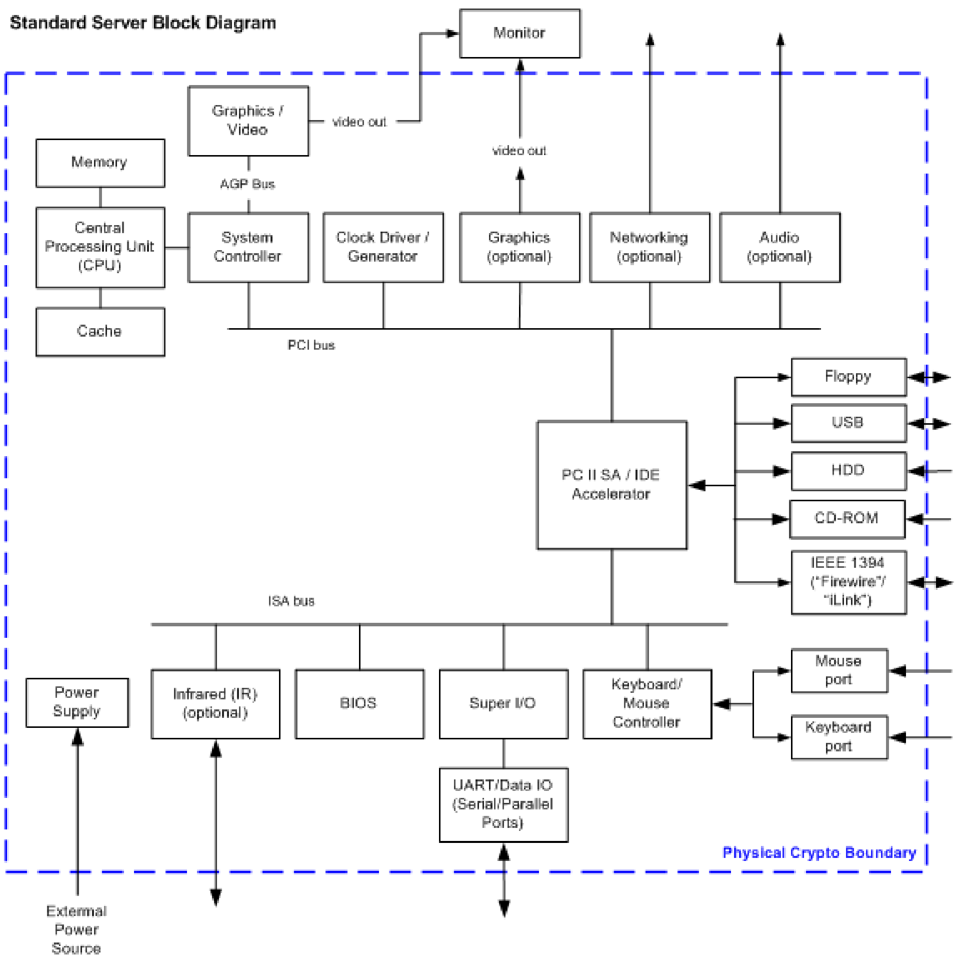


Figure 4 – Physical Block Diagram of a Typical Server

All of these physical ports are separated into logical interfaces defined by FIPS 140-2, as described in Table 3.

Table 3 – Logical Interface, Physical Port, and Module Mapping

Logical Interface	Physical Port Mapping	Module Mapping
Data input	Keyboard, mouse, CD-ROM, floppy disk, and serial/USB/parallel/network ports	Arguments for Application Programming Interface (API) calls that contain data to be used or processed by the module
Data output	Hard disk, floppy disk, monitor, and serial/USB/parallel/network ports	Arguments for API calls that contain module response data to be used or processed by the caller
Control input	Keyboard, CD-ROM, floppy disk, mouse, and serial/USB/parallel/network port	API function calls
Status output	Hard disk, floppy disk, monitor, and serial/USB/parallel/network ports	Arguments for API calls, function return values, error messages

2.3 Roles and Services

The module supports one Crypto-Officer role and one User role. The Crypto-Officer is responsible for the secure installation and initialization of the module, as well as performing key management functions. The User role has access to the module's cryptographic services offered. All roles are implicitly assumed.

As shown in Table 3, the module provides services through its 24 classes. Services from eight classes have been assigned to the Crypto-Officer role and services from the remaining 16 classes have been assigned to the User role. See Table 4 for assignment details.

Table 4 – Roles and Services

Role	Services
Crypto-Officer	Install Module Initialize Module Show Status Execute Self-Tests Zeroize CSPs TC_OPENSSL::CryptFactory TC_OPENSSL::ECGenerator TC_IMPL::KeyManager TC_IMPL::KeyManagerFactory TC_OPENSSL::PKIFactory TC_OPENSSL::RNG TC_OPENSSL::RSAGenerator TC_OPENSSL::SkeyGenerator
User	Initialize Module Show Status Execute Self-Tests Zeroize CSPs TC_OPENSSL::Cipher TC_OPENSSL::Digest TC_OPENSSL::ECKey TC_OPENSSL::ECPoint TC_OPENSSL::Pkey TC_OPENSSL::PrivateKeyInfo TC_OPENSSL::RSAKey TC_OPENSSL::SecureSocket TC_OPENSSL::SecureSocketContext TC_OPENSSL::SocketContextFactory TC_IMPL::SRPContext TC_IMPL::SRPContextFactory TC_OPENSSL::SymmetricKey TC_OPENSSL::SymmetricKey::Factory TC_OPENSSL::X509Certificate TC_OPENSSL::X509Request

Table 5 summarizes services for the Crypto-Officer role. The class names are shown in the first column ("Service"), and the corresponding functions are described in the second column ("Description"). Values given in the rightmost columns correspond to the values listed in Table 7.

Table 5 – Crypto-Officer Services

Service	Description	Input	Output	CSP
Install Module	Install the module on the target platform	Service command	None	None

Service	Description	Input	Output	CSP
Initialize Module	Configure the module	Service command	None	None
Show Status	Output the current status of the module	Function call	Status	None
Execute Self-Tests	Perform module power-up self-tests	Function call	Status	None
Zeroize CSPs	Zeroize plaintext secret and private keys	Function call	Status	All plaintext secret and private CSPs - write
TC_OPENSSL::CryptFactory	Create objects for symmetric and asymmetric cipher and key generation	Function call	Status	None
TC_OPENSSL::ECGenerator	Generate keypairs for ECDSA	Function call, curve name	Status, ECDSA keypair	publicKey – write privKey – write
TC_IMPL::KeyManager	Create master keys	Function call	Status, master key	masterKey – write
	Export master keys	Function call, master key	Status	masterKey – write
	Create system keys	Function call	Status, system key	systemKey – write
	Add system keys from database	Function call, system key	Status	systemKey – write
	Create encryption keys	Function call, algorithm name	Status, encryption key	encryptionKey – write
	Encrypt encryption keys	Function call, encryption key	Status, encrypted encryption key	encryptionKey – read systemKey – read
	Decrypt encryption keys	Function call, encrypted encryption key	Status, decrypted encryption key	encryptionKey – read systemKey – read
	Create keylinks	Function call	Status, unencrypted keylink, encrypted keylink	systemKey – write
	Encrypt keylinks	Function call, keylink	Status, encrypted keylink	systemKey – read
	Decrypt keylinks	Function call, encrypted keylink	Status, decrypted keylink	systemKey – read
TC_IMPL::KeyManagerFactory	Create objects for KeyManager	Function call, algorithm name	Status	None
TC_OPENSSL::PKIFactory	Create, initialize, and terminate objects for X509Certificate and X509Request	Function call	Status	None

Service	Description	Input	Output	CSP
TC_OPENSSL::RNG	Generate random numbers	Function call, seed, number of bits	Status, random number	RNGseed – write, delete
TC_OPENSSL::RSAGenerator	Generate keypairs for RSA	Function call, number of bits	Status, RSA keypair	publicKey – write privKey – write
TC_OPENSSL::SkeyGenerator	Generate keys and initialization vectors (IVs) for AES and Triple DES	Function call, algorithm name, number of bits	Status, symmetric key, IV	symmKey – write

Table 6 summarizes services for the User role. Values given in the rightmost columns correspond to the values listed in Table 7. Notice that the module does not authenticate the User or the Crypto-Officer. The "password" variables listed in the rightmost column are not used to authenticate the User or the Crypto-Officer. They refer to the credentials used by the client applications to verify identities of users of the client application.

Table 6 – User Services

Service	Description	Input	Output	CSP
Initialize Module	Configure the module	Service command	None	None
Show Status	Output the current status of the module	Function call	Status	None
Execute Self-Tests	Perform module power-up self-tests	Function call	Status	None
Zeroize CSPs	Zeroize plaintext secret and private keys	Function call	Status	All plaintext secret and private CSPs - write
TC_OPENSSL::Cipher	Perform symmetric encryption	Function call, algorithm, symmetric key, IV, plaintext	Status, ciphertext	symmKey – read
	Perform symmetric decryption	Function call, algorithm, symmetric key, IV, ciphertext	Status, plaintext	symmKey – read
TC_OPENSSL::Digest	Perform HMAC hashing	Function call, HMAC key, massage	Status, HMAC digest	HMACkey – read
TC_OPENSSL::ECKey	Compute TLS master secret with peer ECDH public key	Function call, peer ECDH public key	Status, TLS master secret	TLSMS – write TLSecdhKey – read
TC_OPENSSL::ECPoint	Downcast abstract pointer to concrete PKey class	Function call, pointer to an elliptic curve point	Status	None

Service	Description	Input	Output	CSP
TC_OPENSSL::Pkey	Perform digital signature generation	Function call, private key, message	Status, signature	privKey – read
	Perform digital signature verification	Function call, public key, message, signature	Status, verification result	publicKey – read
TC_OPENSSL::PrivateKeyInfo	Retrieve private key attributes	Function call, private key	Status, attributes	privKey – read
TC_OPENSSL::RSAKey	Encrypt data using RSA	Function call, RSA public key, plaintext data	Status, encrypted data	publicKey – read
	Decrypt data using RSA	Function call, RSA private key, encrypted data	Status, decrypted data	privKey – read
TC_OPENSSL::SecureSocket	Establish TLS sessions	Function call, X.509 certificates	Status	TLSasymmKey – read TLSecdhKey – write, read, delete TLSMS – write, read, delete TLSsessionKey – write, read, delete
TC_OPENSSL::SecureSocketContext	Initialize TLS contexts	Function call, X.509 certificates	Status	None
TC_OPENSSL::SocketContextFactory	Create objects for SecureSocketContext	Function call	Status	None
TC_IMPL::SRPContext	Set SRP passwords	Function call, username, password	Status	password – write
	Perform SRP authentication on behalf of the caller	Function call, username, password	Status	password – read
TC_IMPL::SRPContextFactory	Create objects for SRPContext	Function call	Status	None
TC_OPENSSL::SymmetricKey	Encapsulates Cipher objects	See entry for the Cipher class	See entry for the Cipher class	See entry for the Cipher class
TC_OPENSSL::SymmetricKey::Factory	Create objects for SymmetricKey	Function call	Status	None
TC_OPENSSL::X509Certificate	Set attributes for X.509 certificates	Function call, X.509 certificate attributes, owner public key	Status	publicKey – read

Service	Description	Input	Output	CSP
	Get attributes from X.509 certificates	Function call, X.509 certificate	Status, X.509 certificate attributes, owner public key	publicKey – write
TC_OPENSSL::X509Request	Sign a X.509 certificate request	Function call, owner public key, CA private key	Status, signed X.509 certificate	publicKey – read CAprivKey – read

2.4 Physical Security

The Tricryption Cryptographic Module is a multi-chip standalone module. The physical security requirements do not apply to this module, since it is purely a software module and does not implement any physical security mechanisms. Although the module is software, the FIPS 140-2 platform is a server that has been tested for and meets applicable Federal Communications Commission (FCC) EMI and EMC requirements for business use as defined in Subpart B of FCC Part 15.

2.5 Operational Environment

The module was tested and validated on general-purpose Microsoft Windows Server 2003 R2 on Intel Pentium 4 and Red Hat Enterprise Linux 5 on AMD Opteron. The operating system running the module must be configured in single user mode as per the instructions provided in Section 3.1 of this document.

2.6 Cryptographic Key Management

The module supports the following CSPs:

Table 7 – List of CSPs

CSP Type	Generation / Input	Storage	Zeroization	Use
AES key	Generated by ANSI X9.31 RNG or input by caller	Plaintext in volatile memory	Upon return of the API call	• Encrypt/decrypt operations
Triple DES key	Generated by ANSI X9.31 RNG or input by caller	Plaintext in volatile memory	Upon return of the API call	• Encrypt/decrypt operations
RSA keys	Generated by ANSI X9.31 RNG or input by caller	Plaintext in volatile memory	Upon return of the API call	• Encrypt/decrypt operations • Key wrapping • Signature generation/verification operations
ECDSA keys	Generated by ANSI X9.31 RNG or input by caller	Plaintext in volatile memory	Upon return of the API call	• Encrypt/decrypt operations • Key wrapping • Signature generation/verification operations

CSP Type	Generation / Input	Storage	Zeroization	Use
ANSI X9.31 RNG Seed	Gathered internally	Plaintext in volatile memory	Upon return of the API call	• ANSI X9.31 RNG seeding material
ANSI X9.31 RNG Seed Key	Gathered internally	Plaintext in volatile memory	Upon return of the API call	• ANSI X9.31 RNG seeding material
HMAC key	Generated by ANSI X9.31 RNG or input by caller	Plaintext in volatile memory	Upon return of the API call	• Keyed hashing operations • Software Integrity Test
Elliptical Curve Diffie-Hellman keys	Generated by ANSI X9.31 RNG	Plaintext in volatile memory	Upon session termination	• Negotiate TLS Master Secret

2.6.1 Key Generation

The module uses an ANSI X9.31 RNG with 2-key Triple DES to generate cryptographic keys. This RNG is a FIPS 140-2 approved RNG as specified in Annex C to FIPS PUB 140-2.

2.6.2 Key Input and Output

The cryptographic module itself does not support key entry or key output; however keys can be passed to the module as parameters from the application via the APIs.

2.6.3 CSP Storage

All CSPs are stored in volatile memory in plaintext.

2.6.4 CSP Zeroization

All CSPs are stored in volatile memory in plaintext. CSPs are zeroized when they are no longer needed. i.e., CSPs are zeroized by a C++ destruct routine. The module itself does not permanently store any CSPs.

2.7 Self-Tests

The module performs the following power-up self-tests:

- Software integrity test using HMAC-SHA1
- Known Answer Test (KAT) for Triple DES encryption and decryption
- KAT for AES encryption and decryption
- KATs for HMAC-SHA message authentication
- KATs for SHA hashing (tested as part of the HMAC-SHA KATs)
- KAT for RSA signature generation and verification
- KAT for RSA encryption and decryption
- KAT for ANSI X9.31 RNG
- Pair-wise consistency test for ECDSA signature generation and verification

Note that these power-up self-tests can also be executed on-demand at any time by re-instantiation of the module from the hosting application.

The module implements the following conditional self-tests:

- Pair-wise consistency test for RSA signature generation and verification, performed when a new RSA keypair is generated.
- Pair-wise consistency test for ECDSA signature generation and verification, performed when a new ECDSA keypair is generated.

- Continuous RNG test, performed when a new random number is generated.

If a self-test fails, an exception will be thrown on the failure. The application is then alerted that a self-test has failed. The module will enter an error state and be unloaded.

2.8 Mitigation of Other Attacks

The module does not claim to mitigate any attacks beyond the FIPS 140-2 level 1 requirements for this validation.

3 Secure Operation

The operator is responsible for installing, uninstalling, configuring, and managing the module and running the power-up self-tests when necessary. Before installing the module, the operator should make sure that the specific operating system is in single user mode.

3.1 Operating System Configuration

The user of the module is a software application. FIPS 140-2 mandates that a cryptographic module be limited to a single user at a time. To meet this requirement, a single instantiation of an application must only access a single instantiation of the Tricryption Cryptographic Module.

For enhanced security, it is recommended that the operator configure the operating system to disallow remote login. See the following instructions for details.

Windows Server 2003 R2:

To configure Windows Server 2003 R2 to disallow remote login, the operator should ensure that all remote guest accounts are disabled in order to ensure that only one human operator can log into Windows at a time. The services that need to be turned off for Windows are:

- Fast-user switching (irrelevant if server is a domain member)
- Terminal services
- Remote registry service
- Secondary logon service
- Telnet service
- Remote desktop and remote assistance service

Once the Windows operating system has been configured to disable remote login, the operator can use the system "Administrator" account to install software, uninstall software, and administer the module.

Red Hat Enterprise Linux 5:

The specific procedure to configure a Linux system for single user mode is described below.

1. Login as the "root" user.
2. Edit the system files /etc/passwd and /etc/shadow, and remove all the users except "root" and the pseudo-users. Make sure the password fields in /etc/shadow for the pseudo-users are either a star (*) or double exclamation mark (!!). This prevents login as the pseudo-users.
3. Edit the system file /etc/nsswitch.conf and make "files" the only option for "passwd", "group", and "shadow". This disables Network Information Service and other name services for users and groups.
4. In the /etc/xinetd.d directory, edit the files "rexec", "rlogin", "rsh", "rsync", "telnet", and "wu-ftpd", and set the value of "disable" to "yes".
5. Reboot the system for the changes to take effect.

Once the Linux operating system has been configured to disable remote login, the operator can use the system "root" account to install software, uninstall software, and administer the module.

3.2 Installation

The following sections provide guidance to ensure that the module is installed securely and executing in an FIPS-Approved mode of operation on the chosen platform.

3.2.1 Installation on Linux Platforms

1. Install the module using the vendor-provided binary executable installer. This will copy the module file "libTCM_FIPS.so" and the corresponding HMAC value file "libTCM_FIPS.hmac" to the appropriate location on the target machine.

2. Select the chosen ERUCES application that will work with the module. (Please refer to the appropriate user guide for the ERUCES application being installed for further details.)

3. Restart the machine, and verify that the module is running correctly by executing the script "tcm_fips_status", which will report "FIPS MODULE: OK" when the module has been invoked successfully.

3.2.2 Installation on Windows Platforms

1. Install the module using the vendor-provided binary executable installer. A Welcome window will appear advising you to exit all programs.

2. Exit the programs as instructed and select the Next button.

3. A window will appear containing the Tricryption FIPS Module End User License Agreement. Read the license agreement, select the "I accept the terms of the License Agreement" option and select the **Next** button.

4. Follow the instructions provided to choose a destination folder for the Tricryption FIPS Module Shared Libraries folder and select the **Next** button.

5. Choose a destination folder for the Tricryption FIPS Module Shared Library Test Application and select the **Next** button.

6. When the installation is completed, select the **Finish** button.

7. To verify that the module was installed correctly, select FIPS Test Application from the ***Start*** > ***All Programs*** > ***ERUCES*** > ***Tricryption Suite*** > ***FIPS Test Application*** program folder. This will launch the program "cmd_fips_status.exe", which will report "FIPS MODULE: OK" when the module has been invoked successfully.

3.3 Initialization

The Tricryption Cryptographic Module itself is not an end-user product. It is provided to the end-users as part of other associated ERUCES applications. As such, the required initialization procedures are described in the installation manual for the appropriate application.

3.4 Authentication

This module does not provide authentication.

3.5 Status

The module reports its working status to the client application through return values of function calls. Additionally, operators can ascertain the status of the module by performing step 3 of the installation instructions listed in Section 3.2 above.

3.6 Management

The operator does not perform any management of the module after installation and configuration. The module’s cryptographic functionality and security services are provided via the Tricryption KeyServer application. The management tasks are conducted by the application. End-users are not supposed to utilize the module without the Tricryption KeyServer application. Only the algorithms listed in Section 2.1 should be invoked by the application. End-user instructions and guidance are provided in the user manual and technical support documents of the application software.

4 Acronyms

Table 8 – Acronyms

Acronym	Definition
AES	Advanced Encryption Standard
ANSI	American National Standards Institute
API	Application Programming Interface
BIOS	Basic Input/Output System
CBC	Cipher Block Chaining
CD	Compact Disc
CMVP	Cryptographic Module Validation Program
CPU	Central Processing Unit
CSEC	Communications Security Establishment Canada
CSP	Cryptographic Server Provider
CSP	Critical Security Parameter
DES	Data Encryption Standard
DLL	Dynamic Link Library
ECB	Electronic Codebook
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FCC	Federal Communications Commission
FIPS	Federal Information Processing Standard
HDD	Hard Disk Drive
HMAC	(keyed-) Hash Message Authentication Code
KAT	Known Answer Test
IDE	Integrated Drive Electronics
IEEE	Institute of Electrical and Electronics Engineers
I/O	Input/Output
IR	Infrared
ISA	Instruction Set Architecture
IT	Information Technology
IV	Initialization Vector
N/A	Not Applicable
NIST	National Institute of Standards and Technology
OS	Operating System
PCI	Peripheral Component Interconnect

Acronym	Definition
PKCS	Public Key Cryptography Standard
RAM	Random Access Memory
RNG	Random Number Generator
ROM	Read Only Memory
RSA	Rivest, Shamir, and Adleman
PCI	Peripheral Component Interconnect
SHA	Secure Hash Algorithm
SO	Shared Object
SRP	Secure Remote Password protocol
TLS	Transport Layer Security
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus