Credant Cryptographic Kernel, Version 1.3 and 1.4

FIPS 140-2 Non-Proprietary Security Policy, Version 1.5
Level 1 Validation
August 2004

1. Introduction

Companies are increasingly using diverse mobile devices to store critical business information, improve productivity and enhance customer relationships. These mobile devices represent one of the most severe and often overlooked security threats to the enterprise. Frequently left unmanaged and with little to no enforced security, these devices are an open door to corporate applications, networks and databases and represent potentially significant financial, legal and regulatory liabilities. Without sufficient management tools and enforced security policies, companies have no way to prevent mobile security breaches, know if information is misused, or trace the source of mobile security incidents.

Architected to protect the mobile enterprise, Credant Mobile Guardian (CMG) is the first security solution that addresses an organization's mobile security issues with centrally managed policy administration and strong on-device user authentication and policy enforcement. This cost-effective solution enables organizations with a growing mobile population to take full advantage of the benefits of today's mobile workplace and remain confident that business critical information is secure.

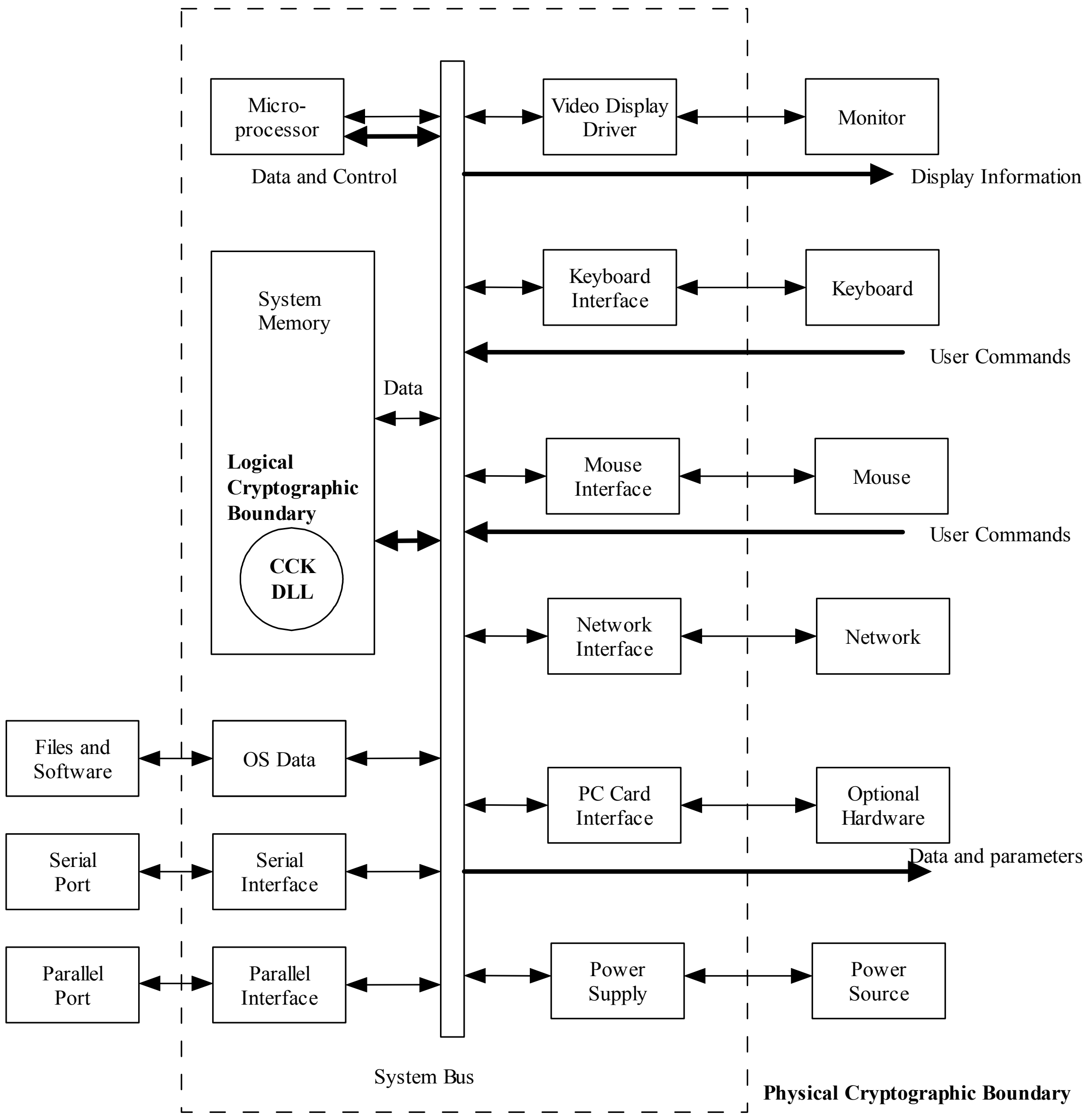
The Credant Cryptographic Kernel (CCK) is the library of cryptographic functions used by the Credant Mobile Guardian (CMG) Suite of mobile security solutions. The CCK takes the form of a (shared or dynamic link) software library, which provides an API to cryptographic functions, including AES, Triple DES, SHA-1, HMAC (SHA-1), and an ANSI X9.31 compliant pseudorandom number generator.

CMG Suite comprises the CMG Server, CMG Gatekeeper, and CMG Shield software products. These three components work together to ensure the security of data on mobile devices. The CMG Shield installs on the mobile device and protects its data from unauthorized access. The CMG Gatekeeper receives policy information from the Server and communicates it to the devices running the Shield. An administrator sets company policies via CMG Server software, and the Server forwards these settings to the instances of the CMG Gatekeeper. When a device synchronizes with its host PC, the Gatekeeper communicates these policies to the device.

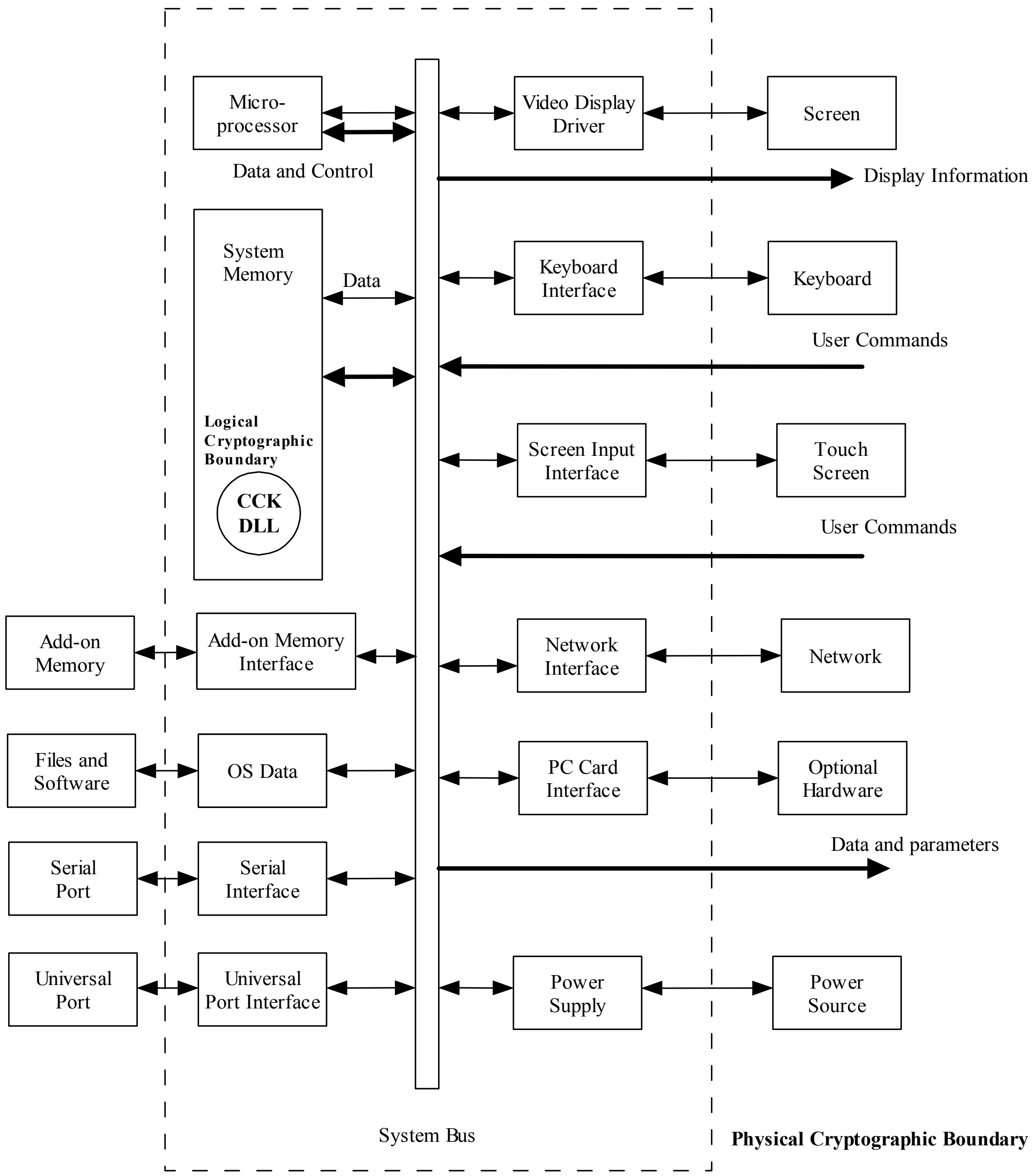
2. Product, Boundary, Module Definition

The CCK library and header file constitute the cryptographic software module for this FIPS 140-2 validation. The logical boundary contains the software modules that comprise the CCK library. The physical boundary for the module is defined as the enclosure of the computer system on which the functions of the library execute.

Windows Physical and Logical Cryptographic Boundaries



PPC and Palm Physical and Logical Cryptographic Boundaries



The module constitutes a multi-chip stand-alone device (listed below), as defined by the FIPS 140-2 standard. The devices on which the CCK runs (as a shared or dynamic link library) include:

- Intel-CPU-based computers running the Windows operating system:
  - Windows 2000, service pack SP1, and
  - Windows XP, service pack SP1
- Personal digital assistants running PalmOS (versions 3.5.1 and greater, but less than 5.0)
- PocketPC 2000, PocketPC 2002 and PocketPC 2003 handheld computers
- Telephony enabled PDAs including:
  - the HandSpring Treo,
  - the Kyocera,
  - the Samsung, and
  - Palm Tungsten W

The CCK was tested as a dynamic link library on a standard, commercially available Intel CPU PC (DELL OptiPlex GX1) running Windows 2000 and on a standard, commercially available X-Scale PPC (Compaq iPaq 3760) running Windows CE 3.0.

3. Roles, Services, Policy

The CCK supports both Crypto Officer (CO) and User roles, where a "user" is the application using CCK services to perform encryption, decryption, hashing, and random number generation. It does not support a maintenance role or a bypass capability. The CCK supports one Approved mode and it supports only FIPS 140-2 approved services (shown in Figure 1).

In the CO role, the CO installs the CCK on a device. It is the CO's responsibility to install the CCK in the Approved mode according to the instructions specified in Section 9 of this Security Policy.

The cryptographic services provided by the software module are shown in the Figure 1. Note that the supported services do not include authentication.

Service Type	**Algorithm**	**FIPS**	**Available in Modes**
Symmetric Cipher	AES	197	Approved
Symmetric Cipher	Triple DES	46-3	Approved
Message Authentication	HMAC (SHA-1)	198	Approved
Message Digest	SHA1	180-1	Approved
Random Number Generation	ANSI X9.31	186-2	Approved

Figure 1. Services offered by the Credant Cryptographic Kernel, applicable algorithm applicable FIPS specification, and availability.

The access granted to security relevant data items of these services for each role is shown in Figure 2. Note that the CCK FIPS 140-2 Vendor Evidence document enumerates the CCK API's in this table along with the parameters passed to each in Appendix E.

Service	**Access (Role)**	**Accessible SRDI**	**Type of Access**	**CCK API(s)**
Installation	CO	None	Execute	--None--
Initialization	User	None	Execute	CCK_initialize
Run Self tests	User	None	Execute	CCK_self_tests, CCK_conditional_test
Show status	User	None	Execute/ Read	CCK_fips_mode, CCK_status, CCK_test_status
AES	User	Read access to keys passed as pointer parameters to constant structures; no other access.	Execute	CCK_set_AES_block_size_and_key, CCK_AES_encrypt, CCK_AES_decrypt, CCK_AES_CBC_encrypt, CCK_AES_CBC_decrypt
Triple DES	User	Read access to keys passed as pointer parameters to constant structures; no other access.	Execute	CCK_DES3_encrypt, CCK_DES3_decrypt, CCK_DES3_CBC_encrypt, CCK_DES3_CBC_decrypt
HMAC (SHA-1)	User	Read access to keys passed as pointer parameters to constant structures; no other access.	Execute	CCK_HMAC_init, CCK_HMAC_destroy, CCK_HMAC_restart, CCK_HMAC_update, CCK_HMAC_truncated_final
SHA1	User	None	Execute	CCK_SHA1_reset, CCK_SHA1_get_hash, CCK_SHA1_update, CCK_SHA1_truncate_and_report, CCK_SHA1_final_and_report, CCK_SHA1_final
RNG	User	None	Execute	CCK_X931RNG_generate_byte

Figure 2. Access to Security Relevant Data Items (SRDI) for each service and role.

Note that installation differs from initialization in that installation does not involve execution of CCK services. Initialization must occur after installation is invoked by the CCK client, and results in execution of several CCK services in the process of creating memory and starting services necessary to support subsequent CCK operation.

The inputs and outputs of each service are shown in Figure 3.

The methods of the cryptographic software module are designed to be invoked by a single process.

Service	Input/Output
Installation	Input: Installation CD Output: Installed CCK software
Initialization	Input: Installed CCK software Output: Initialized CCK software (and client application)
Run Self tests	Input: none Output: self test status
Show status	Input: none Output: module status indicator
AES encryption (ECB & CBC modes)	Input: plaintext, key, IV in CBC mode Output: ciphertext
AES decryption (ECB & CBC modes)	Input: ciphertext, key, IV in CBC mode Output: plaintext
Triple DES encryption (ECB & CBC modes)	Input: plaintext, key, IV in CBC mode Output: ciphertext
Triple DES decryption (ECB & CBC modes)	Input: ciphertext, key, IV in CBC mode Output: plaintext
HMAC (SHA-1)	Input: a file, key Output: authentication code
SHA1	Input: a file Output: hash value
RNG	Input: date/time (D/T) & seed (V) Output: a random byte

Figure 3. Inputs and outputs of each service provided by the CCK.

4. Finite State Model

The CCK uses a finite state model (FSM) to keep track of whether the module is in a valid state for performing cryptographic operations. The FSM resides in a thin layer of code between the API and the underlying cryptographic functions. The FSM guards access to all cryptographic functions and requires that the software module be properly initialized and must pass self tests before allowing cryptographic functions to be performed. It also tracks the state of conditional tests and continuous RNG tests. If any of these tests fail, the FSM goes into an error state, preventing any further cryptographic functioning.

The FSM has the states shown in Figure 4.

State	Description
FSM_CRYPTOOFFICER	Crypto officer installs the CCK
FSM_POWER_ON	Initial (startup) state - self tests not yet run
FSM_RUNNING_SELF_TESTS	Self tests are running
FSM_RUNNING_CONDITIONAL_TEST	Test of specific method being invoked from FSM_USER state
FSM_USER	Self tests have passed. Ready to accept service requests
FSM_ERROR	Self test or conditional tests failed
FSM_POWER_OFF	CCK has been installed but is not running

Figure 4. States of the Finite State Model.

5. Key Management

The CCK does not perform key generation or key establishment.

The CCK does not perform key storage. Other than the key used to decrypt the library authentication data, the CCK maintains keys only in memory and does not store keys to persistent media. Therefore it does not provide the means for key storage or retrieval between successive power up cycles of the device.

The CCK does perform key input. All key values and initial values are generated by code outside the cryptographic software boundary and are passed to the methods in the CCK by pointer reference. That is, they are stored in memory at an address allocated by code outside the cryptographic software boundary. This is then passed to the methods of the CCK.

The CCK does not perform key output. It has no key output methods nor any methods that have the effect of key output.

All secret keys and CSPs (including RNG seeds) used by the CCK are protected by the absence of any methods provided by the CCK API that enable, allow, or contribute to the disclosure, modification, or substitution (authorized or unauthorized) of any key, initial value, or seed passed into or used by the CCK.

All CSPs used internally by the CCK (i.e. not passed to the CCK), such as those used by the random number generator, are protected by being zeroized immediately after use.

However, since the CCK does not own the memory in which the keys and initial values passed to it are stored, zeroization of these keys and initial values is the responsibility of the client code that calls the CCK. The Crypto Officer could also zeroize these keys and initial values by reformatting the hard drive.

Occasionally, the memory containing keys and CSPs can be swapped by the Windows operating system to the hard drive of the PC, or by the PocketPC operating system to other persistent memory. In order to zeroize these keys and CSPs, these swap files (or memory) must be wiped by the User. Wiping the swap files can be performed by reformatting the hard drive of the Windows PC on which the swap file resides, or by initiating a hard reset of the PocketPC device. PalmOS has no such issues since it does not perform virtual memory management.

The HMAC key and signature used to validate the CCK library are protected by being AES-128 encrypted. After validation is performed, the decryption key, decrypted HMAC key, and decrypted HMAC signature are all protected by being zeroized immediately after use.

6. Module Interface

Figure 5 maps elements of the API to the four required components of the logical interface.

Logical Interface Component	**Corresponding API component**	**Physical Ports**
Data Input	API functions that accept input data arguments	Standard Input Ports (e.g., Keyboard)
Data Output	API functions that produce output in arguments and return values	Standard Output Ports (e.g., Serial Port)
Control Input	API functions to initialize and shutdown the module and to run self tests	N/A
Status Output	API functions which return information regarding module status	Standard Output Ports (e.g., Monitor)
Power	N/A	Supplied by device

Figure 5. Logical interface components, API components, physical ports.

7. Self Tests

When the CCK library is loaded, self-authentication is performed to ensure that the library has not been modified. The self-authentication is performed using HMAC (SHA-1) to compute the message authentication code of the library and is compared to an expected value. If the computed and expected values do not match, the attempt to load the library will fail. Otherwise, it will succeed. Subsequent to successful self-authentication, the CCK implements a number of self-tests to ensure that it is functioning properly. On startup, the CCK executes known answer tests (KATs) on all its cryptographic functions (listed in Figure 1) before any cryptographic functions can be executed. In addition, the required continuous RNG test ensures that the RNG generates distinct arrays of bytes on each call.

If any of these tests fail, the FSM controlling the operation of the CCK enters an error state, preventing any further functioning of the CCK. To recover from an error state, the power to the CCK must be recycled. If the CCK remains in the error state after a power recycle, the hard drive of the PC must be reformatted to ensure that all keys and CSP’s used by the CCK are zeroized or a hard reset performed on the PocketPC or Palm device. Thereafter, the CCK must be reinstalled. There are no other means for recovering from an error state.

The self-tests can be run on demand by power cycling the device running the CCK. The CCK library also contains an API, enabling the user to execute the self-tests.

8. Design Assurance

A configuration management system is used to control the versions of the source code components of the cryptographic software module. Each source file component is checked into the Concurrent Versions System (CVS). CVS is a well-known version control system that allows multiple software developers to change the same source files while maintaining records of each version and requiring resolution of conflicting changes. As part of this, CVS assigns each version of a file a unique version number. Each version of every file that is part of a commercial release of the software is tagged with a unique identifying name, and the CCK library is then built from those tagged source files.

Version information governing this Security Policy and operator documents is maintained/tracked in a version control document, which is stored in CVS. The versions of the operator documents are controlled in an archive system.

9. Secure Installation and Operation

Secure installation of the CCK must be performed by an employee playing the role of Crypto Officer at Credant Technologies or by a Crypto Officer at the company using the CCK or its associated products. Installation must be performed according to the instructions in the Installation Guide accompanying the CCK or its associated products. There is no special action the Crypto Officer must perform to ensure that the CCK is operated in FIPS mode. The CCK operates in FIPS mode by default.

Secure operation of the CCK requires that each instance of the library be used by only one user and only one user at a time. In addition, the application that constitutes the User of the CCK must call the “CCK_initialize” method to initialize the CCK. Before initialization, no cryptographic functions are available. When “CCK_initialize” is called, the self-tests are automatically invoked, and if and only if the tests pass, the module is available to perform cryptographic functions.

The operating system should also enforce single-operator mode of operation. Microsoft Windows 2000 operating system supports multiple concurrent (networked) users but only when Terminal Services is activated. These services should not be activated in order to restrict the system to single operator mode.

Please consult your IT administrator and the requisite Microsoft documentation for information on how to ensure that Terminal Services are not activated.