Dell-CREDANT Cryptographic Kernel
Mac Kernel Mode version 1.8
Mac User Mode version 1.8
Linux User Mode version 1.8

FIPS 140-2 Non-Proprietary Security Policy
Security Level 1 Validation

Revision Date: May 05, 2014

Table of Contents

List of Figures

List of Tables

1 Introduction

Dell Data Protection | Encryption offers simple, comprehensive and flexible data security for the entire organization.

1.1 Dell Data Protection | Encryption

As organizations grapple with securing endpoint devices, consumerization, globalization and workforce mobility are creating new challenges. Meanwhile, all you have to do is look at the headlines to see that threats are more coordinated and coming faster. Dell helps businesses through these challenges by enabling an easier path to data protection, compliance, and business continuity.

Dell Data Protection | Encryption (DDP|E) provides organizations with the confidence that their user's data is secure, with a solution designed for simple, comprehensive, and flexible protection. It is a policy-based solution that protects data stored on the system drive and/or external media. Designed for easy deployment, end-user transparency, and hassle-free compliance, the DDP|E portfolio of products delivers a high level of protection, fills critical security gaps and enables organizations to manage encryption policies for multiple endpoints and operating systems—all from a single management console.

The Dell-CREDANT Cryptographic Kernel (CCK) is the library of cryptographic functions used by the DDP|E suite of enhanced security solutions. The CCK takes the form of a (dynamically linked) software library, which provides an API to cryptographic functions, including AES (with ECB, CBC, CTR, CCM, and XTS modes), Triple-DES (with ECB and CBC modes), SHA-1, SHA-256, SHA-384, SHA-512, HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512, and DRBG 800-90 compliant pseudorandom number generator.

The suite is comprised of a Dell Server (Dell Data Protection Enterprise Server or Dell Data Protection Enterprise Server – Virtual Edition), Dell Policy Proxy, and Encryption components. These three components work together to ensure the security of data. The Encryption component installs on the endpoint computer and protects its data from unauthorized access. The Dell Policy Proxy receives key material and policy information from the Dell Server and communicates it to the DDP|E-protected computer. An administrator sets policies via the Dell Server using a management console, and the Dell Server forwards these policies to the instances of the Dell Policy Proxy. When a DDP|E-protected computer polls the Dell Server through the Dell Policy Proxy, it receives policy updates.

> **Note:** In this latest version, CCK has been renamed and is now known as **CmgCryptoLib**. As such, all references in this or any other document to *CCK*, *Dell-CREDANT Cryptographic Module*, or *CREDANT Cryptographic Module* should be understood to be synonymous with CmgCryptoLib.

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.

Figure I Linux User Mode Physical and Logical Cryptographic Boundaries

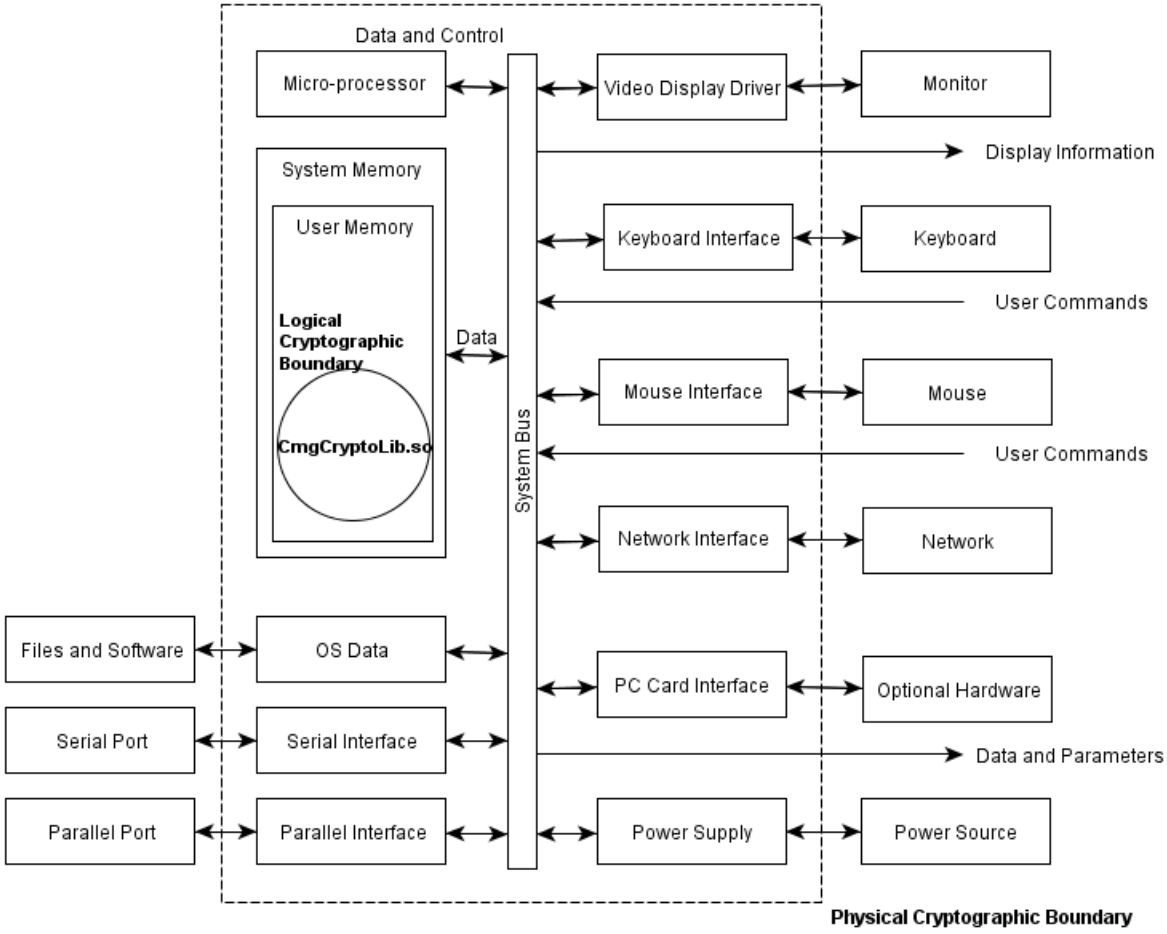


Figure II Mac User Mode Physical and Logical Cryptographic Boundaries

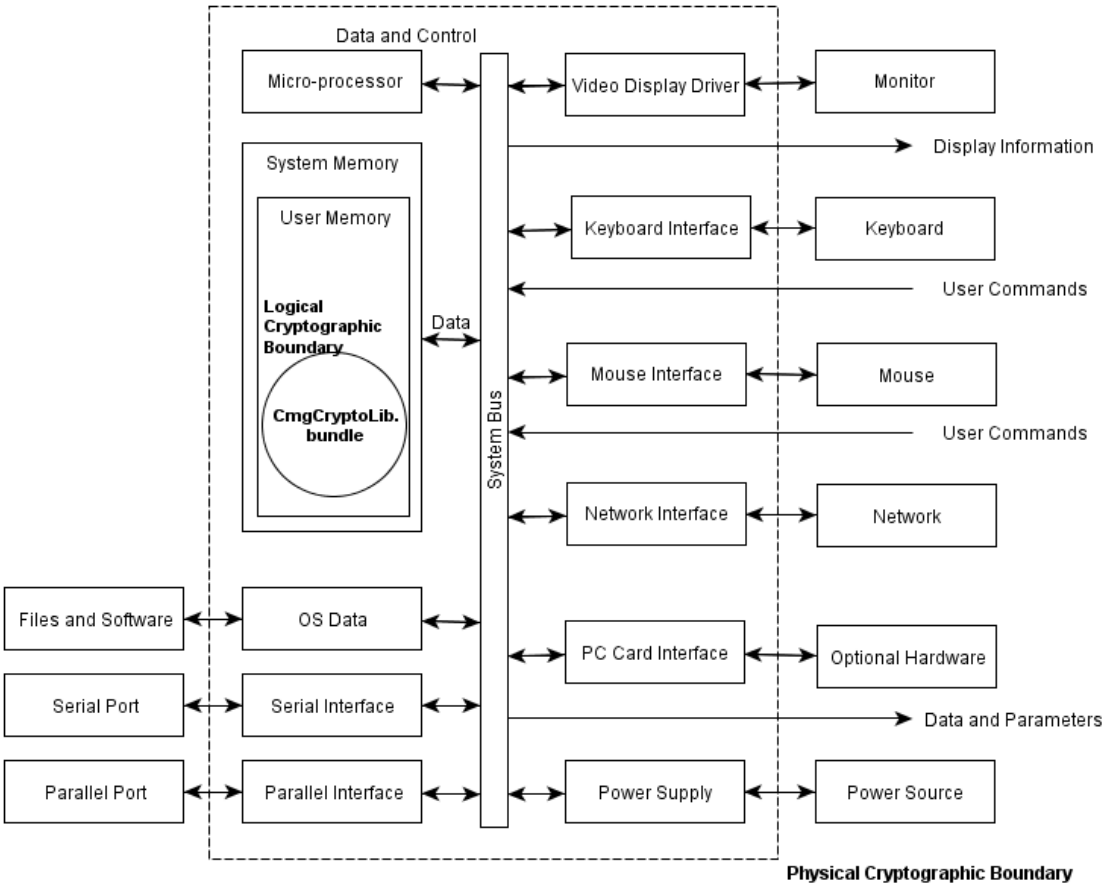
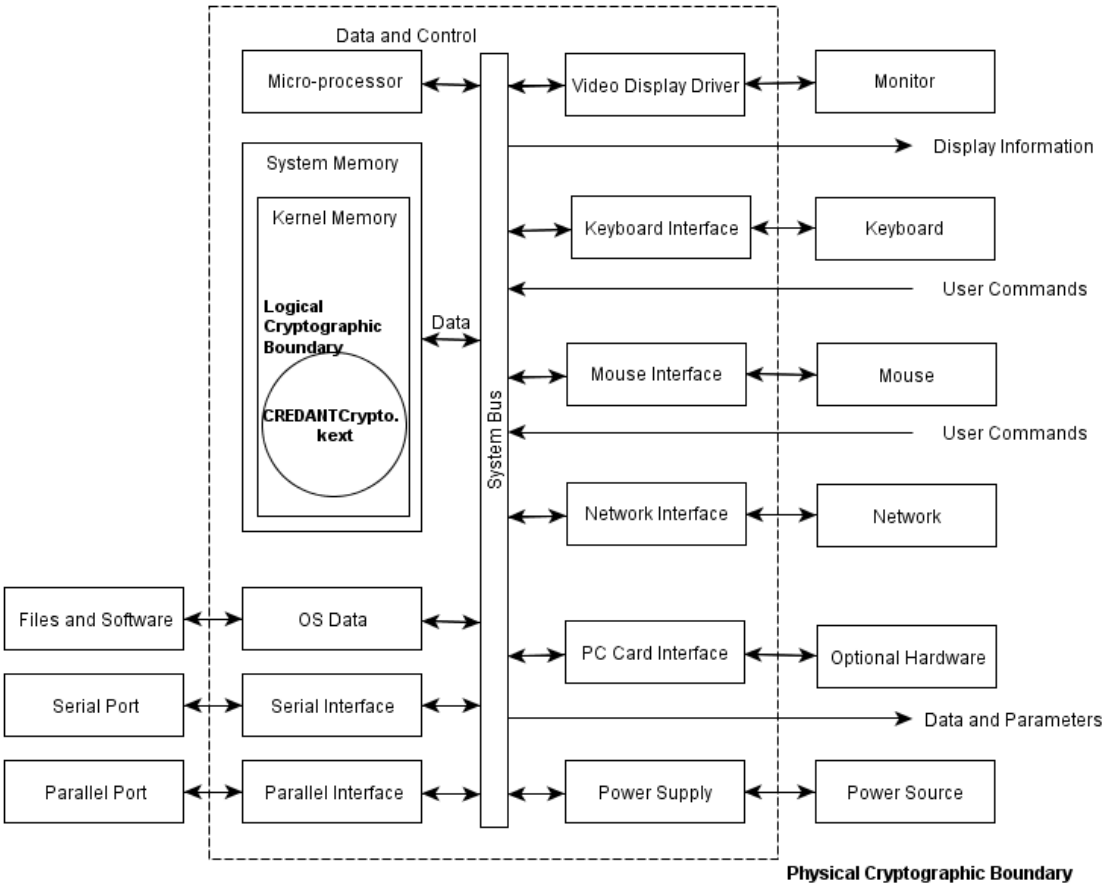


Figure III Mac Kernel Mode Physical and Logical Cryptographic Boundaries



The module constitutes a multi-chip stand-alone device (listed below), as defined by the FIPS 140-2 standard. The Dell-CREDANT Cryptographic Kernel runs in compliance with the requirements for FIPS 140-2 Level 1 security on the following configurations:

- Ubuntu Linux 11.04 (32- and 64-bit, user mode) on a Dell OptiPlex™755
- Mac OSX Lion 10.7.3 (32- and 64-bit, user and kernel mode) on a mid-2010 MacBook Pro (MacBookPro6,2)

The module also runs on the following platforms not included in this validation:

- Mac OSX 10.6 (32- and 64-bit)
- Mac OSX 10.5 (32- and 64-bit)
- Mac OSX 10.4 (32- and 64-bit)

The CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when so ported if the specific operational environment is not listed on the validation certificate.

The cryptographic module achieves an overall security level 1, with the individual level according to each security requirement shown in the table below.

Table A Security Level Achieved in Each Security Category

Security Category	Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services, and Authentication	1
Finite State Model	1
Physical Security	N\A
Operational Environment	1
Cryptographic Key Management	1
EMI\EMC	3
Self-Tests	1
Design Assurance	3
Mitigation of Other Attacks	N\A

The CCK consists of the following files:

- **CmgCryptoLib.so** – The dynamic link library containing the CCK's cryptographic functionality for Linux User mode.
- **CmgCryptoLib.mac** – The file used to support software integrity test of the file CmgCryptoLib.so for Linux User mode.
- **CmgCryptoLib.bundle** – A Mac "bundle" of the CCK library files, including the cryptographic functionality for Mac User mode and the supporting file for the software integrity test. Contains the 32- and 64-bit builds together inside the same file.
- **CREDANTCrypto.kext** – A Mac kernel extension "bundle" of the CCK library files, including the cryptographic functionality for Mac Kernel mode and the supporting file for software integrity test. Contains the 32- and 64-bit builds together inside the same file.

The following table provides an explicit mapping of the module versions, CCK file names, and tested platforms.

Table B Explicit Mapping of the Module Versions, CCK File Names, and Tested Platforms

Module Version	CCK Files	Tested Platforms
Mac Kernel Mode Version 1.8 (32-bit)	CREDANTCrypto.kext	Mac OSX Lion 10.7.3 (32-bit) on a mid-2010 MacBook Pro (MacBookPro6,2)
Mac Kernel Mode Version 1.8 (64-bit)	CREDANTCrypto.kext	Mac OSX Lion 10.7.3 (64-bit) on a mid-2010 MacBook Pro (MacBookPro6,2)
Mac User Mode Version 1.8 (32-bit)	CmgCryptoLib.bundle	Mac OSX Lion 10.7.3 (32-bit) on a mid-2010 MacBook Pro (MacBookPro6,2)
Mac User Mode Version 1.8 (64-bit)	CmgCryptoLib.bundle	Mac OSX Lion 10.7.3 (64-bit) on a mid-2010 MacBook Pro (MacBookPro6,2)
Linux User Mode Version 1.8 (32-bit)	CmgCryptoLib.so CmgCryptoLib.mac	Ubuntu Linux 11.04 (32-bit) on a Dell OptiPlex 755
Linux User Mode Version 1.8 (64-bit)	CmgCryptoLib.so CmgCryptoLib.mac	Ubuntu Linux 11.04 (64-bit) on a Dell OptiPlex 755

Note: The "CREDANTCrypto.kext" file is the same between the 32- and 64-bit platforms and the "CmgCryptoLib.bundle" file is the same between platforms. This is due to how those platforms combine 32- and 64-bit versions of the library. The "CmgCryptoLib.so" files are different between 32- and 64-bit platforms

3 Approved and Non-Approved Modes

The CCK library is capable of operating in both FIPS approved and FIPS non-approved modes. Non-approved mode generally contains the same functionality as approved mode, with the addition of extra non-approved functionality. Differences between approved and non-approved mode are noted where appropriate throughout this Security Policy. Statements about CCK are, in general, intended to apply to both modes unless otherwise noted.

The mode that the library will operate in is determined at initialization time via a parameter specified by the user. The default value, should the user not explicitly specify the mode, is approved mode. To choose the mode of operation, the CCK user creates a structure of variables used for CCK initialization. One of these variables is named "fips_mode" and can be set to the values "FIPS_MODE" or "NON_FIPS_MODE". They may optionally call a CCK API to initialize the structure, which will set "fips_mode" to "FIPS_MODE". The user can manually set the value of "fips_mode" as desired. This structure is then provided as an argument to "CCK_initialize" and the value of "fips_mode" dictates the mode the CCK is in and is stored in a similar variable within the CCK session for the lifetime of that session.

A Crypto User or Crypto Officer may check the FIPS approved verses non-approved state of the CCK session at any time. The API "CCK_fips_mode" returns the values "FIPS_MODE" or "NON_FIPS_MODE" as the state of the session.

There is no internal functionality to facilitate changing the mode between approved and non-approved modes, so a CCK session handle may only operate in one of those modes over its lifetime. No session data from an approved mode session is shared with a session in non-approved mode.

Version 7.0 of the third-party Intel® Integrated Performance Primitives (Intel® IPP) library is statically linked with the Mac user-mode CmgCryptoLib library (but not included in the Linux user-mode or Mac kernel-mode libraries) to accelerate certain cryptographic algorithms. This library is only utilized when CmgCryptoLib is in a non-approved mode. When it is in FIPS approved mode, no code from the IPP library is permitted to execute.

Table C Explicit Mapping of the Module Versions, CCK File Names, and Tested Platforms

Module Version	CCK Files	Tested Platforms
Mac Kernel Mode Version 1.8 (32- and 64-bit)	CREDANTCrypto.kext	• Mac OS X Lion 10.7.3 (32-bit) on a mid-2010 MacBook Pro (MacBookPro6,2) • Mac OS X Lion 10.7.3 (64-bit) on a mid-2010 MacBook Pro (MacBookPro6,2)
Mac User Mode Version 1.8 (32- and 64-bit)	CmgCryptoLib.bundle	• Mac OS X Lion 10.7.3 (32-bit) on a mid-2010 MacBook Pro (MacBookPro6,2) • Mac OS X Lion 10.7.3 (64-bit) on a mid-2010 MacBook Pro (MacBookPro6,2)
Linux User Mode Version 1.8 (32- and 64-bit)	CmgCryptoLib.so CmgCryptoLib.mac	• Ubuntu Linux 11.04 (32-bit) on a Dell OptiPlex 755 • Ubuntu Linux 11.04 (64-bit) on a Dell OptiPlex 755

4 Roles, Services, Policy

Table D Service Type Offered by Algorithm, FIPS Specification, and Availability

Service Type	Algorithm	FIPS Standard	Library	FIPS Algorithm Certificate Number	Available in Roles
Self-Tests	AES, AES CCM, Triple-DES, HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512, SHA-1, SHA-256, SHA-384, SHA-512, DRBG 800-90	N/A	N/A	N/A	Crypto Officer, Crypto User
Key Zeroization	N/A	N/A	N/A	N/A	Crypto Officer, Crypto User
Symmetric Cipher	AES (CTR, CCM, CBC, ECB, XTS)	197	User Mode	#2131	Crypto Officer, Crypto User
			Kernel Mode	#2130	
Symmetric Cipher	Triple-DES (ECB, CBC)	46-3	User Mode	#1354	Crypto Officer, Crypto User
			Kernel Mode	#1353	
Message Authentication	HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	198	User Mode	#1301	Crypto Officer, Crypto User
			Kernel Mode	#1300	
Message Digest	SHA-1	180-1	User Mode	#1855	Crypto Officer, Crypto User
			Kernel Mode	#1854	
Message Digest	SHA-256, SHA-384, SHA-512	180-2	User Mode	#1855	Crypto Officer, Crypto User
			Kernel Mode	#1854	
Random Number Generation	DRBG 800-90	186-2	User Mode	#236	Crypto Officer, Crypto User
			Kernel Mode	#235	

When CCK is in non-approved mode, the following also apply:

Symmetric Cipher	Rijndael (32 byte block size) with and without IPP (CBC, ECB)	197	User Mode	N/A	Crypto Officer, Crypto User
			Kernel Mode		
Symmetric Cipher	AES with IPP (CTR, CCM, CBC, ECB, XTS)	197	User Mode	N/A	Crypto Officer, Crypto User
			Kernel Mode		
Symmetric Cipher	Triple-DES with IPP (ECB, CBC)	46-3	User Mode	N/A	Crypto Officer, Crypto User
			Kernel Mode		
Message Digest	SHA-256, SHA-	180-2	User Mode	N/A	Crypto Officer,

Service Type	Algorithm	FIPS Standard	Library	FIPS Algorithm Certificate Number	Available in Roles
	384, SHA-512 with IPP		Kernel Mode		Crypto User
Random Number Generation	ANSI X9.31	186-2	User Mode Kernel Mode	N/A	Crypto Officer, Crypto User

A summary CCK of services by role and the access granted to critical security parameters of these services for each role is shown in Table E.

Note: The CCK file "CCK_api.h" enumerates the CCK APIs in this table along with the parameters passed to each. Consult the publication NIST SP 800-131A for details on recommendation for transitioning the use of cryptographic algorithms and key lengths.

Table E Access to Security Relevant Data Items (CSP) for Each Service and Role

Service	Access (Role)	Accessible CSP	Type of Access	CCK API(s)
Installation	Crypto Officer*	None	Execute	--None--
Initialization	Crypto Officer, Crypto User	AES key used to wrap the software integrity test key (read access), MAC software integrity test key (read access), Expected result of software integrity test (read access), Computed software integrity test value	Execute, Read, Write	CCK_initialize
Termination	Crypto Officer, Crypto User	Last Triple-DES key used, Last AES key used, DRBG entropy seed (read and write access), DRBG "Key" and "V" state data (read and write access), RNG seed (read and write access)	Execute, Write	CCK_terminate
Key Zeroization	Crypto Officer, Crypto User	Last Triple-DES key used, Last AES key used, HMAC key pair used in a provided HMAC context	Execute, Write	CCK_reset_encryption_decryption_algorithm_state_info, CCK_clear_AES_keys, CCK_clear_DES3_keys, CCK_HMAC_truncated_final_internal, CCK_HMAC_SHA2_final_and_report_internal

Service	Access (Role)	Accessible CSP	Type of Access	CCK API(s)
Run Self Tests	Crypto Officer, Crypto User	None	Execute	CCK_self_tests, CCK_conditional_test, CCK_set_error_state, CCK_self_auth
Show Status	Crypto Officer, Crypto User	None	Execute	CCK_fips_mode, CCK_status, CCK_test_status, CCK_role_auth_status
AES	Crypto Officer, Crypto User	AES key (read access to key passed as parameter, write access to cache the key)	Execute, Read, Write	CCK_set_AES_block_ size_and_key, CCK_set_AES_XTS_ secondary_key, CCK_AES_encrypt, CCK_AES_decrypt, CCK_AES_CBC_encrypt, CCK_AES_CBC_decrypt, CCK_AES_CTR_encrypt, CCK_AES_CTR_decrypt, CCK_AES_XTS_encrypt, CCK_AES_XTS_decrypt, CCK_AES_CCM_encrypt, CCK_AES_CCM_decrypt
Triple-DES	Crypto Officer, Crypto User	Triple-DES key (read access to key passed as parameter, write access to cache the key)	Execute, Read, Write	CCK_DES3_encrypt, CCK_DES3_decrypt, CCK_DES3_CBC_encrypt, CCK_DES3_CBC_decrypt
HMAC-SHA-1	Crypto Officer, Crypto User	HMAC-SHA-1 key pair (read access to key passed as a parameter, write access to store the derived HMAC key pair)	Execute, Read, Write	CCK_HMAC_init, CCK_HMAC_destroy, CCK_HMAC_restart, CCK_HMAC_update, CCK_HMAC_truncated_final
HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	Crypto Officer, Crypto User	HMAC key (read access to key passed as parameter, write access to store the derived HMAC key pair)	Execute, Read, Write	CCK_HMAC_SHA2_init, CCK_HMAC_SHA2_destroy, CCK_HMAC_SHA2_restart, CCK_HMAC_SHA2_update, CCK_HMAC_SHA2_final_and_report
SHA-1	Crypto Officer, Crypto 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
SHA-256, SHA-384, SHA-512	Crypto Officer, Crypto User	None	Execute	CCK_SHA2_reset, CCK_SHA2_update, CCK_SHA2_final_and_report

Service	Access (Role)	Accessible CSP	Type of Access	CCK API(s)
DRBG	Crypto Officer, Crypto User	DRBG entropy seed (read access), DRBG "Key" state data (read and write access), DRBG "V" state data (read and write access), RNG seed (read and write access)	Execute, Read, Write	CCK_DRBG_generate_bytes, CCK_DRBG_reseed_state, CCK_DRBG_check_alive CCK_DRBG_reinstantiate, CCK_perform_DRBG_test

The following only apply when CCK is in non-approved mode:

RNG	Crypto Officer, Crypto User	RNG seed, RNG seed key	Execute, Read, Write	CCK_X931RNG_generate_ Byte

Note: 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 Table F.

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

Table F Inputs and Outputs of Each Service Provided by the CCK

Service	Input/Output
Installation	Input: Installation Software Package Output: Installed CCK software
Initialization	Input: Installed CCK software Output: Initialized CCK software (and client application)
Termination	Input: Initialized CCK software Output: Uninitialized CCK software
Run Self Tests	Input: None Output: Self-test status
Show Status	Input: None Output: Module status indicator
AES encryption (ECB and CBC modes)	Input: Plaintext, key, IV in CBC mode Output: Ciphertext
AES CTR Encryption	Input: Plaintext, key, counter_block, number of bytes to increment Output: Ciphertext
AES XTS Encryption	Input: Plaintext, key, sector block size, sector number Output: Ciphertext
AES CCM Encryption	Input: Plaintext, key, associated data and length, nonce and length, mac length Output: Ciphertext, mac
AES Decryption (ECB and CBC modes)	Input: Ciphertext, key, IV in CBC mode Output: Plaintext
AES CTR Decryption	Input: Ciphertext, key, counter_block, number of bytes to increment Output: Plaintext
AES XTS Decryption	Input: Ciphertext, key, sector block size, sector number Output: Plaintext
AES CCM Decryption	Input: Ciphertext, key, associated data and length, nonce and length, mac block and length Output: Plaintext (if mac is valid)
Triple-DES encryption (ECB and CBC modes)	Input: Plaintext, key, IV in CBC mode Output: Ciphertext
Triple-DES decryption (ECB and CBC modes)	Input: Ciphertext, key, IV in CBC mode Output: Plaintext
HMAC-SHA-1	Input: A sequence of bytes, key Output: Authentication Code
HMAC-SHA-256 HMAC-SHA-384	Input: A sequence of bytes, key Output: Authentication Code

Service	Input/Output
HMAC-SHA-512	
SHA-1	Input: A sequence of bytes Output: Hash value
SHA-256 SHA-384 SHA-512	Input: A sequence of bytes, hash length Output: Hash value
DRBG	Input: --Optional-- additional seed bytes Output: A sequence of random bytes

When CCK is in non-approved mode, the following also apply:

RNG	Input: Date/time (D/T) and seed (V) Output: A random byte

4.1 Roles Related Notes

For the Level 1 Linux and Mac platforms in this validation, the only privilege unique to the Crypto Officer is ability to perform installation of the module.

4.2 Services Related Notes

The implementation of the counter used in the AES CTR implementation is a simple counter incremented by one (1), the first value of which is established by an operator-supplied initial value (a nonce). The size of the counter is 16 bytes, creating a space of 2^128, satisfying the requirement that a repeat will not occur for a long time. A stream of plain text or cipher text submitted to the AES CTR algorithm for encryption or decryption, respectively, would have to be greater than 2^128 16 byte blocks in length to trigger a repeat in the counter value.

An improved seeding source has been implemented for the purpose of providing random bytes in support of DRBG 800-90 algorithm initialization and is based on entropy from a combination of sources, each contributing in proportion to the strength of its entropy; that is, more if stronger (e.g., Unix-style entropy source "/dev/random") and less if weaker (e.g., system time).

4.3 Authentication Related Notes

The cryptographic module does not support any methods of authenticating the operator on the Mac and Linux platforms since it is not required for Level 1.

Within the CCK module, all operators are implicitly made a Crypto User. Whether an operator is a Crypto Officer is determined by the operating system's permissions configuration.

5 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. 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 Table G.

Table G States of the Finite State Model

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

6 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. The aforementioned key, an AES 256 key is stored in an obfuscated format compiled into the module, and is zeroed out after each use (the key still remains persisted in the obfuscated format, but is not persisted in plaintext in memory or in the binary). As such, it does not provide means for key storage or retrieval between successive power-up cycles of the device.

The HMAC key and integrity value used to validate the CCK library are protected by being AES-256 encrypted. The key used for the encryption of the HMAC key and integrity value is hardcoded in an obfuscated manner in the module, and is only loaded into memory in plaintext form when needed to decrypt the validation data contained in the MAC file. After validation is performed, the decryption key, decrypted HMAC key, and decrypted HMAC integrity value are all protected by being zeroized immediately after use.

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 and 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.

Since keys and initial values passed into the CCK are stored in memory allocated by code outside the cryptographic software boundary, zeroization of these artifacts is the responsibility of the code calling the CCK. The Crypto Officer could also zeroize these artifacts by reformatting the hard drive.

Occasionally, the memory containing keys and CSPs can be swapped by the operating system to the hard drive of the computer. In order to avoid availability of these keys and CSPs in plain text, these swap files must either be wiped by the operator or encrypted. When the CCK is executing in Dell's Mac-based products, the swap files are encrypted.

7 Random Number Generation

The CCK implements three random number generators (RNG), summarized in Table H:

Table H RNGs Implemented Within the CCK

RNG	Deterministic/Non-Deterministic	Validated
SP800-90 AES-CTR	Deterministic	Yes
X9.31	Deterministic	No
Generic entropy gathering seed source	Non-deterministic	No

7.1 RNG Seeding

The SP800-90 DRBG is seeded by a combination of the generic entropy seed source and optional user input, both provided at seeding time. This is in accordance with the SP800-90 specification which allows for the seeding process to use a default seeding mechanism and still allow the caller to supplement seeding material as they see fit. This ensures that the quality of the seeding material is not left exclusively to either party. It was implemented with the specification's DF "smoothing" function that transforms a large seed buffer of raw entropy to a suitable working state.

The X9.31 algorithm is seeded by the SP800-90 DRBG. It is not certified and thus not seeded or permitted to execute when the CCK is in FIPS mode.

The generic entropy gathering seed source gathers entropy non-deterministically from a variety of entropy sources. The entropy sources use the current CCK module execution environment and the host platform RNG sources. It is not an implementation of an Approved NDRNG (as there are no Approved NDRNGs at this time). The seeding sources vary by availability on the active platform and the available sources can be individually turned on or off (meaning that they do or do not contribute to the entropy gathering, respectively) by the user at the CCK API level. The entropy collected by this function is used to seed the SP800-90 DRBG, which has its own appropriate entropy "smoothing" function. The entropy gathered is destroyed after seeding the DRBG and is never made available outside of the CCK boundaries.

7.2 RNG Tests

All RNGs run a continuous test (CRNG) on their output to ensure that each output block is distinct from the previous one. A few of the specific entropy sources have their own additional tests to continuously test for failures like long repeated runs or default to zero.

The SP800-90 DRBG and X9.31 RNG have known-answer tests that execute as a part of the CCK library self-tests. A few of the generic entropy gatherer's sources have self-tests that check that entropy is available from the source and passes basic conditions.

8 Module Interface

Table I maps elements of the API to the four required components of the logical interface.

Table I Logical Interface Components, API Components, Physical Ports

Logical Interface Component	Corresponding API Component	Physical Ports
Data Input	API functions that accept input data arguments	Serial port, parallel port, network port, mouse port, PC card interface, keyboard port
Data Output	API functions that produce output in arguments and return values	Serial port, parallel port, network port, PC card interface, video display port
Control Input	API functions to initialize and shut down the module and to run self-tests	Serial port, parallel port, network port, mouse port, PC card interface, keyboard port, power switch
Status Output	API functions which return information regarding module status	Serial port, parallel port, network port, PC card interface, video display port
Power	N/A	Power Supply

9 Self-Tests

When the CCK library is initialized for the first time in approved mode, the software integrity test is performed to ensure that the library has not been modified. The test is not performed when CCK is in non-approved mode. The software integrity test uses HMAC-SHA-512 to compute the message authentication code of the library and compares the digest to an expected value. If the computed and expected values do not match, the attempt to initialize the library will fail, as will all subsequent initialization attempts until the library is re-loaded. Otherwise, it will succeed. This method of self-test applies to all versions of the cryptographic module, except for the Mac Kernel Extension. Following successful software integrity test, 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 Table D) before any cryptographic functions can be executed. In addition, the required continuous RNG test for each RNG ensures that it generates distinct arrays of bytes on each call.

The Mac Kernel Extension (KEXT) version of the cryptographic module is needed at such an early point in the boot process that the file system does not exist. As such, the aforementioned method cannot be used to perform the self-test as it cannot find either the file storing the expected HMAC value or its own binary to perform the HMAC on itself.

For this reason the Mac KEXT version of the library must find itself loaded in memory to compute the HMAC on itself. This approach, however, results in the operating system rearranging or relocating symbols as it loads the module into memory. The solution to this is to create a copy of the module in memory and, with the support of the relocation data generated at compile time, relocate all of the symbols back to their original compiled base locations prior to computing the HMAC.

With this approach, the Mac KEXT self-test function must be provided the expected HMAC value, as well as the relocation data generated at compile time, in the form of a BLOB. This data is provisioned at install time so that it is available to the operator to pass it to the module at initialization time. Once the internal self-test function is called, the cryptographic module finds itself in memory, makes a copy, “unrelocates” the copy, and performs the HMAC on the relevant sections of the copy, comparing the result to the expected value. Until this self-test passes, the module cannot be initialized.

The module implements the following self-tests:

Power-up Tests

- AES
  - ECB encryption and decryption KAT
  - CBC encryption and decryption KAT
  - CTR encryption and decryption KAT
  - XTS encryption and decryption KAT
  - CCM encryption and decryption KAT
- Triple-DES
  - ECB encryption and decryption KAT
  - CBC encryption and decryption KAT
- DRBG
  - KAT
- Software Integrity
  - Integrity self-test of the CCK software module using HMAC-SHA-512
- SHA-1
  - Short message KAT
  - Long message KAT
- SHA-256
  - Short message KAT
  - Long message KAT
- SHA-384
  - Short message KAT
  - Long message KAT
- SHA-512
  - Short message KAT
  - Long message KAT
- HMAC-SHA-1
  - Short message KAT
  - Long message KAT
- HMAC-SHA-256
  - Short message KAT
  - Long message KAT
- HMAC-SHA-384
  - Short message KAT
  - Long message KAT
- HMAC-SHA-512
  - Short message KAT
  - Long message KAT

When CCK is in non-approved mode, the following also apply:

- X9.31 RNG
  - KAT
  - Continuous Test

If an error is encountered during power-up, then a descriptive error code is set inside of the initialization parameter provided by the module operator. The error code will be any of the codes unequal to CCK_INIT_SUCCESS.

Conditional Tests

- DRBG
  - Continuous Test
- NDRNG
  - Continuous Test

When CCK is in non-approved mode, the following also apply:

- X9.31 RNG
  - Continuous Test

If a conditional test encounters an error, then the API that triggered the error will return “false” (if it has a return value) and the internal state will enter FSM_ERROR. This status can be retrieved via the dedicated API call that queries the internal FSM state.

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. See Section 10 for further details.

The self-tests can be run on demand by power-cycling the device running the CCK. The CCK library also contains an API, available only to Crypto Officers, which enables the operator to execute the self-tests.

10 Secure Delivery, Installation and Operation

When a customer purchases Dell software that includes the CCK library, they are able to obtain the software only by secure download from a site which requires authentication and sends the installation files over an encrypted data stream.

Secure installation of the CCK must be performed by an employee playing the role of Crypto Officer at Dell Inc. or by a Crypto Officer at the organization 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.

The host operating system must be configured to a single user mode of operation.

Secure operation of the CCK requires that each instance of the library be used by one and only one operator at a time. The steps to initialize the library depend on the platform.

10.1 Linux User Mode and Mac User Mode Startup Procedures

The application that constitutes the operator 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.

10.2 Mac Kernel Mode Startup Procedures

In the Mac KEXT version of the CCK module, the application that constitutes the operator of the CCK must call the “CCK_auth_mac” to initiate the integrity self-test. If and only if this function succeeds in validating the Mac KEXT integrity, the operator can call “CCK_initialize” to begin using the cryptographic module. Until the “CCK_initialize” function is successfully called, no cryptographic functions are available to the operator application.

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 computer must be reformatted to ensure that all keys and CSPs used by the CCK are zeroized. Thereafter, the CCK must be reinstalled. There are no other means for recovering from an error state.

After the CCK has been uninstalled, the hard disk that the CCK was on should be either a.) reformatted using a full disk format that overwrites the contents of the disk or b.) overwritten using software that overwrites the entire disk with new data. This process should be performed at least once.