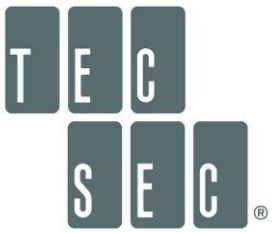


CKM® Cryptographic Module Security Policy

(May 31, 2006)

Abstract:

This document specifies the security policy for the CKM Cryptographic Module (`CKMCRYPTO_FIPS.DLL`) as described in FIPS PUB 140-2.

The information contained in this document represents the current view of TecSec, Inc. on the issues discussed as of the date of publication. Because TecSec must respond to changing market conditions, it should not be interpreted to be a commitment on the part of TecSec, and TecSec cannot guarantee the accuracy of any information presented after the date of publication.

This document is for informational purposes only. TECSEC MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the User.

This document is non-proprietary and allows for the unlimited copying of the document in its entirety.

TecSec may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from TecSec, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The example companies, organizations, products, people and events depicted herein are fictitious. No association with any real company, organization, product, person or event is intended or should be inferred.

TecSec, Constructive Key Management, CKM, CKM Enabled, the CKM Lock & Card Logo, the TecSec logo, Secrypt, XMLsecure and CKMsecure are trademarks or registered trademarks of TecSec, Incorporated. All other names are trademarks of their respective owners.
Microsoft Windows, the Windows logo, Windows NT, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

This product is protected by one or more of the following U.S. patents, as well as pending U.S. patent applications and foreign patents: 5,369,707; 5,369,702; 5,680,452; 5,717,755; 5,898,781; 5,375,169; 5,787,173; 5,440,290; 5,410,599; 5,432,851; 6,075,865; 6,229,445; 6,266,417; 6,490,680; 6,542,608; 6,549,623; 6,606,386; 6,608,901; 6,684,330; 6,694,433; 6,754,820; 6,845,453; 6,885,747.

© 2005 TecSec® Incorporated. All rights reserved.

Table of Contents

1 Introduction

TecSec® Incorporated's Constructive Key Management® (CKM®) Cryptographic Module (`CKMCRYPTO_FIPS.DLL`) (Software version 2.0.0.11) is a FIPS 140-2 Level 1 compliant, general purpose, software based cryptographic module running upon the Microsoft® Windows® Operating System. It is built as a Dynamic Link Library (DLL) and encapsulates several different cryptographic algorithms in an easy-to-use cryptographic component.

1.1 Documents

The *CKM® Cryptographic Module Security Policy* is a component of the submission to CMVP for the validation of the CKM Cryptographic Module.

2 Module Definition

The following sections define in greater detail the boundaries of the Module.

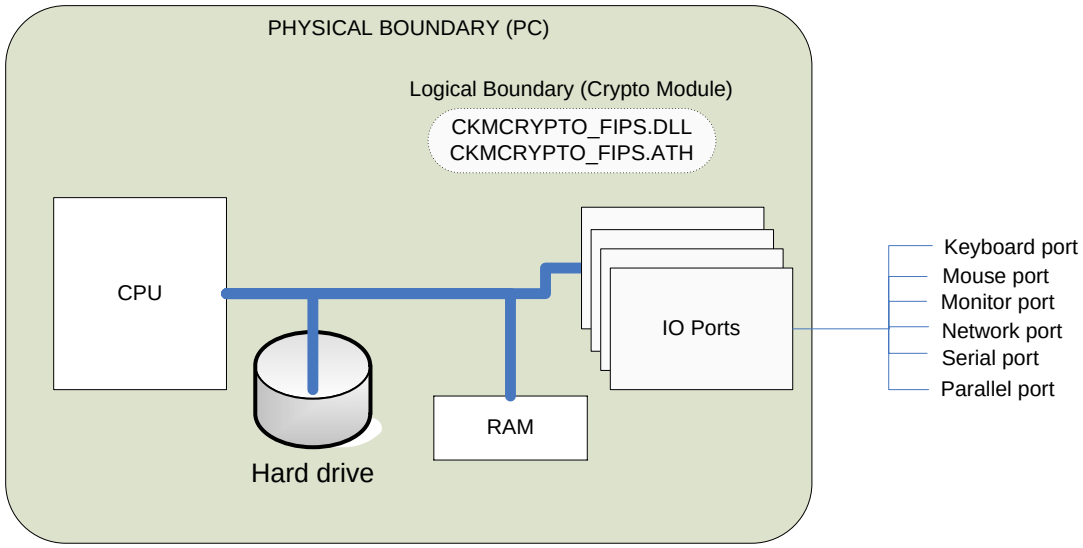
2.1 Hardware Platform

For the purposes of the FIPS 140-2 evaluation, the CKMCRYPTO_FIPS dll was installed and tested on a Dell Dimension 4100 personal computer system with the following components:

- An Intel Pentium III 933 MHz processor
- Dell 4100 Motherboard
- 200 watt Power Supply
- Metal lined hard plastic case
- 512 MB of RAM
- 1 serial port and 1 parallel port
- 1 keyboard, 1 mouse, and 1 monitor port
- 1 network port
- 80 GB hard drive
- Microsoft Windows 2000 operating system (operating in single operator mode) or Microsoft Windows XP operating system (operating in single operator mode).

2.2 Module Diagram

The following diagram shows the logical interconnections between the different hardware components of the Module, as well as the software components.



2.2.1 Module Ports and Interfaces

CKMCRYPTO_FIPS is a software component and as such has two distinct sets of interfaces. The logical interfaces are defined as the C++ Application Programming Interfaces (APIs)

exported by the CKMCRYPTO_FIPS component. The physical interfaces are the standard I/O ports on the personal computer. These include the following:

- 1 Serial port
- 1 Parallel port
- 1 Keyboard port
- 1 Mouse port
- 1 Monitor port
- 1 Network port

The external devices that connect to these ports are outside of the physical boundary and therefore are not part of the CKMCRYPTO_FIPS validation.

The following table will help in identifying the interfaces, their relationships and the physical ports.

Interface	**Logical Interface**	**Physical port**
Data Input	Input parameters to the API functions	Standard PC ports
Data Output	Output parameters to the API functions	Standard PC ports
Control Input	API function calls	CLI and Standard PC Ports
Status Output	Error code return values from the API functions	CLI and PC monitor
Power	Not Applicable	120V 60 Hz AC supplied to the PC

The design of the Module forces the input and output data to be separate and distinct as it flows through the software portion of the Module. These data flows are logical data paths. These logical data paths are primarily used for output data exiting from the Module during functions like key generation, encryption, zeroization of cryptographic keys and other critical security parameters, etc. The hardware portion of the Module can have either plaintext or ciphertext data flowing through the paths at any given time.

The *CKMCryptoAccess.chm* document (API documentation for CKMCRYPTO_FIPS) contains information that identifies all input and output parameters. These input and output parameters are the logical data paths to the software component of the Module. The documentation (CKMCryptoAccess.chm) also identifies all the objects and functions that behave as the Module's control input and status output. The input and output parameters identify (either through the parameter name or in the parameter description) whether the data for that parameter is plaintext or ciphertext.

2.3 Software Environment

The software environments in which the Module was tested are the Microsoft Windows 2000 operating system and the Window XP operating system. This is a commercial off the shelf (COTS) platform. The Module can also execute without modification on the following operating systems:

- Windows 2003 Server (32 bit)(all variants)
- Windows 2000 Server (32 bit)(all variants)

2.4 Software element definition

The software component of the Module consists of the following elements:

- CKMCRYPTO_FIPS.DLL (the software element that contains the cryptographic and security functionality)
- CKMCRYPTO_FIPS.ATH (the TecSec produced signature of the CKMCRYPTO_FIPS.DLL that is used during Power-On self-tests to help ensure that the Module has not been modified).

3 Approved mode of Operation

The Module is in the FIPS 140-2 approved mode of operation whenever the Approved algorithms are used.

4 Roles, Services and Authentication

The Security Policy is used to define the security rules and operations by which the Module operates. The following sections define these rules, and operations in the context of Roles, Services and Authentication in more detail.

4.1 Roles

CKMCRYPTO_FIPS supports the following FIPS 140-2 roles:

Role	**Description**
User	The User is any entity that can access services provided by the Module This includes, but is not limited to, applications running on the Windows 2000 operating system. The User role is implicitly selected when a process calls any API function in CKMCRYPTO_FIPS. The User implicitly has full Read Write and Execute (RWE) access to all functionality of the Module.
Crypto Officer	The Crypto Officer is any entity that can install the Module onto the personal computer, configure the operating system, or access services provided by CKMCRYPTO_FIPS. The Crypto Officer is also defined as a User and therefore may access all of the services available to the User role. The Crypto Officer is implicitly selected when installing CKMCRYPTO_FIPS or configuring the operating system.

Note: The Module does not have maintenance mode and therefore does not support the Maintenance Role.

4.2 Services

This section provides information about the services available within the Module. To see the detailed interface descriptions for these services, look at the respective objects and functions in the *CKMCryptoAccess.chm* document.

The following table defines the FIPS cryptographic functions that are provided by the Module.

Service type	Algorithm	Standard	Implementation interface
Symmetric Ciphers (in CBC and ECB modes)	AES	FIPS 197	ICKMSymmetricAlgorithm
	Triple-DES (2 key)	FIPS 46-3	
	Triple-DES (3 key)	FIPS 46-3	
Digital Signatures	RSA Signatures	PKCS1 v1.5	ICKMRSAAlgorithm
	DSA Signatures	FIPS 186-2	ICKMDSAAlgorithm
Key Generation	DSA Key Generation	FIPS 186-2	ICKMDSAAlgorithm
	DSA Parameter gen and verify	FIPS 186-2	ICKMDSAAlgorithm
Message Digest	SHA-1	FIPS 180-2 With change notice	ICKMHashAlgorithm
	SHA224		
	SHA256		
	SHA384		
	SHA512		
Message Authentication	HMAC-SHA1	FIPS 198	ICKMHashAlgorithm
	HMAC-SHA224		
	HMAC-SHA256		
	HMAC-SHA384		
	HMAC-SHA512		
Random Number Generator	NIST-Recommended Random Number Generator based on ANSI X9.31 Appendix A.2.4 using the 3-Key Triple-DES and AES algorithms	ANSI X9.31 using AES 128	ICKMMiscCrypto or ICKMMiscPRNG
Key Establishment	RSA Key Transport	PKCS1 v1.5	ICKMRSAAlgorithm
Other Functions	Self-test	N/A	ICKMSelfTest

The following table describes the non-FIPS approved algorithms provided in the Module.

Service type	Algorithm	Standard	Implementation interface
Symmetric Ciphers (in CBC and ECB modes)	DES	FIPS 46-3	ICKMSymmetricAlgorithm
	Twofish	----	
	Blowfish	----	
Stream ciphers	P-Squared	Proprietary	ICKMSymmetricAlgorithm
Key Generation	RSA Keys	Using the FIPS 140-2 approved PRNG	ICKMRSAAlgorithm
	DH Keys	FIPS 186-2	ICKMDHAlgorithm
	DH Parameters	FIPS 186-2	ICKMDHAlgorithm
	AES, DES, Triple-DES, Twofish, Blowfish, P-Squared	Using approved PRNG	ICKMMiscCrypto or ICKMSymmetricAlgorithm

Message Digest	MD5	RFC 1321	ICKMHashAlgorithm
Message Authentication	HMAC-MD5	FIPS 198	ICKMHashAlgorithm
Key Establishment	DH Key Agreement	----	ICKMDHAlgorithm
	CKM Key Construction	ANSI X9.69 ANSI X9.73 ANSI X9.96	ICKMCombiner

The following table identifies the Critical Security Parameters (CSPs) and types of available access for the supported services.

Service	Cryptographic Keys and CSPs	Type(s) of Access (RWE)	Role
NIST-Recommended Random Number Generator based on ANSI X9.31 Appendix A.2.4 using the 3-Key Triple-DES and AES algorithms[1]	Seed value, Seed key, random number	W – optionally by User for half of the seed value. The remainder of the seed value and the seed key are internally generated and not accessible to the User. R – random number	User
AES, Triple-DES Encryption and Decryption	Secret Key	RW	User
RSA, DSA Signing	Private Key	RW	User
RSA, DSA Verification	Public Key	RW	User
RSA Key Transport, DH Key Agreement	Public and Private keys	RW	User
SHA(1, 224, 256, 384, 512) Hashing	Hash	RW	User
HMAC (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512)	Secret Key and Hash	RW	User
Self-tests	None	E	User
DSA Key Generation	Public and Private keys	R	User
DSA Parameter Generation	Domain Parameters	R	User
DSA Parameter Verification	Domain Parameters, Seed, Count	W	User

4.3 Authentication

Within the constraints of FIPS 140-2 Level 1, the Module does not directly implement User authentication. It depends on the operating system for operator authentication.

[1] This is the title of the document published on the NIST site. The Module utilizes the AES 128 variant as described within the document.

5 Finite State Model

The *CKMCryptoAccess.chm* document includes the Finite State Model. This document is delivered as part of our CKM Cryptographic SDK and is also included in the distribution to the Cryptographic Modules Testing (CMT) lab.

6 Physical Security

The Module was tested while executing on a standard Intel-compatible personal computer. This platform comprises a multi-chip standalone module that includes standard, production grade components and an enclosure of production grade strength, meeting all FIPS 140-2 level 1 physical security requirements.

7 Operational Environment

7.1 Operating system requirements

The Module is relying on the operating system to protect the memory and code of the Module as it is operating. The Microsoft Windows 2000 and Windows XP operating systems place each process into its own virtual memory space and restrict access from other processes so that no process can directly modify another process.

A process that uses the Module will make a request to the operating system to load the Module into the process's virtual memory and map the exported functionality of the Module so that the process can access the security functions within the Module.

The Module is completely independent. It does not communicate with other processes. This helps to ensure the protection of private and secret keys within the Module.

The Module is restricted to a Single Operator Mode of Operation as specified in the FIPS 140-2 requirements. The operating system is responsible for performing all multi-tasking operations in such a manner that other processes cannot intervene when the Module is active at a particular instance in time.

7.2 Module Integrity

The integrity of the Module is ensured through the use of self-tests that are run automatically by the Module before any service is made available to the User. These self-tests are also available to the User so that they can further ensure that the Module is performing properly. The self-tests are designed so that any cryptographic state that existed within the algorithm object is either not used, or is cleared. This is done so that there can be no compromise of cryptographic information during a self-test.

During the creation of the Module, the Module is signed using the Microsoft Authenticode and a Verisign code signing certificate. This step is performed so that the Crypto Officer can perform a manual check using functionality provided by the Operating System to ensure that the Module is not modified. This process is further extended so that the Module can perform its own check of its integrity. This second stage uses a TecSec generated RSA key pair (different from the Verisign key pair. The public key is stored within the Module and therefore within the

Authenticode signature) and is used to verify a second signature of the Module (including the Authenticode signature). This signature is stored in the CKMCRYPTO_FIPS.ATH file. During the Module load procedure (each time it is loaded into the process's virtual memory) the Module reads the CKMCRYPTO_FIPS.ATH file and uses the RSA Public key (stored in the Module) to verify the signature of the Module. If this verification process fails for any reason, the Module flags a FIPS self-test error and will not perform any cryptographic services for the User. Access to the matching private key (used to create the signature) is restricted to only authorized TecSec personnel.

Any failure of the signature checks or of any self-tests or of any of the continuous tests will place the Module into the FIPS error state which will disable all cryptographic operations until the Module is returned to the Power-Off state.

The use of these mechanisms, in combination with the individual algorithm self-tests, help to ensure that the Module integrity is unmodified and performing as validated.

7.3 CSP Protection Mechanisms

The CSPs provided by the Module are internally protected through the following means (beyond the Module level protections described above).

- All private/secret keys are stored in RAM. The keys are not persisted by the Module.
- When the user signals that they are finished using the CSP, the key(s) used by the CSP are destroyed as follows:
    1) The key value is overwritten with zeros.
    2) The memory that was holding the key is deallocated.

7.4 Other Assumptions

Proper FIPS configuration and usage of the Module requires following instructions in the *Crypto Officer Guide.doc* document. In addition, the rules described in Section 4 of this document must also be followed.

8 Cryptographic Key Management

Any keys generated by the Module are generated using the module's random number generator (RNG). The Module does not store the keys in any form of persistent storage. The keys are only held in memory for as long as the Module requires them, and are then zeroized by overwriting the entire key buffer with zeros. All key buffers are also zeroized upon initialization so that any residual key fragments due in part to Power-Off followed by Power-On cannot compromise key material.

8.1 Key Generation

All keys generated by the Module are generated using the module's FIPS approved RNG. The following standards and/or processes are used in the key generation.

- DSA keys and parameter sets are generated according to FIPS 186-2.

- RSA keys are generated using the Approved RNG to generate the private prime numbers p and q. The following process is used to generate RSA prime values:
    1) Compute a random number p to be half of the bit size of the desired key
    2) Ensure it is reasonably prime using the following:
        a. Divide the number by the small primes from 2 to 17863. If no remainder go to step 1
        b. Apply the Fermat method. If it fails go to step 1
        c. Apply the Miller-Rabin method using 50 iterations. If it files go to step 1
    3) Perform the same steps as above to generate the q value.
    4) Multiply p and q to make sure that the result is the desired number of bits. If not go to step 3 and recompute q.
- All symmetric and Message Authentication keys are generated using the Approved RNG to generate a byte string of the required length for the algorithm. Triple-DES keys (which contain a parity bit in each byte) do not have the parity bit computed. The parity bit is random. The Triple-DES algorithm implemented in the Module does not require, nor does it use, the parity bits.

In all of these operations the RNG is used in the approved manner including the continuous RNG test as well as checks to make sure that the seed and key are not the same values.

The Module uses a system whereby only half of the seed value can be known to the User. The other half of the seed and the RNG key value are determined by the collection of entropy within the Module. The Module uses the AES 128 algorithm for the G function that equates to the strength of a 3072 bit Diffie-Hellman key, or a 3072 bit RSA key (according to NIST Special Publication 800-57 Part 1 August, 2005). The Module only generates asymmetric keys up to 1024 bits in length (due to US export approval limitations). Also the RNG will periodically re-key itself by gathering new entropy (gathered from numerous Microsoft Windows internal data structures that contain dynamic data, as well as microsecond timers and other similar sources) and using this as new PRNG seed and key values.

The raw data collected in the entropy process is run through a combination of SHA-1 and SHA-256 hash algorithms in a complex relationship so that the entropy is distributed throughout both resultant values. An 8 byte portion of the SHA-1 result value is returned to the user when the user computes entropy. This value is typically used as the first half of the seed value for the PRNG. The second half of the seed value and the PRNG key is extracted from the SHA-256 value that is stored internally to the module and is not available to the user. The estimated entropy is 200 bits per entropy computation, with the system performing at least two entropy computations before the first random value is computed.

All of these measures together provide assurance that an attacker would have to perform as many or more operations to try to recreate the key than it would take to perform a brute force algorithm attack (guessing the key).

No intermediate key values, beyond those required by the key generation standards, are exportable from the Module.

8.2 Key Establishment

The Module provides the following non-FIPS approved key establishment methods.

Method	Description
DH Key Agreement	Diffie-Hellman keys of 512 and 1024 bits are supported. (provides either 56 or 80 bits of encryption strength)
RSA Key Transport	RSA (for Key Transport) is supported for RSA keys of 768 and 1024 bits in length (provides either 69 or 80 bits of encryption strength).
CKM Key Construction (non-compliant)	The CKM Key Construction allows for the construction/reconstruction of 512 bit symmetric keys while only transferring benign information between the participants.

The Module does not perform key establishment by itself. It relies on the User to choose the appropriate technique and key sizes to ensure that the wrapped key is properly protected.

8.3 Key Entry and Output

Keys are not input or output from the module; key establishment is used.

8.4 Key Storage

The Module does not store keys in any persistent manner on any persistent media.

8.5 Key Destruction

The Module stores keys in memory only as long as they are required. As soon as the key is no longer needed, it is zeroized by writing zeros over the entire key buffer before the memory is returned to the process/operating system. The user tells the Module that the key is no longer needed whenever any of the following events occurs.

1) The user calls the “Final” routine for the algorithm in question. The detailed explanation and identification of the “Final” routine is contained within the CKMCryptoAccess.chm document.
2) The algorithm is released from use. This happens when the user calls the Release function for all pointers to the algorithm object. The detailed explanation of this is included in the CKMCryptoAccess.chm document.

9 Bypass Mode

The Module does not have a bypass mode.

10 Self-tests

The following self-tests are performed within the Module. In addition, the module performs an internal integrity test as described in section 7.2.

10.1 Power-On

The following FIPS-approved algorithm tests are initiated upon Power-On.

- Internal integrity test as described in section 7.2
- Triple-DES TwoKey ECB KAT for encrypt and decrypt
- Triple-DES TwoKey CBC KAT for encrypt and decrypt
- Triple-DES ThreeKey ECB KAT for encrypt and decrypt
- Triple-DES ThreeKey CBC KAT for encrypt and decrypt
- AES 128-bit key ECB KAT for encrypt and decrypt
- AES 128-bit key CBC KAT for encrypt and decrypt
- AES 192-bit key ECB KAT for encrypt and decrypt
- AES 192-bit key CBC KAT for encrypt and decrypt
- AES 256-bit key ECB KAT for encrypt and decrypt
- AES 256-bit key CBC KAT for encrypt and decrypt
- SHA-1 and HMAC-SHA-1 KAT
- SHA-224 and HMAC-SHA-224 KAT
- SHA-256 and HMAC-SHA-256 KAT
- SHA-384 and HMAC-SHA-384 KAT
- SHA-512 and HMAC-SHA-512 KAT
- DSA Signature Verify KAT (1024-bit)
- DSA Signature Creation and Verify (1024-bit)
- RSA Signature Verify (1024-bit)
- RSA Key Wrap (1024-bit)
- RSA Key Unwrap (1024-bit)
- RSA Signature Creation and Verification (1024-bit)
- RNG Mono bit test (20000 bits)
- RNG Poker test (20000 bits)
- RNG runs test (20000 bits)
- RNG long runs test (20000 bits)
- RNG KAT test

The RSA Key Wrap and RSA Key Unwrap tests are performed together. A test value is encoded using PKCS #1 v1.5 padding using the block type 2 padding. This value is then encrypted using a test RSA key (1024 bit). This value is then decrypted with the corresponding RSA private key in standard format and decoded using PKCS #1 v1.5. The result is then compared with the original test value. Then the encrypted value is also decrypted with the matching RSA Private key in CRT mode, and decoded using PKCS #1 v1.5. This result is also compared with the original test value. If any step in the process fails, then the test fails and the module is placed into the FIPS-Error state.

10.2 Conditional

The following FIPS-approved algorithm tests are run conditionally.

- Continuous random number generator test

- DSA Key Gen (1024-bit)
- RSA Key Gen (1024-bit crt & std)

11 Design Assurance

The Module is designed, developed, and deployed in a manner that protects its integrity throughout the process. Guidance is provided to Crypto Officers and Users of the Module.

11.1 Configuration Management

The Module source code is stored in a Concurrent Versions System (CVS) source code repository to ensure the integrity of the Module throughout its development. The system stores distinct versions of the Module's source code. While in storage, files are protected against unauthorized modification through the need for username and password authentication and through the fact that the server is only accessible from the internal network.

11.1.1 Versioning

Every time a file is checked in with a change, a new version is assigned to that file by CVS. This allows the file to be retrieved for any version at any time it is needed. Also, whenever binary objects are built, the version number of the binary object is updated within the internal versioninfo structure (which allows Users and Crypto Officers to view the current version number of the binary through functionality in Windows) and a symbolic label is placed on the entire set of code that was used to build the binary. This label is called a Tag.

11.1.2 FIPS Testing

All of the documents described in section 1.1 above will be included to support the FIPS testing effort. This information will be placed on a CD that will be clearly labeled as to contents and version. As some of the materials on this CD are proprietary and consist of protected Intellectual Property (IP), the CD will be marked as proprietary. The following table will describe the different elements on the CD and their proprietary nature.

Document, file, or package	**Proprietary status**
Security Policy	Non-proprietary
CKMCryptoAccess.chm	Proprietary
CKM Cryptographic Module v2 Source Code Description.doc	Proprietary
CKMCryptoDesign.chm	Proprietary
Source Code	Proprietary
Crypto Officer Guidance	Non-proprietary
CKM Desktop version 2.3	Commercially available CKM product

11.2 Delivery and Operation

The Module is delivered to the User as part of the CKM Desktop version 2.3 installation package. The optional CKMCrypto API (includes sample applications, the *CKMCryptoAccess.chm* user documentation, libraries, headers, and customer support) is available through TecSec's Customer Support and Services department.

For the purposes of the FIPS testing, the *CKM Cryptographic Module v2 Source Code Description.doc* document contains the annotated list of source code for the Module, the version of each file, and its relative location within the source code archive. The source code is packaged into a Zip file that is included on the CD (see 11.1.2 above). All of the documents described in section 1.1 above will be included in this distribution. This information will be placed on a CD that will be clearly labeled as to contents and version. As some of the materials on this CD are proprietary and protected IP, the CD will be marked as proprietary.

11.3 Guidance Documents

The *Crypto Officer Guide.doc* specifies the requirements and procedures that the Crypto Officers need to follow to properly install, configure and maintain the Module.

The *CKMCryptoAccess.chm* document (which includes User Guidance) explains the proper use of the Module and its cryptographic services and functions.

12 Mitigation of Other Attacks

This Module is not designed to specifically mitigate any attacks beyond those already specified in this Security Policy (as required by FIPS 140-2 Level 1).

13 For More Information

For the latest information on TecSec products and services, please visit http://www.tecsec.com

For the latest information on the Microsoft operating systems, please visit http://www.microsoft.com/windows

Appendix A: Acronyms

The following acronyms are used throughout this document:

AES	Advanced Encryption Standard
ANSI	American National Standards Institute
ATH	Authorization file that contains the TecSec signature of the Module
CBC	Cipher Block Chaining - An algorithm mode of operation.
CHM	Compiled HTML Help file
CKM®	Constructive Key Management
CRT	Chinese Remainder Theorem
DES	Data Encryption Standard
DH	Diffie-Hellman
DLL	Dynamic Link Library
DSA	Digital Signature Algorithm
ECB	Electronic Code Book - An algorithm mode of operation.
ES	Encryption Scheme (See PKCS#1)
FIPS	Federal Information Processing Standards
HMAC	Hashed Message Authentication Code
HTTP	Hyper Text Transfer Protocol
IV or IVEC	Initialization Vector
KAT	Known Answer Test
MAC	Message Authentication Code
MD5	Message Digest 5
NIST	National Institute of Standards and Technology
PIN	Personal Identification Number
PKCS	Public Key Cryptography Standards
PKCS#1	The RSA Cryptography Standard
PRNG	Pseudo Random Number Generator
RNG	Random Number Generator
RSA	Rivest, Shamir, and Adleman
SHA	Secure Hash Algorithm
STD	Standard

Appendix B: Master Components List

The CKMCRYPTO_FIPS.dll Module is a software component that is intended to operate as part of an application on a personal computer that is running the Microsoft® Windows® 2000 or Microsoft® Windows® XP operating system. The cryptographic Module includes the validated CKMCRYPTO_FIPS.dll and the hardware elements of the personal computer. Neither the operating system nor the calling application is a component of the Module.

Hardware Components

The following listed hardware elements are components of the FIPS 140-2 Module submission, as FIPS 140-2 requires a specification of the hardware upon which the software is run. The components are commercial off the shelf components and integrated circuits designed to meet standard commercial grade usage and needs. This includes parameters such as vibration, shock, reliability, temperature and power. The following items are included:

- The PC case (Metal or hard plastic)
- The main processor (CPU)
- The hard disk
- Memory (RAM)
- CD-ROM drive
- Floppy disk drive

Software Components

The following software elements are components of the Module.

- CKMCRYPTO_FIPS.DLL (The executable binary code)
- CKMCRYPTO_FIPS.ATH (The internal signature for the executable binary code)

The *CKM Cryptographic Module v2 Source Code Description.doc* document lists all of the source files that are used to create the CKMCRYPTO_FIPS.DLL.