Hummingbird Cryptographic Module version 1.0

FIPS 140-2 Non-Proprietary Security Policy version 1.11 July 6, 2007

Revision History

Revision Number	Date	Details
1	October 20, 2006	Initial Version
2	November 3, 2006	revisions, still in draft
3	January 22, 2007	Inserted Algorithm Certificate Numbers
4	February 2, 2007	Updates deriving from functional testing Still in draft status.
5	February 5, 2007	Enabled document revision numbering
6	February 19, 2007	Edits based on testing lab comments
7	February 22, 2007	Changed name of module
8	February 27, 2007	Revised comment on non-FIPS crypto
9	February 28, 2007	First page footer
10	July 3, 2007	Added list of references, other minor changes
11	July 5, 2007	Minor changes regarding RSA key establishment, key sizes

Table of Contents

1. Introduction

1.1 Overview

This document is the non-proprietary FIPS 140-2 security policy for the *Hummingbird Cryptographic Module* which meets the FIPS 140-2 level 1 requirements. This document is a required part of the FIPS 140-2 validation process.

This Security Policy details the secure operation of the *Hummingbird Cryptographic Module* as required in Federal Information Processing Standards Publication 140-2 (FIPS 140-2) as published by the National Institute of Standards and Technology (NIST) of the United States Department of Commerce.

This Security Policy addresses technical aspects as required for the validation of a FIPS 140-2 cryptographic module, including the required operations and capabilities as required by FIPS 140-2. It is available online at the NIST Cryptographic Module Validation website, http://csrc.nist.gov/cryptval/140 -1/1401val2006.htm .

- The Open Text Corporation website (http://www.Open Text.com) contains information on the products available from Open Text Corporation.

- Refer to the NIST Validated Modules website (http://csrc.nist.gov/cryptval/1401/1401val2006.htm) for contact information regarding sales or technical information.

1.2 Terminology

Hereafter the Hummingbird Cryptographic Module will simply be referred to as the Cryptographic Module or the Module. This document will be referred to as the Security Policy.

2. Module Specification

2.1 Overview

The Hummingbird Cryptographic Module is a shared library in the form of a DLL that is used in Hummingbird's Connectivity Product Line.

The Module supports Connectivity Software such as FTP for Windows, HostExplorer, Exceed, and Connectivity Secure Shell. The cryptographic capabilities of the library are used to implement encryption and decryption services, as well as protocols such as SSL and SSH.

Exceed (the Windows-based X11 server software) can use Connectivity Secure Shell as a secure transport. FTP for Windows supports SSL sessions as well as SSH transport.

HostExplorer supports SSL and SSH as well. Note that although not FIPS compliant, these applications also support Kerberized connections.

For the purposes of FIPS- 140- 2 validation the Module is classified as a *multi-chip stand-alone Module*.

2.2 Cryptographic Boundary

The logical cryptographic boundary for the Module is the library itself. An in-core memory cryptographic digest (HMAC-SHA-1) is computed on the Cryptographic Module memory image and compared to a pre-computed digest value inserted in the Module at build time in order to verify that it has not been altered, as part of the power up self test.

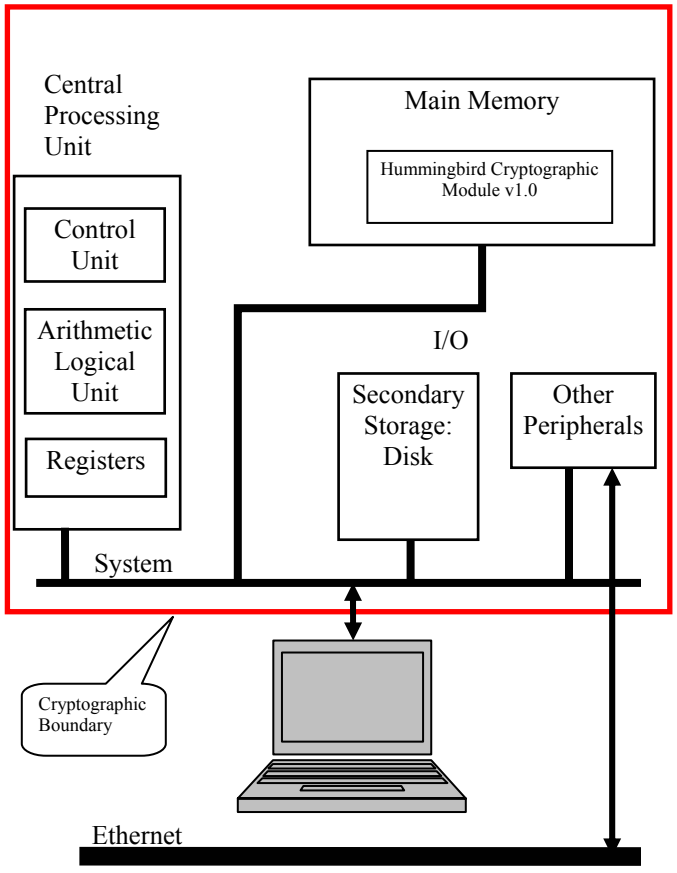

Figure 1 Block Diagram of the Cryptographic Module

The physical cryptographic boundary is the computer hardware itself, including the CPU, memory, system bus and all peripherals: network devices, keyboards, and console. The

Cryptographic Module only communicates with the application program that calls it – it creates no files and does not communicate with other processes, either via inter-process or network communication.

This is shown in *Figure 1 Block Diagram of the Cryptographic Module*.

The Cryptographic Module provides: confidentiality, integrity, key establishment, and message digest services via the algorithms in *Table 1 Cryptographic Algorithms Available in FIPS Mode*

Table 1 Cryptographic Algorithms Available in FIPS Mode

Algorithm	Standard	FIPS Validation Certificate #
RSA (key wrapping, key establishment methodology provides between 80 and 150 bits of encryption strength).	PKCS#1 version 1.5	206
DSA	FIPS 186-2	201
Triple DES - CBC, CFB8, CFB64, ECB, OFB modes	FIPS 46-3	505
AES - CBC, CFB8, CFB128, ECB, OFB each with 128, 192, or 256 bit keys	FIPS 197	492
HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	FIPS 198	247
SHA-1, SHA-224 , SHA-256, SHA-384, SHA-512	FIPS 180-2	563
Diffie-Hellman (key agreement, key establishment provides between 80 and 256 bits of encryption strength)	PKCS#3	N/A
RNG	ANSI X9.31	273

Table1: Approved Cryptographic Algorithms

2.3 Non-Approved Cryptographic Algorithms

When running in FIPS mode, all non-FIPS algorithms (see *Table 2 Non-FIPS-Approved Algorithms*) in the Cryptographic Module are disabled. RSA and Diffie-Hellman methodology is used for key agreement/key wrapping (these are non-approved but allowed). Diffie-Hellman provides between 80 bits and 256 bits of encryption strength,

while RSA provides between 80 and 150 bits of encryption strength. We support key sizes from 1024 to 4096 bits for RSA, and key sizes from 1024 to 15360 for Diffie-Hellman.

Table 2 Non-FIPS-Approved Algorithms

Cipher Type	Algorithm
Symmetric	DES, Blowfish, Cast, RC2, RC4, RC5
Public Key	ECC (Eliptic Curve Cryptography)
Authentication Codes and Hash Functions	MD2, MD4, MD5, MDC2, RIPEMD
Random Number Generation	Message-digest based PRNG

2.4 Approved Mode of Operation

The Cryptographic Module is built as a dynamic link library under Microsoft Windows. It loads into memory at a fixed location. Address resolution is performed by the loader at application startup – the routines are not linked dynamically.

When the Module is loaded into memory, it is by default not initialized and is in non-FIPS mode, with an internal global flag *FIPS_mode* set to false. In this un-initialized state all cryptographic algorithms are enabled.

To initialize the Module and to enter FIPS mode, a single call to *FIPS_mode_set()* needs to be made by the application program that loaded the Cryptographic Module. This call will calculate an HMAC-SHA-1 digest of the FIPS object code and compare it to a value that was calculated and stored into the DLL when the Cryptographic Module was linked. If this integrity test fails, then the Module fails to initialize in FIPS mode.

If this integrity test succeeds, then the power-up self test is performed. If any component of the power-up self test fails, then the Cryptographic Module fails to initialize in FIPS mode (a global error flag is set to *FIPS_selftest_fail* and the *FIPS_mode* flag is set to FALSE, and then the Module transitions to the error state). If all self-tests are successful, then the *FIPS_mode* flag is set to TRUE.

If the *FIPS_mode* flag is TRUE, then *FIPS_mode_set()* returns 1; otherwise, it returns 0.

At this point, when the *FIPS_mode* flag is TRUE, the Cryptographic Module is now in FIPS mode and only FIPS-approved cryptographic routines are available.

The Cryptographic Module performs ANSI X9.31 compliant pseudo-random number generation.

There are two modes of operation for the Cryptographic Module: FIPS mode and non-FIPS mode (which is the default at startup time). Although there are a number of other non-FIPS cryptographic functions implemented in the Cryptographic Module (shown in *Table 2 Non-FIPS-Approved Algorithms*), when FIPS mode is started these functions are actively disabled. Note that DES is not supported in FIPS mode, although it is available in non-FIPS mode.

3. Ports and Interfaces

The physical ports of the Cryptographic Module are those of the computer upon which it executes.

The logical interface of the Cryptographic Module is an Application Programming Interface (API). This logical interface exposes services that applications may utilize directly or extend to add support for new data sources or protocols. The API provides functions that may be called by the referencing application.

The API interface provided by the Cryptographic Module is mapped onto the FIPS 140- 2 logical interfaces: data input, data output, control input, and status output.

Table 3 Relationship of Cryptographic Module Interfaces to FIPS 140-2 Logical Interfaces

FIPS 140-2 Interface	Logical Interface	Physical Interface
Data Input interface	Input parameters to all functions that accept input from Crypto-Officer or User entities	Ethernet/Network Port, USB Port, Parallel Port
Data Output interface	Output parameters from all functions that return values from Crypto-Officer or User entities	Ethernet/Network Port, USB Port, Parallel Port
Control Input interface	All API functions that are input into the Module by the Crypto-	Keyboard and Mouse

FIPS 140-2 Interface	Logical Interface	Physical Interface
	Officer and User entities	
Status Output interface	Information returned via exceptions (return / exit codes) to Crypto-Officer or User entities	Monitor
Power Interface	Initialization Function	Power Supply

4. Roles and Services

The Cryptographic Modules meets all of the FIPS 140-2 requirements for services and roles, implementing the *User* and *Crypto Officer* roles. No authentication is performed for these roles, as allowed by FIPS 140-2. *Table 4 Summary of Services* summarizes the services provided by the Cryptographic Module. Note that except for Module Installation and Initialization (which can only be performed by the Crypto Officer) all services can be performed by either role. Both of these roles are implicitly assumed by the entity that accesses services from the Cryptographic Module.

Table 4 Summary of Services

Roles	Service	Critical Security Parameters	Algorithm	Access
User, Crypto Officer	Symmetric Encryption/ Decryption	symmetric key	AES, Triple DES	Read Write Execute
User, Crypto Officer	Digital Signatures	public/private keys	RSA, DSA	Read Write Execute
User, Crypto Officer	Message Digest	none	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	Read Write Execute

Roles	Service	Critical Security Parameters	Algorithm	Access
		HMAC key	HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	
User, Crypto Officer	Random Number Generation	seed key	ANSI X9.31	Read Write Execute
User, Crypto Officer	Show Status	none	N/A	Execute
User, Crypto Offficer	Zeroization	Keys	N/A	Write, Execute
Crypto Officer	Module Initialization	none	N/A	Read Execute
User, Crypto Officer	Self Test (includes integrity, known answer, and pair-wise consistency tests)	HMAC-Key	N/A	Execute
User, Crypto Officer	Key Establishment	Diffie-Hellman, RSA	Diffie-Hellman, RSA	Read Write Execute

5. Operational Environment

5.1 Operating System Platform

The Cryptographic Module has been tested on 32 bit Microsoft Windows XP Professional with SP 2 installed. The software Module maintains compliance when running on other versions of 32 bit Microsoft Windows.

5.2 Hardware

Table 5 Test Machine Specifications

Component	Type
Processor	Intel Pentium 4 (2.80GHz)
RAM	2 GB
Hard Disk Size	75 GB
Chip set / motherboard	Intel D865GLC
USB ports	Intel 82801EB USB
Ethernet Controller	Intel Pro/1000 CT Network Connection
DVD Drive	HT-DT-ST DVD-ROM GDR8163B
Power Supply	standard 300 watt PC power supply

6. Key and CSP Management

6.1 Critical Security Parameters (CSPs)

Table 6 Critical Security Parameters describes all of the CSP's that may reside within the Module. The Module does not permanently store keys as it is up to the application to manage these. The Rules of Operation section define how keys and CSP's must be managed to maintain FIPS approved mode of operation.

Table 6 Critical Security Parameters

Key	Key Type	Storage	Use	Role
Symmetric Keys	Triple-DES, AES	Not Stored	data encryption and decryption.	User, CO
Asymmetric Keys	RSA, DSA	Not Stored	Signature Generation and verification	User CO
Wrapping Keys	RSA	Not Stored	Key transport and key	CO

Key	Key Type	Storage	Use	Role
			establishment	User
Diffie-Hellman Keys	Diffie-Hellman	Not Stored	Key establishment	User, CO
RNG Key	Triple-DES	Not Stored	Used as part of the ANSI X9.31 key generation	User, CO

6.2 Key Generation

The Cryptographic Module uses the ANSI X9.31 Appendix 2.4 RNG for generating all keys.

6.3 Key Entry and Output

All keys that are input or output by an application from within the physical boundary must be encrypted with a FIPS Approved algorithm of the appropriate strength. Secret keys used for key establishment must be wrapped with RSA before being output by the application using the Cryptographic Module to perform cryptographic operations.

6.4 Storage of Keys and CSPs

The Cryptographic Module does not store any critical security parameters (CSPs) in persistent media; while the Cryptographic Module is initialized any CSPs reside temporarily in RAM and are destroyed at the end of the session. Any keys or other CSPs otherwise stored in persistent media must be protected using a FIPS Approved algorithm.

6.5 Destruction of CSPs

When no longer needed, CSPs contained within the application must be zeroized by overwriting in a way that will make them unrecoverable. The *fips_rand_bytes()* function in the Cryptographic Module can be used to generate random data to overwrite the storage location of a CSP.

7. Self-Tests

At startup (when loaded by the calling application) the Cryptographic Module performs a number of power-up and conditional self-tests to ensure operational integrity and correct operation. Power up tests may be performed at any time by calling *FIPS_selftest()* – if it returns TRUE then the Cryptographic Module is in FIPS mode.

No FIPS mode cryptographic capability will be available until all power-up self-tests have completed successfully.

Failure of any self-test causes the Cryptographic Module to enter the Error state. This will cause the Module to exit, and thus will cause the application process to terminate.

The integrity test consists of computing an HMAC-SHA-1 digest of the Cryptographic Module in memory and comparing it to the value calculated and stored into the Cryptographic Module when it was linked. If it is identical, then the test passes.

7.1 Power-Up Tests

The power-up self-tests for the following algorithms use a known answer test (KAT) as shown in Table 7.

Table 7 Power-Up Known Answer Tests

Algorithm	Known Answer Test
AES	encryption and decryption with 128 bit key
Triple DES	encryption and decryption
SHS	SHA-1, SHA-224, SHA-384, SHA-256, SHA-512
DSA	pairwise consistency test (signing and signature verification) – allowed by FIPS 140-2 in lieu of a KAT for DSA
RSA	pairwise consistency test with a 1024 bit key and a KAT with a 1024 bit key
HMAC	HMAC-SHA-1 HMAC-SHA-224 HMAC-SHA-256 HMAC-SHA-384 HMAC-SHA-512
RNG	random number generation KAT from known IV

7.2 Conditional Tests

In addition to the power - up tests, the Module performs several conditional tests including pair-wise consistency tests on newly generated public and private key pairs, as shown in Table 8 Conditional Tests.

Conditional tests are performed automatically as necessary and cannot be turned off. All conditional tests relate to services available only to users. Thus, conditional and critical function tests are not performed at any time in response to Crypto-Officer actions.

Table 8 Conditional Tests

Algorithm	Conditional Test
DSA	pairwise consistency test (signing and signature verification)
RSA	pairwise consistency test (public encryption and private decryption)
RNG	Continuous RNG test to compare generated output with previously generated output

8. Design Assurance

8.1 Source Code Control

Source control for the Cryptographic Module is maintained in a version control repository, which allows for version control, change control, and problem tracking. This version control system assigns a unique numerical ID that tracks every change in the source code repository. The Cryptographic Module is designed using a high level programming language (C++).

Documentation is maintained in Microsoft Word format, with *change tracking* enabled. Every significant change results in a new second level version number, displayed on the title page and every page header. The supporting documentation consists of the Security Policy, and the Vendor Evidence document, the diagram of the Finite State Machine.

Both source code and documentation are maintained in a Subversion repository, which is an Open Source revision control system (http://subversion.tigris.org/).

8.2 Setup and Initialization

The Cryptographic Module is installed under the Windows operating system using the Windows Installer.

After installation, no further initialization of the Cryptographic Module is required, other than the initialization performed by the operating system at Module load time (relocation of addresses, etc. as may be required). The Module is started in non-FIPS mode and must enter FIPS mode via a call from the loading application program.

9. Rules of Operation

1) The Cryptographic Module is initialized in the FIPS mode of operation using the FIPS_mode_set() function call.
2) The replacement or modification of the Cryptographic Module by unauthorized intruders is prohibited.
3) The Operating System enforces authentication method(s) to prevent unauthorized access to Cryptographic Module services
4) All Critical Security Parameters are verified as correct and are securely generated, stored, and destroyed.
5) All host system components that can contain sensitive cryptographic data (main memory, system bus, disk storage) must be located in a secure environment.
6) The referencing application accessing the Cryptographic Module runs in a separate virtual address space with a separate copy of the executable code.
7) The unauthorized reading, writing, or modification of the address space of the Cryptographic Module is prohibited.
8) The writable memory areas of the Cryptographic Module (data and stack segments) are accessible only by a single application so that the Cryptographic Module is in *single-user* mode, i.e. only the one application has access to that instance of the Cryptographic Module.
9) The operating system is responsible for multitasking operations so that other processes cannot access the address space of the process containing the Cryptographic Module. Secret or private keys that are input or output from an application must be input or output in encrypted form using a FIPS Approved algorithm.

10. Mitigation of Other Attacks

No mitigation of other attacks is performed.

11. Physical Security

As the Cryptographic Module is software-only, no physical security is claimed.

12. References

1. FIPS PUB 140- 2 , *Security Requirements for Cryptographic Modules* , May 2001, National Institute of Standards and Technology

2. *Derived Test Requirements for FIPS PUB 140- 2, Security Requirements for Cryptographic Modules* , 15 November 2001 (draft), National Institute of Standards and Technology

3. *Implementation Guidance for FIPS PUB 140- 2 and the Cryptographic Module Validation Program* , January 21, 2005, National Institute of Standards and Technology

4. FIPS PUB 197 , *Advanced Encryption Standard (AES)*, 26 November 2001, National Institute of Standards and Technology

5. FIPS PUB 46- 3 , *Data Encryption Standard (DES)*, 25 October 25 1999, National Institute of Standards and Technology

6. FIPS PUB 81 , *DES Modes of Operation* , 2 December 1980, National Institute of Standards and Technology

7. The Advanced Encryption Standard Algorithm Validation Suite (AESAVS) , 15 November 2002, National Institute of Standards and Technology

8. NIST Special Publication 800- 20 , *Modes of Operation Validation System for the Triple Data Encryption Algorithm (TMOVS): Requirements and Procedures* , April 2000, National Institute of Standards and Technology

9. NIST Special Publication 800- 17 , *Modes of Operation Validation System (MOVS): Requirements and Procedures* , February 1998, National Institute of Standards and Technology

10. FIPS 180- 1 , Secure Hash Standard (SHS), 17 April 1995, National Institute of Standards and Technology

11.*Network Security with OpenSSL,* John Viega et. al., 15 June 2002, O'Reilly & Associates

12. FIPS 171, National Institute of Standards and Technology, 27 April 1992, http: / / c s rc.nist.gov/publications / fips / fips171 / fips171.txt

13.RFC 2246, *The TLS Protocol* , T. Dierks, C. Allen, January 1999, http: / /www.ietf.org/ rfc / r fc2246.txt

.

15.*Handbook of Applied Cryptography* , Alfred Menezes, October 1996, CRC Press. The relevant page describing a RNG implementation is available online at http: / /www.cacr.math.uwaterloo.ca /hac / about / chap5.pdf .

16. *X9.31- 1988, Digital Signatures using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)* , September 9, 1998, American National Standards Institute.