Trust Digital Crypto Library Cryptographic Module (A library for Windows 32, Windows CE, MS SmartPhone, Windows Mobile 5 and Smartphone 5, Palm OS, and Symbian)

Security Policy

Author: Mike Shahbazi
Updates: Norm Laudermilch
Module Software version: 3.0 and 3.0.0.1

Date: December 2004
Update: October 2006

Abstract: This document describes the Trust Digital Cryptographic Module Security Policy submitted for validation, in accordance with the FIPS publication 140-2, level 1.

Contents

INTRODUCTION

The Trust Digital Crypto Library Cryptographic Module (hereafter referred to as CM) is a software library implementing a number of cryptographic functions. At the code level, the module is the same on all platforms. The module can be compiled to run on a number of different systems. This module provides cryptographic services accessible from software programs written in C/C++ through Application Program Interfaces (APIs). The DLL (dynamically linked library) format of this module allows it to be embedded in existing applications targeted for Palm, Pocket PC, and Symbian operating systems. Though not tested, the RIM Operating System (OS) is supported. The table below identifies each tested operating system with its corresponding CM binary format.

OPERATING SYSTEM	MODULES
Windows 32-bit based systems	CryptoLib.dll
Windows SmartPhone	CryptoLib.dll
Windows Pocket PC	CryptoLib.dll
Windows Mobile 5	CryptoLib.dll
Windows Mobile 5 Smartphone	CryptoLib.dll
Palm OS	CryptoLib.prc CryptoLib DB.pdb
Symbian	CryptoLib.dll

The CM has been designed and implemented to meet the Level 1 requirements of FIPS Publication 140-2 (FIPS 140-2). The CM supports the following FIPS Approved algorithms:

- AES (Certs. #69 and #456)
- TDES (Certs. #177 and #475)
- SHA-1 (Certs. #164 and #520)
- HMAC-SHA-1 (Certs. #164 and #520, Vendor Affirmed),

as well as FIPS-Approved Random Number Generation (RNG), ANSI X9.31 (A.2.4).

The CM also supports the following non-FIPS Approved algorithms:

- Blowfish
- TwoFish
- RC4
- TEA
- Fast XOR
- MD5

The CM is designed and written in the C programming language. Additionally, it has a C++ interface to ensure a good object-oriented architecture for Trust Digital products and third-party module developers. The module is used in several Trust Digital applications. The current operational environment of the CM is depicted in Figure 1.

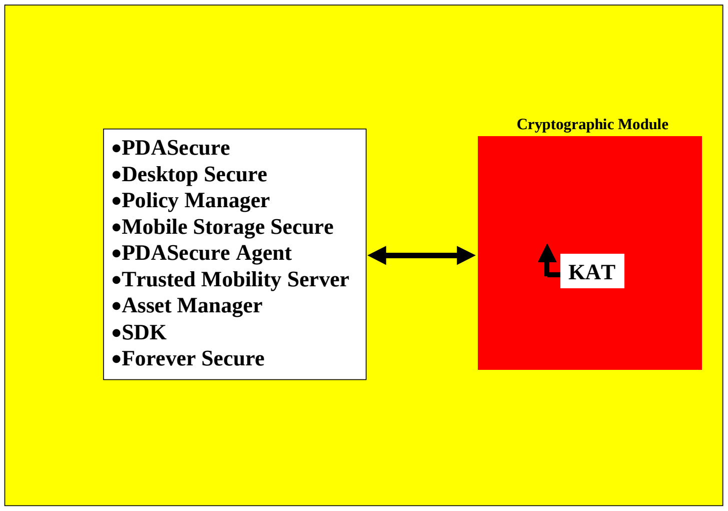


Figure 1: Cryptographic Module (CM) Operating Environment

Most Trust Digital products, as well as Third Party applications (via the SDK), use the CM for authentication and encryption purposes. In Figure 1, the arrows indicate calls made by processes to the CM. Internal to the CM, the Known Answer Tests (KATs) make calls to the CM in order to validate its correct operation prior to its use by applications. The CM is identical at the source code level for all identified hardware platforms and operating systems.

OPERATIONAL ENVIRONMENT

The Windows system that supports the CM must be configured in single user mode. This is accomplished by having only an administrator account setup on the Windows machine and the disabling of all services that could potentially lead to additional users (e.g. telnet, ftp). In contrast to a desktop PC, a PDA is intended for the personal use of a single user. In order to regard a PDA as a trusted environment for CM execution, the following conditions should be met:

1) A valid security policy should be applied on the device so that any unauthorized access and execution of untrusted software is prevented.
2) Physical port security is applied. Major physical ports on PDAs are serial, infrared and USB ports. Also, most PDAs communicate with a PC through a sync cradle, which supports communication between the devices using a proprietary hardware interface. To ensure the environment is trusted, the PDA device must be physically isolated from the cradle hardware and physical port activity prevented while performing critical sensitive data processing.

The CM was tested on the operating systems identified below. We used no special electronics, devices or power supplying schemes other than the standard parts provided with the PDA and PC devices.

It is a responsibility of the Crypto Officer to configure the operating system to operate securely and prevent remote login. These settings will be different for different operating systems and device types.

The following operational environments were tested in FIPS mode:

- Windows NT 4.0 Workstation Operating System, SP 6 (in single user mode).
- Windows 2000 Workstation (single user mode)
- Windows XP Pro (single user mode)
- Palm® OS 4.1
- Palm® OS 5.2.1
- Palm® OS 5.2.1 H
- Palm® OS 5.4.5
- Microsoft Pocket PC 3.0 (Pocket PC 2002)
- Microsoft Pocket PC 3.0 (Pocket PC Phone Edition 2002)
- Microsoft Pocket PC 4.20 (Windows Mobile 2003)
- Microsoft Smart Phone 2002
- Microsoft Windows Mobile 5
- Microsoft Windows Mobile 5 Smartphone
- Symbian OS 7.0

The CM is also suitable for similar platforms from the same or other manufacturers using comparable processors and equivalent or greater system resources.

CRYPTOGRAPHIC MODULE SPECIFICATION

The cryptographic boundary

The cryptographic boundary of the CM is a logical boundary around the binary code, which comprises the CM. The CM is a software module that performs all cryptographic functions and is defined as a multi-chip stand-alone module, in FIPS 140-2 parlance. The CM runs as a library or as embedded code under different operating systems installed in commercially available PCs, or in a variety of PDA devices. The CM source code is common across all supported platforms. Each supported OS requires a separate build of the CM in a format required by the client application's architecture. This requirement for a separate build does not change the version number of the software, as it is the same code compiled for a different operating system. Applications interface to the CM via the CM API

Self-Test

The CM implements a number of power-up self-tests to validate the proper functioning of the cryptographic functions, as well as a continuous self-test for the RNG. These tests cover all the FIPS-approved functions available from the CM.

The following identifies the order in which the tests are run:

- Self-test:
  - Power-up Self-tests:
    - Software Library Integrity:
      - HMAC of CM executable code
    - Algorithm KATs:
      - HMAC-SHA-1, AES, TDES
    - RNG Integrity
  - Conditional Test:
    - RNG Continuous Test

Known answers for cryptographic algorithms test are stored in a hard-coded database within the CM and memory comparison functions are used to compare the results. If the tests are not passed, the CM will return an error message via a return value from the API, enter the "CM Failure" error state, and halt. Since the CM will be unavailable to perform cryptographic functions, the User application cannot utilize any cryptographic functions until a hard reset of the device is performed and a valid CM library is provided that passes the tests.

The CM code integrity check is performed using HMAC-SHA-1. A keyed-hash of the CM's code and data is calculated and checked for validity.

An integrity test is performed on the RNG. This test generates two random values and compares them to confirm they are different. If the two generated values match, the test fails and the CM enters the error state. This test is also used to initialize the RNG and the first value for the continuous test.

All self-testing procedures are implemented in sub-program calls that do not return control to the device until the tests are completed.

Power-up Self-Testing

When the CM starts loading, power-up self-testing is initiated automatically. It is comprised of the CM code integrity test and KATs of the FIPS-approved cryptographic algorithms, and the RNG integrity test. If any of the tests fail, the CM returns an error message, a return value from the API, enters the "CM Failure" state, and halts. No functions provided by the CM are accessible when in the error state.

Module Software Services

The CM provides a set of cryptographic security services to Users (i.e., Applications). The set of security services includes data confidentiality, data integrity and authentication support. The security services are supported by both FIPS-Approved and by non-FIPS Approved algorithms. The CM provides the following services with non-FIPS Approved algorithms:

- Symmetric encryption: RC4, Blowfish, Twofish, TEA, XOR
- Hash: MD5

FIPS Module Software Services (FIPS Mode)

The CM provides the following services with FIPS-approved algorithms:

- Hash: FIPS 180-2, SHA-1 and FIPS 198, HMAC-SHA-1
- Symmetric Encryption: FIPS 46-3, Triple DES and FIPS 197, AES
- Random Number Generation: ANSI X9.31, Appendix 2.4

To operate in FIPS Mode, the user of the CM shall only uses FIPS-Approved algorithms. All Data Encryption Keys (DEKs) must be generated using the FIPS Approved RNG or entered into the module in encrypted form. DEKs shall only be output from the module in encrypted form using Key Encrypted Keys (KEKs) generated/entered by FIPS approved methods (entered plaintext via API or generated by the RNG). The underlying operating system must be configured for single-user mode, as applicable.

All symmetric encryption algorithms use Electronic Code Book (ECB) mode of encryption, in accordance with FIPS-81.

Module Hardware Services

The CM is a software implementation. Therefore, it is dependent upon the underlying platform CPU hardware to provide processing and memory storage services.

Electromagnetic Interference/Electromagnetic Compatibility

The CM has been tested on devices that meet FCC Class B compliance for Level 3 modules.

Cryptographic Module Ports and Interfaces

As stated above, the CM is a software module that is dependent upon the underlying platform CPU hardware to provide processing and memory storage services. The CM has C and C++ based APIs to provide the means to integrate with an application. The module is implemented as a library of cryptographic primitives.

All parameters passed to the CM routines and C++ object classes are through *Data Input* interfaces. These interfaces accept keying material for CM processing that is stored by the User application in a local database.

All control flags passed to the CM routines and C++ object classes are through *Control Input* interfaces.

All parameters returned by the CM routines and C++ object classes are through *Data Output* interfaces. These interfaces pass ciphertext keying material (which is to be stored by User application, since the CM doesn’t provide key persistence) and data.

All error codes returned by the library routines and C++ object classes are through *Status Output* interfaces.

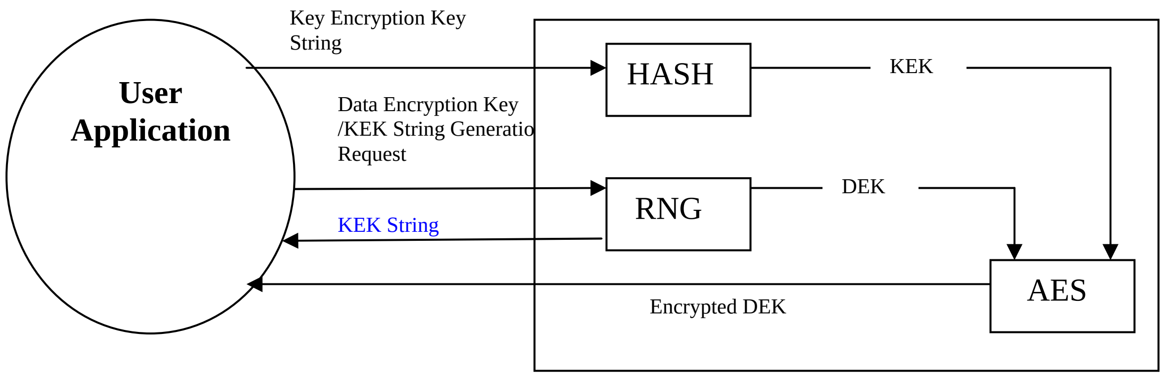
Module Key Management Services

The CM generates keying material and provides it to User applications, but it does not provide key persistence. Therefore, it is the User application’s responsibility to ensure that encrypted working keys are stored and protected. The CM is responsible for wiping its own internal keys without any easily recoverable trace when they no longer needed. The User application must implement persistence of the encrypted keys and authentication data needed to initialize the next security session. The keys are exported in encrypted form from the CM to the User application.

Key Generation

The CM RNG generates keying material for the symmetric ciphers. This is accomplished by the application issuing a request to the CM API to return the desired number of bytes. The RNG employs the FIPS-approved algorithm specified in ANSI X9.31, *Digital Signature Using Reversible Public Key Cryptography*, Appendix 2.4. The figure below depicts the data encryption key generation process. The data encryption key (DEK) is returned to the User application for use while the application is active. A Key Encryption

Key is used for the output of the DEK. The KEK is entered into the module in plaintext via an API parameter, The KEK can be generated using the module's RNG (an API call to the RNG returns a random value that can be used as the key) or generated by the calling application.


User Application
Key Encryption Key String
Data Encryption Key /KEK String Generatio Request
KEK String
HASH
KEK
RNG
DEK
AES
Encrypted DEK

Key Distribution and Storage

In a Windows environment, all keys generated and/or otherwise processed by the CM reside in the same Windows process space. PDA operating systems use a similar architecture.

The User application passes the encrypted cryptographic keys to the CM, as needed, for data encryption/decryption. Keys are decrypted and used in the CM on behalf of, and for immediate use by, the calling User application. Key distribution is performed by the calling application. The module supports the use of KEKs for the distribution of DEKs.

Zeroization of Keys

Due to the nature of the CM, a limited number of critical security parameters exist in the module. Included are any cryptographic keys (DEKs and KEKs) currently residing in the module, and the internal state of the RNG. The CM zeroizes all critical security parameters when those are no longer needed, such as when the User application detaches from the CM or the zeroization command has been issued. Zeroization is performed by setting all the CM memory units, which contain sensitive information, to the value of "zero." The CM retains control of the processor until zeroization is complete.

Protection of Keys

The CM has its own protected memory space in which it runs. When the CM generates or uses the User application's DEK, the DEK resides in the CM's protected memory space. The platform OS memory management mechanism ensures that no process can access the process space of any other process, including its memory.

CRYPTOGRAPHIC MODULE SECURITY POLICY SPECIFICATION

Identification and Authentication Policy

The CM enables user authentication by the calling application, but does not perform authentication itself. Separation of roles is implicit based on API call.

Access Control Policy

The CM supports two roles: Crypto Officer and User, as specified by FIPS 140-2. The User application will adopt either role, as necessary, during execution.

ROLE	*TYPE OF AUTHENTICATION*	*AUTHENTICATION DATA*
Crypto Officer	None	N/A
User	None	N/A

A User application, acting on behalf of a human user, can read and write security-relevant data only through invocation of a service by means of the CM API. Some CM services can be invoked by the Crypto Officer role, while others are invoked by the User role. The table below identifies, which API calls can be made by the Crypto Officer and User roles while in FIPS mode.

API COMMAND	CRYPTO OFFICER	USER
Load Library		
Start Up	X	
RNDInit (automatically performed)	X	
TestHMAC (automatically performed)	X	
TestLib (automatically performed)	X	
Symmetric Key		
EncryptionLibTDesKeyInit	X	
EncryptionLibTDesKeySchedule	X	
EncryptionLibTDesEncDecBlock		X
EncryptionLibTDesKeyDestroy	X	
EncryptionLibrijndaelInitUT	X	
EncryptionLibrijndaelReleaseUT	X	
EncryptionLibrijndaelCipherInit	X	
EncryptionLibrijndaelMakeKey	X	
EncryptionLibrijndaelMakeBinKey	X	

EncryptionLibrijndaelBlockEncryption		X
EncryptionLibrijndaelBlockDencryption		X
EncryptionLibrijndaelKeyDestroy	X	
Random Number Generation		
EncryptionLibRNDCheckIntegrity	X	
EncryptionLibRNDInit	X	
EncryptionLibRND	X	
Hash		
EncryptionLibEncDigestSHA1		X
Wipe (zeroize)		
EncryptionLibDestroyResourceDatabase	X	
EncryptionLibDestroyRecordDatabase	X	

The following table identifies the Access Rights for the services specified above that make use of a key or a CSP.

Service	Key, CSP	Type of Access
Startup None	**Internal RNG State**	**Write**
EncryptionLibTDesKeyInit	**DEK/KEK**	**Write**
EncryptionLibTDesKeySchedule	**DEK**	**Write**
EncryptionLibTDesEncDecBlock	**DEK/KEK**	**Execute**
EncryptionLibTDesKeyDestroy	**DEK/KEK**	**Write**
EncryptionLibrijndaelInitUT	**DEK/KEK**	**Write**
EncryptionLibrijndaelReleaseUT	**DEK**	**Write**
EncryptionLibrijndaelCipherInit	**DEK**	**Write**
EncryptionLibrijndaelMakeKey	**DEK**	**Write**
EncryptionLibrijndaelMakeBinKey	**DEK**	**Write**
EncryptionLibrijndaelBlockEncryption	**DEK/KEK**	**Execute**
EncryptionLibrijndaelBlockDencryption	**DEK/KEK**	**Execute**
EncryptionLibrijndaelKeyDestroy	**DEK/KEK**	**Write**
EncryptionLibRNDCheckIntegrity	**Internal RNG State**	**Write**
EncryptionLibRNDInit	**Internal RNG State**	**Write**
EncryptionLibRND	**Internal RNG State, DEK/KEK**	**Write (all)**
EncryptionLibEncDigestSHA1	**None**	**None**
EncryptionLibDestroyResourceDatabase	**Internal RNG State,**	**Write**

	DEK/KEK	
EncryptionLibDestroyRecordDatabase	**Internal RNG State, DEK/KEK**	**Write**

Physical Security Policy

The CM is implemented completely in software and is a multi-chip standalone embodiment, as defined in FIPS 140-2. Since the CM is totally a software solution, physical security is provided solely by the host platform and, therefore, the CM itself is not subject to the physical security requirements of FIPS 140-2.

Mitigation of Other Attacks Policy

The CM does not mitigate any other specific attacks.

GUIDANCE DOCUMENTS

Organization-specific security policies will be used to augment the security policy of the CM.

ACRONYMS

AES	Advanced Encryption Standard
API	Application Program Interface
CM	Cryptographic Module
DB	Database
DES	Data Encryption Standard
DLL	Dynamically Linked Library
DSS	Digital Signature Standard
FIPS	Federal Information Processing Standard
HMAC	Keyed Hashing for Message Authentication
KAT	Known Answer Test
OS	Operating System
PC	Personal Computer
PDA	Personal Digital Assistant
RNG	Random Number Generator
SDK	Software Development Toolkit
SHA-1	Secure Hash Algorithm 1
TDES	Triple DES
TEA	Tiny Encryption Algorithm
USB	Universal Serial Bus
XOR	Exclusive Or