TriCipher, Inc.
TriCipher Common Crypto Core Module

(Software Version: 3.9)

FIPS 140-2
Non-Proprietary Security Policy

Level 1 Validation

Document Version 0.4

Prepared for:

TriCipher, Inc.
12007 Sunrise Valley Drive
Suite 355
Reston, VA 20191
Phone: (650) 372-1300
Fax: (650) 372-1301
http://www.tricipher.com

Prepared by:

Corsec Security, Inc.

10340 Democracy Lane, Suite 201
Fairfax, VA 22030

Phone: (703) 267-6050
Fax: (703) 267-6810
http://www.corsec.com

Revision History

Version	Modification Date	Modified By	Description of Changes
0.1	2006-08-09	Cas Stulberger	Initial draft
0.2	2007-06-15	Darryl H. Johnson	Modfied to address lab comments.
0.3	2007-07-11	Darryl H. Johnson	Defined the module boundary as only the Common Crypto Core Module of the Crypto Core Library
0.4	2007-12-04	Darryl H. Johnson	Addressed editorial changes per CMVP comment

Table of Contents

Table of Figures

Table of Tables

1 Introduction

1.1 Purpose

This is a non-proprietary Cryptographic Module Security Policy for the TriCipher Common Crypto Core Module from TriCipher, Inc.. This Security Policy describes how the TriCipher Common Crypto Core Module (CCCM) 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 TriCipher Common Crypto Core Module is referred to in this document as the Common Core Crypto Module, CCCM, or 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 TriCipher website (http://www.tricipher.com) contains information on the full line of products from TriCipher.
- The CMVP website (http://csrc.nist.gov/cryptval/) contains contact information for answers to technical 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 document
- Finite State Machine
- Other supporting documentation as additional references

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

2 TriCipher Common Crypto Core Module

2.1 Overview

The TriCipher Common Core Library (CCL) provides an interface to client developers for crypto programming and supports TriCipher's Client software during normal crypto operations. The crypto classes of the CCL are separated from the other utility functions and form a new module called Common Crypto Core Module, or CCCM. The CCCM provides an interface to a user for crypto programming and supports the client software doing normal crypto operations. The module defines a fixed logical boundary in the following forms:

- Module Interface that allows the upper level Application Programing Interfaces (APIs) to call out a crypto operation.
- Services and Roles that define the provided crypto services and roles that can operate the services.

The module is implemented in a library form. On the Windows platform it takes on the embodiment of a dynamically linked library (DLL) and exports the necessary crypto APIs to the upper layer components that are ported to other kinds of platforms. The module can also be compiled under the Sun Java Desktop System (JDS) Linux 2.4.19 operating system.

The TriCipher Common Crypto Core Module is a multi-chip standalone module validated at the following FIPS 140-2 Section levels:

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

2.2 Module Interfaces

The CCCM is a software module that meets overall level 1 FIPS 140-2 requirements. The logical cryptographic boundary of the module consists of the CCCM software and was tested on Windows XP or Sun JDS Linux 2.4.19. The module is composed of a single binary file, which is cross-compiled on the specified operating systems in both User and Kernel-modes. The table below summarizes binary forms running for different OS.

Table 2 – Binary Forms of the Module

Operating System	Binary File Name

Operating System	Binary File Name
Windows XP	CryptoDLL.dll
Sun JDS Linux System	libtrccryptom.so

Figure 1 below illustrates the logical block diagram of the CCL, as well as the logical crypto boundary of the CCCM.

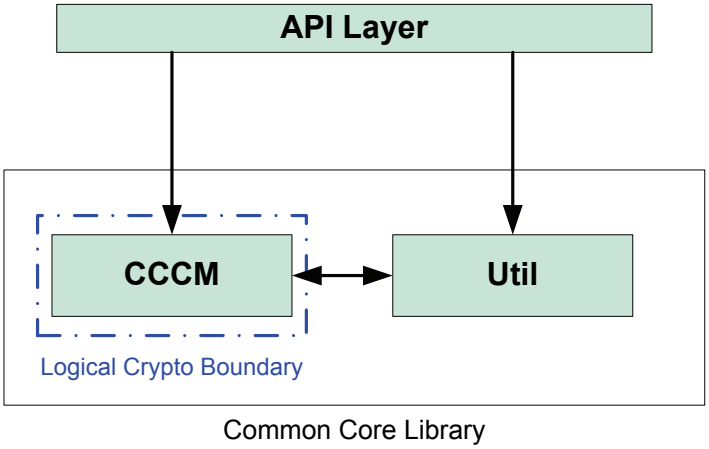

Figure 1 – Module Logical Block Diagram

The module is validated for use on a general purpose computer (GPC) running Windows XP or Sun JDS Linux 2.4.19. In addition to the binaries, the physical device consists of the integrated circuits of the motherboard, the central processing unit (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, CD-ROM drive, power supply, and fans. The physical cryptographic boundary of the module is the hard opaque metal and plastic enclosure of the computer. The block diagram for a standard personal computer (PC) is shown below in Figure 2.

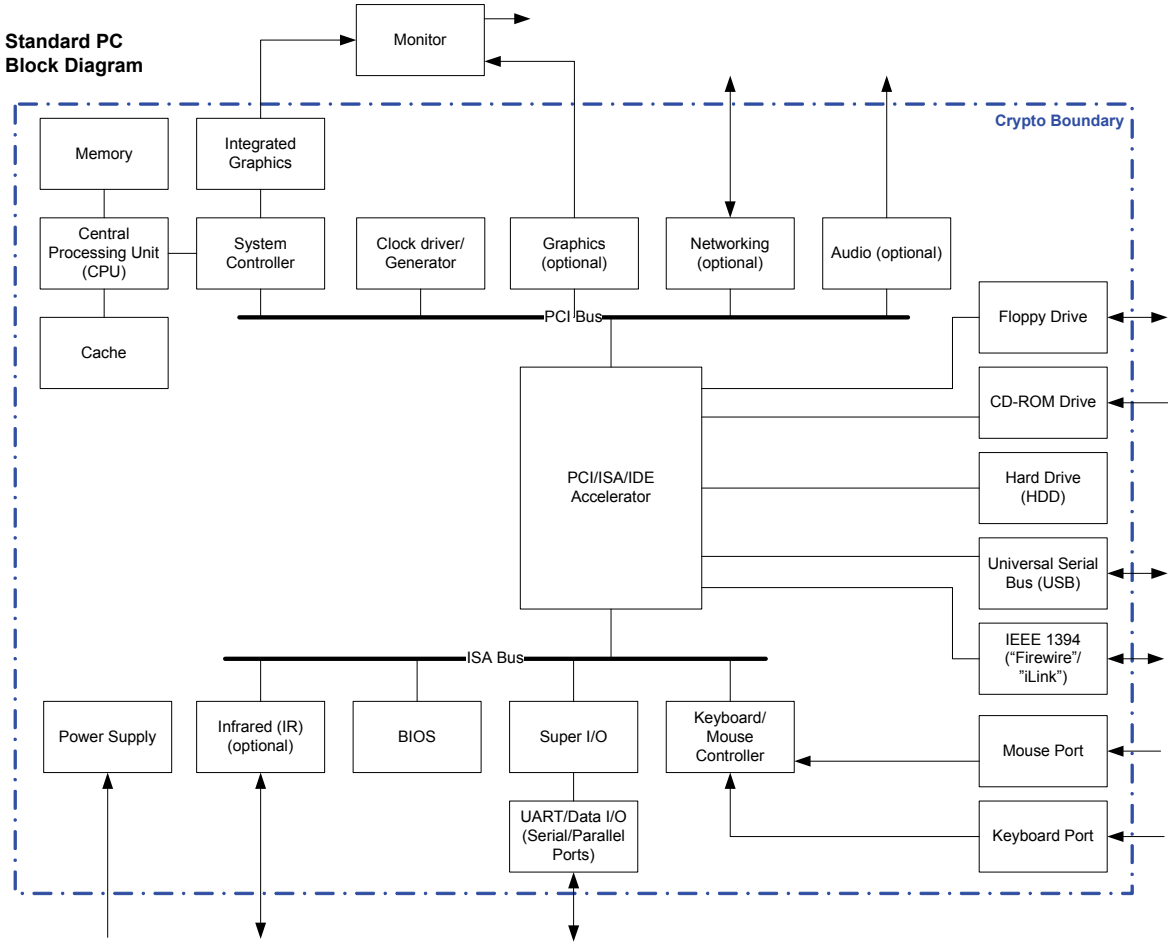

Figure 2 – Standard PC Physical Block Diagram

All of these physical interfaces are separated into logical interfaces defined by FIPS 140-2, as described in the following table:

Table 3 – FIPS 140-2 Logical Interfaces

FIPS 140-2 Logical Interface	Module Port/Interface	Module Mapping
Data Input 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 Output Interface	Hard Disk, floppy disk, monitor, and serial/USB/parallel/network ports	Arguments for a function that specify where the result of the function is stored.
Control Input 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.
Status Output Interface	Hard Disk, floppy disk, monitor, and serial/USB/parallel/network ports	Return values for function calls.
PC Power Interface	Power Interface	Not Applicable

2.3 Roles and Services

There are two roles in the module (as required by FIPS 140-2) that operators may implicitly assume: a Crypto Officer role and a User role. The operator of the module assumes either of the roles based on the operations performed without any authentication. Both of the roles and their responsibilities are described below.

2.3.1 Crypto Officer Role

The Crypto Officer role has the ability to install and configure the module, initiate and run the power-up self-tests, and output the current system status. Descriptions of the services available to the Crypto Officer role are provided in the table below.

Table 4 – Mapping of Crypto Officer Role's Services to Inputs, Outputs, CSPs, and Type of Access

Service	Description	Input	Output	CSP and Type of Access
Install module	Installs and configures the dll according to the operating system	PRNG Seed, HMAC-SHA-1 key, Triple-DES key, RSA key material	.module installed	All - Read, Write
Initiate Self Tests	The CO can intitate self-tests at will be restarting the CCL	Command	Status Output information regarding self-tests	None
Show status	The CO can view system status messages via the Status Output interface	Commands	Status output information regarding system status	None

2.3.2 User Role

The User role accesses the module's cryptographic services that include encryption/decryption and authentication functionalities. Descriptions of the services available to the User role are provided in the table below.

Table 5 – Mapping of User Role's Services to Inputs, Outputs, CSPs, and Type of Access

Service	Description	Input	Output	CSP and Type of Access
SHA1.hash	Performs a SHA1 hash of the data	Data	Hashed data	SHA1 - Read/Write
Res.encrypt	Encrypts the TimeStamp, sequence Number, and a confounder using Triple-DES	Key	Encrypted TimeStamp, sequence Number, and a confounder	Triple-DES Key - Read
Res.verify	Checks if the seqNum = seqNum+1	Key, Encrypted Bytestream, Timestamp	True or False	Triple-DES Key - Read
PKCS1blockFormat.encodeForPrivKeyOp	Encodes the data according to Public Key Cryptography Standards (PKCS) #1	Bytestream, modulus Length, encrypted data bytestream	Encoded data	PKCS1 - Read
PKCS1blockFormat.decodeForPrivKeyOp	Decode the data encoded for Private key operation according to the PKCS1 standards	Bytestream, modulus Length, encrypted data bytestream	Decoded data	PKCS1 - Read
PKIData.signRSA	RSA signature	Modulus, Private Key Exponent	RSA Signature	RSA Key - Read
PKIData.verifyRSA	RSA Signature verification; can be used for RSA signatures generated directly using the RSA private key	Modulus, Public Key Exponent	True or False	RSA Key - Read
PKIData.modExp	Do modular exponentiation of the data	key Exponent, key Modulus, data, result	Modular exponentiation of the data	Key components - Read
SymKeyRing.encrypt	Continues encrypting a ByteStream	Key, IV, Bytestream data	Encrypted data	RSA key pair - Read
SymKeyRing.decrypt	Continues decrypting a ByteStream	Key, IV, Encrypted data	Decrypted data	RSA key pair - Read
SymKeyRing.encryptChunk	Encrypts of a ByteStream and deletes the IV if final is true	Bytestream, final Boolean value	Encrypted stream.	RSA key pair - Read
SymKeyRing.decryptChunk	Decrypts a ByteStream and deletes the IV if final is true	Encrypted Stream, and final Boolean value	Decrypted stream	RSA key pair - Read

Service	Description	Input	Output	CSP and Type of Access
SymKeyRing.hmac	Calls and returns a HMAC of data	Key, data, length	HMAC of the data	HMAC - Read
SymKeyRing.verifyHmac	Verifies HMAC	Data, length, hmac	True or False	HMAC - Read
SymKeyRing.keyDerive	Used to generate the inbound and outbound encryption and HMAC keys within the SymKeyRing, using R12 as the input key	Key, data, length	Key	HMAC Key - Read/Write
SymKeyRing.RANDbyteStream	Used for FIPS186-2 RNG for deriving FIPS-compliant Triple-DES keys	Seed, random bytestream, random length	0 or random bytestream	Triple-DES keys - Write
SymKeyRing.exportKeys	Exports keys	Keys	none	Symmetric Keys - Read
SymKeyRing.importKeys	Gets new symmetric keys	Keys	none	Symmetric Keys - Write
KeyController.genPKCS10CertReq	Generates a Private Key Certificate Request	Private Key, Salt, Counter, Alg String	Private Key Certificate Request	RSA Key - Read
KeyController.genRSAKey	Generate the private RSA key and return it	Key size	RSA private key	RSA key pair - Write
CRController.genPKCS10CertReq	Generate the BER version of the PKCS#10 Certificate request	subject's DN, private key, salt, iterCount and message digest algorithm	BER version of the PKCS#10 Certificate request	None
CRController.addCustomExtension	Adds information to the Certificate	String, value	None	None
CRController.setIssuerCert	Sets the Certificate Issuer	Certificate	True or False	None
CRController.setReqVersion	Sets the Request Version	Version	None	None
openSSLKShelper.hashAndSign	Hashes and signs the data	Bytestream	Hashed and signed bytestream	HMAC - Write RSA Key - Read
openSSLKShelper.encrypt	Encrypts data using the RSA public key and zeroizes the key	Bytestream	Encrypted bytestream	RSA key pair - Read

Service	Description	Input	Output	CSP and Type of Access
openSSLKShelper.decrypt	Decrypts a bytestream using the RSA private key and zeroizes the key	Encrypted bytestream	Bytestream	RSA key pair - Read
openSSLKShelper.simpleEncrypt	Encrypts data using the RSA public key and does not zeroize the key	Bytestream	Encrypted bytestream	RSA key pair - Read
openSSLKShelper.simpleDecrypt	Dencrypts data using the RSA public key and does not zeroize the key	Encrypted bytestream	Bytestream	RSA key pair - Read
openSSLKShelper.getKeyPair	Gets the parts used to calculate the RSA key	Boolean	Ex, Nx, getDx, Dx, Px, Qx, DmP1x, DmQ1x, IQmPx	RSA components - Read
openSSLKShelper.setKeyPair	Sets the parts used to calculate the RSA key	Ex, Nx, Dx, Px, Qx, DmP1x, DmQ1x, IqmPx	none	RSA components - Write
openSSLKShelper.generateKeyPair	Generates a RSA key pair	Key size	none	RSA key pair - Write
WindowsKShelper.hashAndSign	Hashes and signs the data	ByteStream	Hashed and Signed Bytestream	HMAC - Write RSA Key - Read
WindowsKShelper.encrypt	Encrypts data using the RSA public key and zeroizes the key	Bytestream	Encrypted bytestream	RSA key pair - Read
WindowsKShelper.decrypt	Decrypts a bytestream using the RSA private key and zeroizes the key	Encrypted bytestream	Bytestream	RSA key pair - Read
WindowsKShelper.simpleEncrypt	Encrypts data using the RSA public key and does not zeroize the key	Bytestream	Encrypted bytestream	RSA key pair - Read
WindowsKShelper.simpleDecrypt	Dencrypts data using the RSA public key and does not zeroize the key	Encrypted bytestream	Bytestream	RSA key pair - Read
WindowsKShelper.getKeyPair	Gets the parts used to calculate the RSA key	Boolean	Ex, Nx, getDx, Dx, Px, Qx, DmP1x, DmQ1x, IqmPx	RSA components - Read

Service	Description	Input	Output	CSP and Type of Access
WindowsKShelper.setKeyPair	Sets the parts used to calculate the RSA key	Ex, Nx, Dx, Px, Qx, DmP1x, DmQ1x, IqmPx	none	RSA components - Write
WindowsKShelper.generateKeyPair	Generates a RSA key pair	Key size	none	RSA key pair - Write
Random.random_seed	Creates a random seed	Seed	none	Seed - Write
Random.random	Provides OpenSSL with a seed	Length	bytestream	None
integrityCheck	Does an Integrity check of the .dll	None	Pass or Fail	All

2.3.3 Physical Security

The CCCM is a multi-chip standalone module, and is purely a software module. The physical security requirements do not apply to this module, since it is a software module and does not implement any physical security mechanisms.

Although the module consists entirely of software, the FIPS 140-2 evaluated platform is a GPC that has been tested for and meets applicable Federal Communication Commission (FCC) Electromagnetic Interference (EMI) and Electromagnetic Compatibility (EMC) requirements for business use as defined in Subpart B of FCC Part 15.

2.3.4 Operational Environment

The module runs on the general purpose Windows XP and Sun JDS Linux 2.4.19 operating systems. Linux and Windows XP must be configured for single-user mode per NIST CMVP guidance. The module was tested on Windows XP and Sun JDS Linux 2.4.19. Single-user mode configuration instructions for Linux and Windows XP can be found in the *Secure Operation* section.

2.3.5 Cryptographic Key Management

The CCCM implements the following FIPS-approved algorithms:

- Deterministic Random Number Generator (RNG) – FIPS 186-2 (certificate #341)
- HMAC SHA-1 – FIPS 198 (certificate #310)
- RSA (1024-bit) – PKCS #1 (certificate #273)
- SHA-1 – FIPS 180-2 (certificate #649)
- Triple-DES CBC mode – FIPS 46-3 (certificate #571)

Additionally, the module utilizes the following non-FIPS-approved algorithm implementation:

- DES
- Message Digest 5 (MD5)
- Non Approved RNG
- RSA (PKCS #5)

The module supports the following critical security parameters:

Table 6 – List of Cryptographic Keys, Cryptographic Key Components, and CSPs

Key	Key Type	Generation / Input	Output	Storage	Zeroization	Use
BER RSA Key	OpenSSL RSA Key	Input: RSA Private Key In BER format. Generation: use the constructor KeyController:: KeyController	Never output	Volatile memory only	Zeroized when no longer used	Dx, Nx and Ex from the BER format of the RSA key.
Unix RSA Key	CryptoAPI RSA Key	Input: Key size. Generation: OpenSslKSHelper:: generateKeyPair	Never output	The storage should be under user's home directory. Normally it is ~/.trc/trc.cfg. On Unix-like platforms, the user decides where he likes to store them.	No	Use the CryptoAPI key pair to sign and verify a challenge.
Windows RSA Key	CryptoAPI RSA Key	Input: Key size. Generation: WindowsKSHelper:: generateKeyPair	Never output	Windows Key Store.	No	Use the CryptoAPI key pair to sign and verify a challenge.
Symmetric Key	Triple-DES Key	Input: Seed. Generation: Then use the FIPS 186-2 RNG with the seed to generate a number of random bytes as the return key.	Never output	Volatile memory only	Zeroized when no longer used	Used as the inbound and outbound traffic encryption and decryption key.
FIPS 186-2 RNG Seed	160 bits	Internally generated	Never output	Volatile memory only	Zeroized when no longer used	Produce RNG seed

2.3.6 Self-Tests

The TriCipher Common Crypto Core Module performs the following self-tests at power-up:

- Software integrity check
- Known Answer Tests (KATs)
  - Triple-DES KAT
  - RSA pairwise consistency check
  - SHA-1 KAT
  - SHA-1 HMAC KAT
  - FIPS 186-2 RNG KAT

The CCCM performs the following conditional self-tests:

- Continuous Random Number Generator (RNG) test
- RSA pairwise consistency check

When the self tests fail, an exception will be thrown on the failure. The user is then alerted that the self tests failed, the module will not load, and the module will enter an error state. When in the error state, execution of the CCCM is halted, which inhibits the output of data from the module.

2.3.7 Design Assurance

Tricipher uses a Concurrent Versions System (CVS) server, which is based on version 1.0. It is running on RedHat Linux OS on a Dell 2400 server.

Additionally, Microsoft Visual SourceSafe (VSS) version 6.0 is used to provide configuration management for the CCCM's FIPS documentation. This software provides access control, versioning, and logging.

2.3.8 Mitigation of Other Attacks

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

3 Secure Operation

The TriCipher Common Crypto Core Module meets Level 1 requirements for FIPS 140-2. The sections below describe how to place and keep the module in FIPS-approved mode of operation.

3.1 Initial Setup

The Initial setup consists of loading the .dll file on the Windows XP platform or the .so file on the Linux system.

3.2 Crypto Officer Guidance

The Crypto Officer is responsible for installing/uninstalling, configuring, and managing the module. Before installing the module, the Crypto Officer should make sure that the operating system is in single-user mode.

3.2.1 Operating System Setup

The Crypto Officer must maintain control of the installation media.

FIPS 140-2 mandates that a cryptographic module be limited to a single user at a time. Before the module can be installed, the Crypto Officer must have a GPC running Windows XP configured for single-user mode or Linux configured for single-user mode.

To ensure that Linux is running in single-user mode, the Crypto Officer must delete or disable all accounts except for the root account. Additionally, to ensure only one user can be logged in at a time, the root account must be configured to only allow console access logins and all remote server services must be disabled (e.g., telnet or rlogin server daemon).

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.

More information can be found at http://csrc.nist.gov/cryptval/140-1/CMVPFAQ.pdf.

To configure Windows XP for single-user mode, the Crypto Officer must ensure that all remote guest accounts are disabled. This will ensure that only one human operator can log into the OS at a time.

Once the operating system has been properly configured, the Crypto Officer (system "root" account) can be used for installing/uninstalling software and administrating the module.

3.2.2 Initialization

The software module will be provided to the users by TriCipher. The module is installed during installation of the host application. The installation procedure is described in the product's manual.

Before installing the module, the operator needs to ensure that the system runs one of the specified operating systems. The module must be installed, configured, and started before operators may utilize its features.

3.2.3 Management

The Crypto Officer does not perform any management on the PC or the module after installation and configuration.

3.2.4 Zeroization

The Symmetric key and temporary keys are deleted after their use is over. Only the Unix RSA key and the Windows RSA key are stored. The Unix RSA key is stored under user's home directory and protected by the host OS. The Windows RSA key is stored in the Windows Key Store and protected by the host OS. All other keys and key components are generated on the fly and wiped out after use.

3.3 User Guidance

The User accesses the module's cryptographic functionalities. Although the User does not have any ability to modify the configuration of the module, they should check that the host application is enabled and providing cryptographic protection.

4 Acronyms

Table 7 – Acronyms

Acronym	Definition
API	Application Programming Interface
CCCM	Common Crypto Core Module
CCL	Common Core Library
CMVP	Cryptographic Module Validation Program
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSE	Communications Security Establishment
CSP	Critical Security Parameter
CVS	Concurrent Versions System
DES	Data Encryption Standard
DLL	Dynamically Linked Library
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FCC	Federal Communication Commission
FIPS	Federal Information Processing Standard
GPC	General Purpose Computer
HMAC	(Keyed-) Hash MAC
JDS	Java Desktop System
KAT	Known Answer Test
MAC	Message Authentication Code
MD5	Message Digest 5
NIST	National Institute of Standards and Technology
NVLAP	National Voluntary Laboratory Accreditation Program
OS	Operating System
PC	Personal Computer
PKCS	Public Key Cryptography Standards
RAM	Random Access Memory
RNG	Random Number Generator
ROM	Read Only Memory
RSA	Rivest Shamir and Adleman
SHA	Secure Hash Algorithm
SSL	Secure Socket Layer
VSS	Visual SourceSafe