Security Policy

RedCannon Cryptographic Module v1.3.0

FIPS 140-2

Level 1 Validation

Version 1.3

October 20, 2005

Table of Contents

1. INTRODUCTION 3

1.1 PURPOSE 3
1.2 REFERENCES 3

2. REDCANNON CRYPTO MODULE VERSION 1.3.0 4

2.1 OVERVIEW 4
2.2 CRYPTOGRAPHIC MODULE 4
2.3 MODULE PORTS AND INTERFACES 5
2.4 ROLES, SERVICES AND AUTHENTICATION 5
2.5 PHYSICAL SECURITY 6
2.6 OPERATIONAL ENVIRONMENT 6
2.7 CRYPTOGRAPHIC KEY MANAGEMENT 6
2.8 SELF-TESTS 10
2.9 DESIGN ASSURANCE 10
2.10 MITIGATION OF OTHER ATTACKS 10

3. OPERATION OF THE REDCANNON CRYPTO MODULE VERSION 1.3.0 11

4. ACRONYM LIST 12

1. Introduction

1.1 Purpose

This is a non-proprietary Cryptographic Module Security Policy for the RedCannon Crypto Module Version 1.3.0. This security policy describes how the RedCannon Crypto Module Version 1.3.0 meets the Level 1 security requirements of FIPS 140-2 and while tested on Microsoft Windows XP Professional with Service Pack 2, it is a cross-platform module also capable of running on Microsoft Windows versions 98/98SE/Me/NT4/2000. This policy was prepared as part of FIPS 140-2 validation of the RedCannon Crypto Module Version 1.3.0.

FIPS 140-2 (Federal Information Processing Standards Publication 140-2 — Security Requirements for Cryptographic Modules) details the U.S. Government requirements for cryptographic modules. More information about the FIPS 140-2 standard and validation program is available on the NIST website at http://csrc.nist.gov/cryptval/.

1.2 References

This document deals only with operations and capabilities of the RedCannon Crypto Module Version 1.3.0 in the technical terms of a FIPS 140-2 cryptographic module security policy. More information is available on the RedCannon Crypto Module Version 1.3.0 application from the following sources:

- Overview information of RedCannon products and services as well as answers to technical or sales related questions, refer to: http://www.redcannon.com

2. RedCannon Crypto Module Version 1.3.0

2.1 Overview

The RedCannon Crypto Module Version 1.3.0 (referenced hereafter as the crypto module) provides cryptographic support for the RedCannon line of products. The crypto module is used to create, manage and delete cryptographic keys as well as to perform cryptographic operations.

To provide security cryptographic services, the crypto module provides access to both symmetric and asymmetric key based encryption algorithms, message digest, message authentication code, RSA signature generation and verification, and pseudo random number generation functions.

The crypto module can be used for multiple functions within the RedCannon applications. It provides a structured set of APIs, which can be called to perform these functions. This provides flexibility for the module and the ability to add new applications for the crypto module functions in the future without changing the module itself.

Utilizing the crypto module, RedCannon applications can create encryption keys, which can then be used to encrypt data. The APIs provide the ability to encrypt both static data (such as files) as well as data streams (such as VPN traffic). The crypto module also provides the ability to perform cryptographic MAC operations and Message Digest operations.

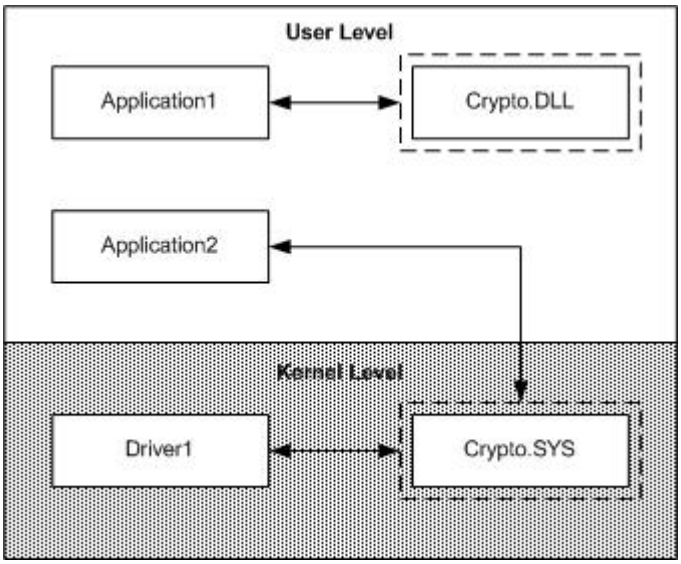

Figure 1 Interaction of different forms of the Crypto module with other applications

2.2 Cryptographic Module

The RedCannon Crypto Module Version 1.3.0 is classified as a multi-chip standalone module for FIPS 140-2 purposes. The cryptographic module is capable on running on any commercially available IBM compatible PC under Windows 98/98SE/Me/NT4/2000/XPOperating Systems (OS). The module was tested for FIPS compliance on a GPC running Windows XP Professional configured in the single user mode.

The module is compiled into three forms, allowing for flexibility in use: a Kernel-level driver, a Windows DLL for external application use, and a self-decrypting executable for decrypting files on non-RedCannon protected systems, though only one of these forms is in use at any one time. The self-decrypting executable is not included as a tested component of the FIPS 140-2 validation.

2.3 Module Ports and Interfaces

The RedCannon Crypto Module Version 1.3.0 is classified as a multi-chip standalone module for FIPS 140-2 purposes. As such, the module's cryptographic boundary include binaries viz rccd.sys (Kernel-level driver) or rccd.dll (DLL), a folder to store user based keys (the location of this folder is determined at runtime by the application using the crypto module), a PC running an operating system and interfacing with the computer, keyboard, mouse screen, floppy drive, CD-ROM drive, speaker, serial ports, parallel ports, and power plug.

The RedCannon Crypto Module Version 1.3.0 provides a logical interface via an Application Programming Interface (API). The API provided by the module is mapped to the FIPS 140-2 logical interfaces: data input, data output, control input, and status output. All of these physical interfaces are separated into the logical interfaces from FIPS as described in the following table:

FIPS 140-2 Logical Interface	Module Mapping
Data Input Interface	Parameters passed to the module via API calls
Data Output Interface	Data returned by the module via the API
Control Input Interface	Control input through the API function calls
Status Output Interface	Information returned via exceptions and calls
Power Interface	Does not provide a separate power or maintenance access interface beyond the power interface provided by the computer itself

Table 1 – FIPS 140-2 Logical Interfaces

2.4 Roles, Services and Authentication

The cryptographic module provides Crypto Officer (CO) and User roles. All the services exported by the module are common to both the roles except key zeroization. Only the Crypto-officer is allowed to perform key zeroization. Since the module is validated at security level 1, it does not provide an authentication mechanism.

The table below lists all the services provided by the module and the role to which each service belongs.

Exported Services	Role
CRYPT_REQ_ENCRYPT	User/CO
CRYPT_REQ_CREATE_DIGEST_CONTEXT	User/CO
CRYPT_REQ_DELETE_DIGEST_CONTEXT	User/CO
CRYPT_REQ_DIGEST_UPDATE	User/CO
CRYPT_REQ_DIGEST_FINAL	User/CO
CRYPT_REQ_CREATE_HMAC_CONTEXT	User/CO
CRYPT_REQ_DELETE_HMAC_CONTEXT	User/CO
CRYPT_REQ_HMAC_UPDATE	User/CO
CRYPT_REQ_HMAC_FINAL	User/CO
CRYPT_REQ_GENERATE_RSA_KEYS	User/CO
CRYPT_REQ_DH_GENERATE_KEY_AGREEMENT	User/CO
CRYPT_REQ_DH_COMPUTE_KEY	User/CO
CRYPT_REQ_REGISTER_USER	User/CO
CRYPT_REQ_UNREGISTER_USER	User/CO
CRYPT_REQ_REGISTER_MODULE	User/CO
CRYPT_REQ_UNREGISTER_MODULE	User/CO
CRYPT_REQ_INSERT_PASSWD	User/CO
CRYPT_REQ_GENERATE_RANDOM_KEY	User/CO
CRYPT_REQ_INSERT_KEY	User/CO

CRYPT_REQ_REMOVE_KEY	CO
CRYPT_REG_GET_REFERENCE_COUNT *	User/CO
CRYPT_REQ_REGISTER_APP	User/CO
CRYPT_REQ_UNREGISTER_APP	User/CO
CRYPT_REQ_CREATE_KEY_FILE	User/CO
CRYPT_REQ_LOAD_KEY_FILE	User/CO
CRYPT_REQ_INSERT_RECOVERY_KEY	User/CO
CRYPT_REQ_DELETE_FILE	CO
CRYPT_REQ_CHANGE_KEY	User/CO
CRYPT_REQ_GET_FIRST_KEY_PARAM	User/CO
CRYPT_REQ_GET_NEXT_KEY_PARAM	User/CO
CRYPT_REQ_GET_KEY_PARAM	User/CO
CRYPT_REQ_GENERATE_RAND_NUM	User/CO
CRYPT_REQ_GET_CRYPTO_VERSION	User/CO

Table 2- Services and Authorized Roles

* Available only incase of the driver form of the module

2.5 Physical Security

Since the module is implemented solely in software, the physical security section of FIPS 140-2 is not applicable.

2.6 Operational Environment

The RedCannon Crypto Module Version 1.3.0 is implemented as a loadable module that is run (without modification) on Microsoft's Windows 98/98SE/Me/NT4/2000/XPplatforms. The module is compiled into three forms, allowing for flexibility in use: a Kernel-level driver, a Windows DLL for external application use and self decrypting executable for decrypting files on non RedCannon protected systems. The module is specifically tested on Windows XP Professional. The module is always distributed in its binary form to discourage unauthorized modification. Additionally, a software integrity check is used within the module to help ensure that the code has not been accidentally or ineptly modified from its validated configuration.

2.7 Cryptographic Key Management

The RedCannon Crypto Module Version 1.3.0 implements the following algorithms. The FIPS approved column specifies whether the algorithm is available in the FIPS-mode (non-approved algorithms, except DH, are not available).

Algorithm	FIPS Approved
DH[1]	No
RSA(signature generation, signature verfication and key generation) (1024)	Yes
RSA (Key Generation) (non-compliant)	No
AES (CBC - 128, 192, 256)	Yes
DES (CBC)[2]	Yes
TDES (CBC)	Yes
TWOFISH (CBC 128, 192, 256)	No
BLOWFISH (CBC 32 - 448)	No
SERPENT (CBC 128, 192, 256)	No
CAST (CBC 128)	No
MD5 (128)	No
SHA-1 (160)	Yes
SHA-2 (256, 512)	Yes
HMAC SHA-1	Yes
HMAC SHA-256	Yes
HMAC SHA-512	Yes
HMAC MD5	No
ANSI X9.31 PRNG	Yes

Table 3 List of algorithms

Keys are generated by using the ANSI X9.31 PRNG or are electronically input from a user for symmetric key algorithms. Electronic key entry is supported through the crypto module's API, and keys are entered directly into the module. The module also supports the generation of public-private key pairs for RSA and DH algorithms.

The module provides two options for generating RSA key pairs:

1. FIPS approved RSA key generation: In this case the module implements the FIPS approved ANSI X9.31 standard to generate RSA key pairs.
2. Non-FIPS approved RSA key generation: This is an alternative method provided by the module to generate RSA key pairs. This method is not FIPS approved and as such keys generated using this method must not be used in FIPS mode.

The following list of keys is used by the module. They are generated or inserted as specified and stored within the module as necessary.

Name	Created	Size(s) in bits	Purpose
AES_key	Inserted/Generated	128, 256, 192	Encryption, Decryption
DES_key	Inserted/Generated	64	Encryption, Decryption
3DES_key	Inserted/Generated	128,192	Encryption, Decryption
RSA_Private_key	Inserted/Generated	1024 mod size	Signing, Verifying, Encryption, Decryption
RSA_Public_key	Inserted/Generated	1024 mod size	Signing, Verifying, Encryption[3], Decryption[3]
DH_key[1]	Inserted/Generated	768, 1024, 1536, 2048, 3072, 4096	Key Establishment

1 Although DH is a non-FIPS approved algorithm, it is allowed to be used in the FIPS mode with bit sizes 1024 or greater. The key establishment methodology provides between 80-bits and 150-bits of encryption strength.
2 For legacy system use only. Transitional phase only - valid until May 19, 2007
3 RSA key transport – as per PKCS#1 - is allowed to be used in the FIPS mode. The key establishment methodology provides 80-bits of encryption strength.

HMAC_SHA1_key	Inserted/Generated	Variable	HMAC creation
HMAC_SHA256_key	Inserted/Generated	Variable	HMAC creation
HMAC_SHA512_key	Inserted/Generated	Variable	HMAC creation
HMAC_SHA512_IT_key	Hard-coded	256	Software Integrity Testing
PRNG_key1 (TDES Key)	Generated	64	PRNG Generation
PRNG_key2 (TDES Key)	Generated	64	PRNG Generation

Table 4 List of keys

RedCannon Cryptographic Module SRDI/Role/Service Access Policy	Security Relevant Data Item	HMAC_SHA1_key	HMAC_SHA256_key	HMAC_SHA512_key	HMAC_SHA512_IT_key	PRNG_key1 (TDES Key)	PRNG_key2 (TDES Key)	DH_key	RSA_Public_key	RSA_Private_key	3DES_key	DES_key	AES_key
Role/Service													
Crypto-Officer Role (Only)													
CRYPT_REQ_DELETE_FILE		d	d	d	-	-	-	d	d	d	d	d	d
CRYPT_REQ_REMOVE_KEY		d	d	d	-	-	-	d	d	d	d	d	d
Crypto-officer/User Role													
CRYPT_REQ_ENCRYPT		-	-	-	-	-	-	r/x	r/x	r/x	r/x	r/x	r/x
CRYPT_REQ_CREATE_DIGEST_CONTEXT		-	-	-	-	-	-	-	-	-	-	-	-
CRYPT_REQ_DELETE_DIGEST_CONTEXT		-	-	-	-	-	-	-	-	-	-	-	-
CRYPT_REQ_DIGEST_UPDATE		-	-	-	-	-	-	-	-	-	-	-	-
CRYPT_REQ_DIGEST_FINAL		-	-	-	-	-	-	-	-	-	-	-	-
CRYPT_REQ_CREATE_HMAC_CONTEXT		r/x	r/x	r/x	-	-	-	-	-	-	-	-	-
CRYPT_REQ_DELETE_HMAC_CONTEXT		-	-	-	-	-	-	-	-	-	-	-	-
CRYPT_REQ_HMAC_UPDATE		x	x	x	-	-	-	-	-	-	-	-	-
CRYPT_REQ_HMAC_FINAL		x	x	x	-	-	-	-	-	-	-	-	-

CRYPT_REQ_GENERATE_RSA_KEYS		-	-	-	-	-	-	-	w	w	-	-	-
CRYPT_REQ_DH_GENERATE_KEY_AGREEMENT		-	-	-	-	-	-	-	-	-	-	-	-
CRYPT_REQ_DH_COMPUTE_KEY		-	-	-	-	-	-	w	-	-	-	-	-
CRYPT_REQ_REGISTER_USER		-	-	-	-	-	-	-	-	-	-	-	-
CRYPT_REQ_UNREGISTER_USER		-	-	-	-	-	-	-	-	-	-	-	-
CRYPT_REQ_REGISTER_MODULE		-	-	-	-	-	-	-	-	-	-	-	-
CRYPT_REQ_UNREGISTER_MODULE		-	-	-	-	-	-	-	-	-	-	-	-
CRYPT_REQ_INSERT_PASSWD		w	w	w	-	-	-	-	-	-	w	w	w
CRYPT_REQ_GENERATE_RANDOM_KEY		w	w	w	-	-	-	-	-	-	w	w	w
CRYPT_REQ_INSERT_KEY		w	w	w	-	-	-	w	w	w	w	w	w
CRYPT_REG_GET_REFERENCE_COUNT		-	-	-	-	-	-	-	-	-	-	-	-
CRYPT_REQ_REGISTER_APP		-	-	-	-	-	-	-	-	-	-	-	-
CRYPT_REQ_UNREGISTER_APP		-	-	-	-	-	-	-	-	-	-	-	-
CRYPT_REQ_CREATE_KEY_FILE		w	w	w	-	-	-	w	w	w	w	w	w
CRYPT_REQ_LOAD_KEY_FILE		r	r	r	-	-	-	r	r	r	r	r	r
CRYPT_REQ_INSERT_RECOVERY_KEY		-	-	-	-	-	-	-	w	-	-	-	-
CRYPT_REQ_CHANGE_KEY		w	w	w	-	-	-	w	w	w	w	w	w
CRYPT_REQ_GET_FIRST_KEY_PARAM		r	r	r	-	-	-	r	r	r	r	r	r
CRYPT_REQ_GET_NEXT_KEY_PARAM		r	r	r	-	-	-	r	r	r	r	r	r
CRYPT_REQ_GET_KEY_PARAM		r	r	r	-	-	-	r	r	r	r	r	r
CRYPT_REQ_GENERATE_RAND_NUM		-	-	-	-	w/x	w/x	-	-	-	-	-	-
CRYPT_REQ_GET_CRYPTO_VERSION		-	-	-	-	-	-	-	-	-	-	-	-

r: read w: write x: execute d: delete

Table 5 Key details

All keys are encrypted before they are stored, regardless of the purpose of the key. Keys are encrypted with a key derived from the user's password, and as such these keys are considered to be plaintext for FIPS purposes. Keys are stored in the module's internal data structures, which are not exposed to external access. Keys are assigned unique identifiers for external access, allowing external modules to access keys without exposing the key itself. When keys are set for deletion using the CRYPT_REQ_DELETE_FILE service, the key file is zeroized by overwriting the key file multiple times (if specified by the user) to ensure it cannot be retrieved. If the key is deleted using the CRYPT_REQ_REMOVE_KEY service, the key is overwritten once with zeros.

Please note that the HMAC_SHA512_IT_key key can be deleted by deleting the module from the host computer.

2.8 Self-Tests

The RedCannon Crypto Module Version 1.3.0 performs several power-up self-tests including known answer tests for the algorithms (AES, DES, 3DES, SHA-1, SHA-256, SHA-512, HMAC-SHA-1, HMAC-SHA256, HMAC-SHA512, PRNG and RSA). The crypto module also performs a self-integrity check using HMAC-SHA-512 to verify the module has not been damaged or tampered with.

The crypto module performs two conditional tests: continuous tests on the PRNG (approved as well as non-approved) each time it is used to generate random data, and a pair-wise consistency test each time the module generates RSA key pairs.

Algorithm	Power-Up Self Tests	Conditional Self Tests
AES KAT	Yes	No
DES KAT	Yes	No
TDES KAT	Yes	No
SHA-1 KAT	Yes	No
SHA-256 KAT	Yes	No
SHA-512 KAT	Yes	No
HMAC-SHA-1 KAT	Yes	No
HMAC-SHA-256 KAT	Yes	No
HMAC-SHA-512 KAT	Yes	No
RSA	Yes	Yes
PRNG	Yes	Yes

Table 5 List of self-tests

2.9 Design Assurance

RedCannon maintains versioning for all source code and associated documentation through Microsoft Visual SourceSafe 6.0.

2.10 Mitigation of Other Attacks

The RedCannon Crypto Module Version 1.3.0 does not employ security mechanisms to mitigate specific attacks.

3. Operation of the RedCannon Crypto Module Version 1.3.0

The RedCannon Crypto Module Version 1.3.0 contains both FIPS-approved and non-FIPS-approved algorithms. FIPS-mode execution is determined by the algorithm being invoked. In FIPS mode only Approved algorithms must be used. For example, if AES is being used to encrypt some plaintext, then the module is operating in FIPS-mode, but if the Twofish algorithm was being used, it would not be in FIPS-mode.

Additionally, keys generated in a FIPS mode of operation must not be used while accessing non-FIPS approved services and vice-versa. To ensure this, the user must zeroize all keys while transitioning between modes.

The RedCannon Crypto Module Version 1.3.0 is designed for installation and use on a computer configured in single user mode, and is not designed for use on systems where multiple, concurrent users are active.

4. Acronym List

Acronym	Definition
3DES	Triple Data Encryption Standard
AES	Advanced Encryption Standard
DES	Data Encryption Standard
DH	Diffie-Hellman
DSA	Digital Signature Algorithm
MD2	Message Digest Algorithm 2
MD4	Message Digest Algorithm 4
MD5	Message Digest Algorithm 5
RC4	Rivest's Code 4
RSA	Rivest, Shamir, Adleman
SHA	Secure Hash Algorithm