SSH Communications Security Corp
SSH Cryptographic Library
Software Version 1.2.0

FIPS 140-2 Non-Proprietary Security Policy

Level 1 Validation
Document Version 1.7

February 2006

© Copyright 2004 SSH Communications Security Corp
This document may be freely reproduced and distributed whole and intact including this Copyright Notice.

Table of Contents

© Copyright 2006 SSH Communications Security Corp

This document may be freely reproduced and distributed whole and intact including this Copyright Notice.

1. Introduction

1.1. Purpose

This is a non-proprietary Cryptographic Module Security Policy for the SSH Cryptographic Library Software Version 1.2.0 from SSH Communications Security Corp. This security policy describes how SSH Cryptographic Library Software Version 1.2.0 meets the security requirements of FIPS 140-2 and how to run the module in a secure FIPS 140-2 mode. This policy was prepared as part of the Level 1 FIPS 140-2 validation of the module.

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/.

The SSH Cryptographic Library Software Version 1.2.0 is referred to in this document as the Cryptographic Library or the module.

1.2. References

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

- The SSH Communications Security Corp website (http://www.ssh.com/) contains information on the full line of products from SSH Communications Security Corp.

- The NIST Validated Modules website (http://csrc.ncsl.nist.gov/cryptval/) contains contact information for answers to technical or sales-related questions for the module.

1.3. Document Organization

The Security Policy document is one document in a complete FIPS 140-2 Submission Package. In addition to this document, the complete Submission Package contains:

- Vendor Evidence document
- Other supporting documentation as additional references

This Security Policy and the other validation submission documentation were produced by Corsec Security, Inc. under contract to SSH Communications Security Corp. With the exception of this Non-Proprietary Security Policy, the FIPS 140-2 Validation Documentation is proprietary to SSH Communications Security Corp and is releasable only

© Copyright 2006 SSH Communications Security Corp

This document may be freely reproduced and distributed whole and intact including this Copyright Notice.

under appropriate non-disclosure agreements. For access to these documents, please contact SSH Communications Security Corp.

© Copyright 2006 SSH Communications Security Corp

This document may be freely reproduced and distributed whole and intact including this Copyright Notice.

2. SSH CRYPTOGRAPHIC LIBRARY SOFTWARE VERSION 1.2.0

2.1. Overview

The SSH Cryptographic Library Software Version 1.2.0 was developed by SSH Communications Security Corp, one of the world's leading developers of Internet-based data security technologies and solutions. The Cryptographic Library was designed to be used with computer security applications such as Secure Shell, and is available on its own, as a DLL or a shared library, or as part of SSH Secure Shell.

The Cryptographic Library has been designed to be efficient, simple to use, and easy to extend. It includes a C-language Crypto API and Math API that provide security applications with cryptographic functions, utilizing common cryptographic algorithms, including AES, Triple-DES, DSA, and RSA.

The Cryptographic Library is divided into the following functional components:

- Cryptographic random number generators
- Cryptographic hash functions
- Message authentication code (MAC) functions
- Symmetric key encryption/decryption algorithms
- Public key cryptography and digital signatures
- Algebraic group management
- Key agreement methods

2.2. Cryptographic Module

The SSH Cryptographic Library Software Version 1.2.0 is classified as a multi-chip standalone module for FIPS 140-2 purposes. As such, the module must be tested upon a particular operating system and computer platform. In particular, since the module is available as a DLL or shared library, it is to be used on a standard PC running a current Windows, Solaris, HP-UX, or AIX operating system in single user mode.

Logically, the cryptographic module is composed of a single software library. Physically, the cryptographic boundary of the module is the PC case, which physically encloses the complete set of hardware and software, including the Cryptographic Library and the operating system.

© Copyright 2006 SSH Communications Security Corp

This document may be freely reproduced and distributed whole and intact including this Copyright Notice.

The module is intended to meet overall FIPS 140-2 level 1 requirements (see Table 1).

Section	Section Title	Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1
3	Roles, Services, and Authentication	1
4	Finite State Model	1
5	Physical Security	1
6	Operational Environment	1
7	Cryptographic Key Management	1
8	EMI/EMC	3
9	Self-tests	4
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A

Table 1 – Intended Level Per FIPS 140-2 Section

2.3. Module Interface

As a multi-chip standalone module being tested on a standard PC, the module's physical interfaces consist of the keyboard port, mouse port, monitor port, CD-ROM drive, floppy disk, serial ports, USB ports, parallel ports, network ports, and power plug. All of these physical ports are mapped into the logical, software interfaces of the module and then into logical, FIPS140-2 interfaces, as described in the following table:

FIPS 140-2 Logical Interface	Logical Interface	Standard PC Physical Port
Data Input Interface	Function calls that accept, as their arguments, data to be used or processed by the module	Keyboard, mouse, CD-ROM drive, floppy disk, and serial/USB/parallel/network ports
Data Output Interface	Arguments for a function that specify where the result of a function is stored	Floppy disk, monitor, and serial/USB/parallel/network ports
Control Input Interface	Function calls utilized to initiate the module and the function calls used to control the operation of the module	Keyboard, mouse, CD-ROM drive, floppy disk, and serial/USB/parallel/network ports
Status Output Interface	Return values	Floppy disk, monitor, and serial/USB/parallel/network ports
Power Interface	Not Applicable	Power plug

© Copyright 2006 SSH Communications Security Corp

This document may be freely reproduced and distributed whole and intact including this Copyright Notice.

Table 2 – FIPS 140-2 Logical Interfaces

The logical interfaces are separated through the API used to access the module. Data Inputs are certain function calls, including specific arguments to a function. These specific arguments are pointers to input data, such as data to be signed or encrypted. Control Inputs are also certain function calls, including specific arguments. However, these arguments control how the function is executed and specify, for example, which key or algorithm to use. The set of Data Input functions and the set of Control Input functions are mutually exclusive. The result of a function is Data Output. Some functions include an argument that specifies where the result (Data Output) should be stored. The return values of the Cryptographic Library functions are Status Outputs.

2.4. Roles and Services

The Cryptographic Library meets all FIPS 140-2 level 1 requirements for Roles and Services, implementing both a *User* role and a *Crypto Officer* role. There is no Maintenance role.

As allowed by FIPS 140-2, the Cryptographic Library does not support authentication for these roles. Roles are implicitly selected through the use of functions, and only one role may be active at a time.

The Crypto Officer is responsible for initializing the module and switching the mode of operation to FIPS mode after initialization. After the successful switch to FIPS mode, both the Crypto Officer and the User will be able to utilize the cryptographic functionality of the module. In fact, they will have the same cryptographic functions available. These functions are grouped into the different types of services provided by the module (see Appendix A). The services available to the Crypto Officer and User role are provided in the table below:

Role	Service	CSPs Accessed
Crypto Officer	Initialize Cryptographic Library	-
Crypto Officer	Certification mode management	-
Crypto Officer/User	Perform self-tests	Integrity Test Key
Crypto Officer/User	Return state of module	-
Crypto Officer/User	Free strings allocated by module	-
Crypto Officer/User	Convert returned error to user-printable strings	-
Crypto Officer/User	Get Cryptographic Library version	-

© Copyright 2006 SSH Communications Security Corp

This document may be freely reproduced and distributed whole and intact including this Copyright Notice.

Crypto Officer/User	Get progress information	-
Crypto Officer/User	Generate random numbers using the chosen PRNG	Seed
Crypto Officer/User	Generate random numbers using the internal RNG	Seed
Crypto Officer/User	Compute Hash	-
Crypto Officer/User	Compute MAC	HMAC SHA-1 Symmetric Key
Crypto Officer/User	Encrypt/decrypt using symmetric algorithms	DES, Triple-DES, or AES Symmetric Key
Crypto Officer/User	Create and manipulate public and private keys	RSA or DSA Private Key; Seed (if keys are generated); DH Secret
Crypto Officer/User	Import and export private keys, public keys, groups, and randomizers	RSA or DSA Private Key; DH Secret
Crypto Officer/User	Sign and verify digital signatures	RSA or DSA Private Key
Crypto Officer/User	Generate and manipulate algebraic groups	Seed (if groups are generated)
Crypto Officer/User	Precompute randomizers for groups	Seed
Crypto Officer/User	Precompute data for private key, public key or group	-
Crypto Officer/User	Establish shared secret using Diffie-Hellman	Seed (if Secret is generated); DH Shared; DH Shared Secret
Crypto Officer/User	Zeroize Cryptographic Library	All
Crypto Officer/User	Uninitialize Cryptographic Library	-

Table 3 – Roles and Services

2.5. Physical Security

The Cryptographic Library is a software module tested for use on a standard PC running a current Windows, Solaris, HP-UX, or AIX operating system in single user mode. The module was tested against FIPS 140-2

© Copyright 2006 SSH Communications Security Corp

This document may be freely reproduced and distributed whole and intact including this Copyright Notice.

level 1 requirements on a standard PC running Windows XP, Solaris 8, HP-UX 11i, and AIX 4.3.3. The platform provides production grade equipment, industry-standard passivation, and a production grade enclosure.

In addition to the tested platforms, Cryptographic Library supports Windows 2000, Windows 2003 Server, Solaris 2.6, Solaris 9, Solaris 10, AIX 5.3, HP-UX 10.20, HP-UX 11.00, HP-UX 11.22, HP-UX 11.23 and RedHat Enterprise Linux versions 2.1 and 3.0.

Although the Cryptographic Library consists entirely of software, the FIPS 140-2 tested platform is a standard PC, which has been tested for and meets applicable FCC EMI and EMC requirements for business use as defined by 47 Code of Federal Regulations, Part15, Subpart B.

2.6. Cryptographic Key Management

The module implements the following FIPS-approved algorithms:

Type	Algorithm	Standard
Asymmetric	RSA Digital Signature (Vendor Affirmed)	PKCS #1
	DSA (Cert#82)	FIPS 186-2
Symmetric	AES (ECB, CBC, CFB, OFB, CTR) (Cert#52)	FIPS 197
	3-Key Triple-DES (TECB, TCBC, TCFB64, TOFB) (Cert#162)	FIPS 46-3
	DES (ECB, CBC, CFB, OFB – for legacy use only) (Cert#207)	FIPS 46-3
Hash	SHA-1 (Cert#145)	FIPS 180-1
MAC	HMAC SHA-1 (Cert#145, Vendor Affirmed)	FIPS 198
PRNG	Annex A.2.4 of ANSI X9.31 (Triple- DES)	ANSI X9.31
	Annex A.4 of ANSI X9.62 (SHA-1)	ANSI X9.62
	Appendix 3.1 of FIPS 186-2 (SHA-1)	FIPS 186-2
	Appendix 3.2 of FIPS 186-2 (SHA-1)	FIPS 186-2

Table 4 – FIPS-Approved Algorithms supported by the Module

The module implements the following non-FIPS-approved algorithms:

- MD5
- SHA-256
- HMAC MD5
- HMAC SHA-1 96
- CAST-128
- Blowfish

© Copyright 2006 SSH Communications Security Corp

This document may be freely reproduced and distributed whole and intact including this Copyright Notice.

- Twofish
- Arcfour
- RSA (PKCS#1 v1 and v1.5, encryption/decryption)
- RSA (PKCS#1, MD5 signatures)
- Diffie-Hellman

The following table summarizes the module's CSPs:

Type of CSP	Generation	Main Usage
DES, Triple-DES, AES, and HMAC SHA-1 Symmetric Keys	External – entered through type-specific allocation functions	MAC function (HMAC SHA-1) and symmetric encryption/ decryption (AES, DES, Triple-DES)
RSA Private Key (D)	Internal – using ssh_private_key_generate or ssh_private_key_define or External – entered through ssh_pk_import	RSA: signature generation
DSA Secret Key (X)	Internal – using ssh_private_key_generate or ssh_private_key_define or External – entered through ssh_pk_import	DSA: signature generation and required to compute Public Key Y
DH Secret	Internal – ssh_pk_group_dh_setup_async or ssh_pk_group_generate_randomizer	Diffie-Hellman: required to derive public value and shared secret
DH Shared Secret	Internal – ssh_pk_group_dh_agree_async	Create symmetric keys (outside the module)
24-byte Triple-DES Key	External – supplied noise through ssh_random_add_noise, with the output of the function hashed using SHA-256	Seed key for ANSI X9.31 PRNG
64-byte Seed	External – supplied noise through ssh_random_add_light_noise or ssh_random_add_entropy	ANSI X9.62 and FIPS 186-2 PRNGs
Integrity Test Key	External – hard-coded HMAC key generated by SSH	Software integrity test using HMAC

© Copyright 2006 SSH Communications Security Corp

This document may be freely reproduced and distributed whole and intact including this Copyright Notice.

		SHA-1

Table 5 – Summary of the Module's CSPs

The User and Crypto Officer have read and write access to all non-persistently stored CSPs, and only read access to the hard-coded integrity test key.

All CSPs remain in the process space of a single User, with the operating system protecting memory and process space from unauthorized access.

2.6.1. Key Generation

The only keys that can be generated by the module are public/private keys for RSA and DSA signature schemes, and the secret and public values for the Diffie-Hellman protocol. FIPS-approved PRNGs are used to generate these keys.

2.6.2. Key Establishment

The module uses Diffie-Hellman to agree upon shared secrets (not keys). These shared secrets are exported out of the module without further processing. (Applications can use these exported shared secrets to externally create symmetric keys.)

2.6.3. Key Entry and Output

CSPs are entered into and output from the module electronically. Symmetric keys are entered into the module through the allocation functions used to allocate and initialize a MAC or (symmetric) Cipher context. Symmetric keys are not output from the module. When the context is no longer needed, a free function is called to free the context and to zeroize any key-sensitive material in the context. Private keys, public keys, and Diffie-Hellman's secret and public value can be entered into and output from the module using the uniform import and export interface. All private keys exported from the module must be encrypted using a FIPS-approved algorithm. Imported private keys may be encrypted.

2.6.4. Key Storage

The module employs hard-coded keys for the software integrity test and the known answer tests. All other CSPs are stored as plaintext in RAM and are generated or entered into the module while its API is being exercised by the User or Crypto Officer.

2.6.5. Key Zeroization

Non-persistently stored keys can be zeroized using either the master zeroize function or type-specific zeroize functions. Hard-coded keys are zeroized when the Cryptographic Library is deleted from the system.

© Copyright 2006 SSH Communications Security Corp

This document may be freely reproduced and distributed whole and intact including this Copyright Notice.

2.7. Self-Tests

The Cryptographic Library runs power-up and conditional self-tests to verify that it is functioning properly. The power-up self-tests are performed during initialization. Conditional self-tests are executed whenever specific conditions are met. The module implements the following self-tests:

Software Integrity Test: The module employs a software integrity test in the form of an HMAC SHA-1.

Cryptographic Algorithm KATs: Known Answer Tests (KATs) are run at power-up for the following algorithms:

- AES KAT (ECB)
- Triple-DES KAT (ECB, CBC, OFB, and CFB)
- DES KAT (ECB, CBC, OFB, and CFB)
- SHA-1 KAT
- HMAC SHA-1 KAT
- PRNG KAT

Pair-Wise Consistency Checks: Pair-wise consistency checks are performed at power-up and after RSA, DSA and Diffie-Hellman keys have been generated. The following pair-wise consistency checks are implemented:

- RSA pair-wise consistency check
- DSA pair-wise consistency check
- Diffie-Hellman pair-wise consistency check

Critical Function Test: At power-up the module runs the math library self-tests.

Statistical Random Number generator Tests:

- The monobit test
- The poker test
- The runs test
- The long runs test

© Copyright 2006 SSH Communications Security Corp

This document may be freely reproduced and distributed whole and intact including this Copyright Notice.

Continuous Random Number Generator Test: This test is performed to detect failure of the module's random number generator.

If any of these self-tests fail, the module will enter the error state. The error condition can be cleared by releasing all crypto object instances and successfully executing the uninitialize function.

2.8. Design Assurance

SSH Communications Security Corp manages and records source code and associated documentation files using the Concurrent Versions System (CVS).

2.9. Mitigation of Other Attacks

The Cryptographic Library does not employ security mechanisms to mitigate specific attacks.

© Copyright 2006 SSH Communications Security Corp

This document may be freely reproduced and distributed whole and intact including this Copyright Notice.

3. SECURE OPERATION

The Cryptographic Library meets Level 1 requirements for FIPS 140-2. The sections below describe how to place and keep the module in FIPS-approved mode of operation.

The module has two modes of operation: non-FIPS mode and FIPS mode. Non-FIPS mode is the default mode after initialization. The module can enter the FIPS mode by calling a specific function that switches to FIPS mode. This transition is allowed only if the module is in the OK state and no allocated crypto object instances exist, as depicted in Figure 1.

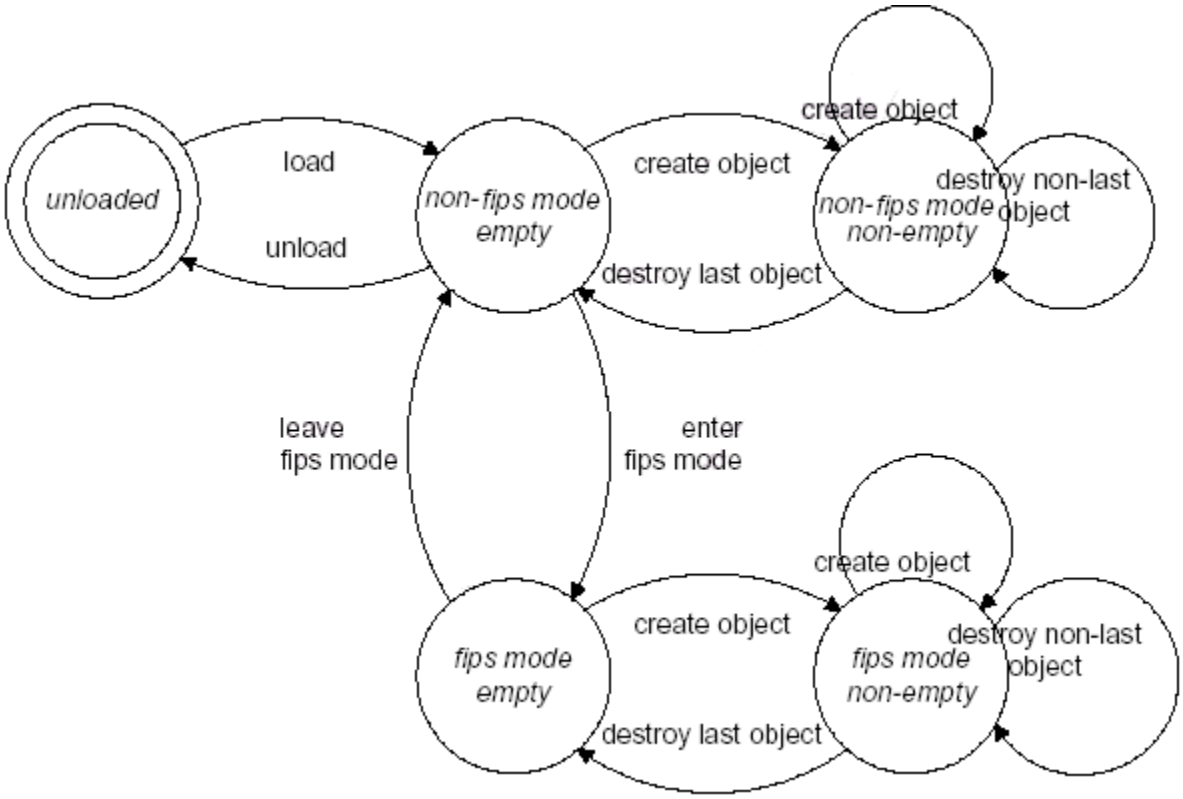

Figure 1 – Transition between Different Modes of Operation

When the module is in FIPS mode, the following functional changes will occur:

- Only FIPS-approved algorithms are available
- Private keys can be exported only in encrypted form

© Copyright 2006 SSH Communications Security Corp

This document may be freely reproduced and distributed whole and intact including this Copyright Notice.

3.1. Crypto Officer Guidance

Before any cryptographic processing can be performed, the Crypto Officer must initialize the module by calling the ssh_crypto_library_initialize function. During initialization, the module enters the self-test state and performs the power-up self-tests. If the self-tests pass, the module enters the OK state. At this point, the ssh_crypto_library_initialize function has been successfully executed and the SSH_CRYPTO_OK return value is returned by the module to notify the Crypto Officer. Since non-FIPS mode is the default mode after initialization, the Crypto Officer must switch the current mode to FIPS mode by calling the ssh_crypto_set_certification_mode and specifying that FIPS mode is the mode to switch to. If the switch was successful, the module will notify the Crypto Officer by returning the SSH_CRYPTO_OK return value.

The User guidance described below also applies to the Crypto Officer.

3.2. User Guidance

The User must verify that the current mode is FIPS mode by calling the ssh_crypto_get_certification_mode. If the function returns “FIPS mode”, the User can proceed.

When the module is in FIPS mode only FIPS-approved algorithms are allowed to be used. To help Users identify these algorithms, the User can call type-specific functions to list FIPS-approved algorithms implemented by the module. The User can also use type-specific functions to query whether a specific algorithm is FIPS-approved.

The User can explicitly call the ssh_crypto_library_self_tests function to verify that the module is functioning correctly. If all self-tests pass, the module notifies the User by returning the return value SSH_CRYPTO_OK.

To prevent the disclosure of CSPs, the User should zeroize the CSPs when they are no longer required. CSPs can be zeroized by calling a type-specific free function. Alternatively, the ssh_crypto_library_zeroize function can be called to zeroize all existing crypto objects and CSPs.

© Copyright 2006 SSH Communications Security Corp

This document may be freely reproduced and distributed whole and intact including this Copyright Notice.

4. APPENDIX A – FUNCTIONS

This appendix contains a list of the functions provided by the module. The function set is divided into the different services provided by the module, as shown in Table 6.

Service	Functions
Initialize Cryptographic Library	ssh_crypto_library_initialize
Certification mode management	ssh_crypto_get_certification_mode
	ssh_crypto_set_certification_mode
Perform self-tests	ssh_crypto_library_self_tests
Return state of module	ssh_crypto_library_get_status
Free strings allocated by module	ssh_crypto_free
Convert returned error to user-printable strings	ssh_crypto_status_message
Get Cryptographic Library version	ssh_crypto_library_get_version
Get progress information	ssh_crypto_library_register_progress_func
Generate random numbers using the chosen PRNG	ssh_random_add_entropy
	ssh_random_add_light_noise
	ssh_random_allocate
	ssh_random_enable_fips_continuous_checks
	ssh_random_free
	ssh_random_get_bytes
	ssh_random_supported
	ssh_random_get_supported
	ssh_random_is_fips_approved
	ssh_random_name
Generate random numbers using the internal RNG	ssh_random_add_noise
	ssh_random_get_byte
	ssh_random_stir
Compute Hash	ssh_hash_allocate
	ssh_hash_asn1_oid
	ssh_hash_digest_length
	ssh_hash_duplicate
	ssh_hash_final
	ssh_hash_free
	ssh_hash_get_supported
	ssh_hash_input_block_size

© Copyright 2006 SSH Communications Security Corp

This document may be freely reproduced and distributed whole and intact including this Copyright Notice.

	ssh_hash_is_fips_approved
	ssh_hash_name
	ssh_hash_reset
	ssh_hash_supported
	ssh_hash_update
Compute MAC	ssh_mac_allocate
	ssh_mac_final
	ssh_mac_free
	ssh_mac_get_block_length
	ssh_mac_get_max_key_length
	ssh_mac_get_min_key_length
	ssh_mac_get_supported
	ssh_mac_is_fips_approved
	ssh_mac_length
	ssh_mac_name
	ssh_mac_reset
	ssh_mac_supported
	ssh_mac_update
Encrypt/decrypt using symmetric algorithms	ssh_cipher_ get_supported
	ssh_cipher_supported
	ssh_cipher_is_fips_approved
	ssh_cipher_name
	ssh_cipher_allocate
	ssh_cipher_get_block_length
	ssh_cipher_get_iv
	ssh_cipher_set_iv
	ssh_cipher_get_iv_length
	ssh_cipher_get_key_length
	ssh_cipher_get_max_key_length
	ssh_cipher_get_min_key_length
	ssh_cipher_get_has_fixed_key_length
	ssh_cipher_transform
	ssh_cipher_transform_with_iv
	ssh_cipher_free
Create and manipulate public and private keys	ssh_private_key_copy
	ssh_private_key_define
	ssh_private_key_generate
	ssh_private_key_is_fips_approved
	ssh_private_key_derive_public_key
	ssh_private_key_free
	ssh_private_key_get_info
	ssh_private_key_name
	ssh_private_key_select_scheme
	ssh_public_key_name
	ssh_public_key_is_fips_approved

© Copyright 2006 SSH Communications Security Corp

This document may be freely reproduced and distributed whole and intact including this Copyright Notice.

	ssh_public_key_copy
	ssh_public_key_define
	ssh_public_key_free
	ssh_public_key_get_info
	ssh_public_key_get_supported
	ssh_public_key_select_scheme
Import and export private keys, public keys, groups, and randomizers	ssh_pk_import
	ssh_pk_export
Sign and verify digital signatures	ssh_private_key_derive_signature_hash
	ssh_private_key_max_signature_input_len
	ssh_private_key_max_signature_output_len
	ssh_public_key_derive_signature_hash
	ssh_private_key_sign_async
	ssh_private_key_sign_digest_async
	ssh_public_key_verify_async
	ssh_public_key_verify_digest_async
Generate and manipulate algebraic groups	ssh_pk_group_copy
	ssh_pk_group_free
	ssh_pk_group_generate
	ssh_pk_group_get_info
	ssh_pk_group_select_scheme
	ssh_public_key_get_predefined_groups
Precompute randomizers for groups	ssh_pk_group_generate_randomizer
	ssh_pk_group_count_randomizers
Precompute data for private key, public key or group	ssh_private_key_precompute
	ssh_public_key_precompute
	ssh_pk_group_precompute
Establish shared secret using Diffie-Hellman	ssh_pk_group_dh_setup_max_output_length
	ssh_pk_group_dh_agree_max_output_length
	ssh_pk_group_dh_setup_async
	ssh_pk_group_dh_secret_free
	ssh_pk_group_dh_agree_async
Zeroize Cryptographic Library	ssh_crypto_library_zeroize
Uninitialize Cryptographic Library	ssh_crypto_library_uninitialize

Table 6 – Functions provided by the Module

© Copyright 2006 SSH Communications Security Corp

This document may be freely reproduced and distributed whole and intact including this Copyright Notice.

Table 6 includes all functions that provide services available in a FIPS mode of operation. In addition to the functions listed in Table 6, the module has the functions listed in Table 7. When in FIPS mode, the functions in Table 7 no longer provide useful services, but rather, always output the return values detailed below.

Functions	Return Value
ssh_private_key_get_info	This function allows you to query public parameters, but not secret parameters (P, Q, D and U for RSA, X for DSA). For the latter case, the function will fail and return SSH_CRYPTO_UNSUPPORTED
ssh_private_key_create_proxy	SSH_CRYPTO_UNSUPPORTED
ssh_public_key_create_proxy	SSH_CRYPTO_UNSUPPORTED
ssh_dh_group_create_proxy	SSH_CRYPTO_UNSUPPORTED
ssh_proxy_key_get_key_handle	SSH_CRYPTO_UNSUPPORTED
ssh_proxy_key_rgf_decrypt	SSH_CRYPTO_UNSUPPORTED
ssh_proxy_key_rgf_encrypt	SSH_CRYPTO_UNSUPPORTED
ssh_proxy_key_rgf_sign	SSH_CRYPTO_UNSUPPORTED
ssh_proxy_key_rgf_verify	SSH_CRYPTO_UNSUPPORTED
ssh_public_key_encrypt_async	SSH_CRYPTO_UNSUPPORTED
ssh_public_key_max_encrypt_input_len	Since asymmetric encryption/decryption is not supported, the function will return zero
ssh_public_key_max_encrypt_output_len	Since asymmetric encryption/decryption is not supported, the function will return zero
ssh_private_key_decrypt_async	SSH_CRYPTO_UNSUPPORTED
ssh_private_key_max_decrypt_input_len	Since asymmetric encryption/decryption is not supported, the function will return zero
ssh_private_key_max_decrypt_output_len	Since asymmetric encryption/decryption is not supported, the function will return zero
ssh_pk_provider_register	SSH_CRYPTO_UNSUPPORTED

Table 7 – Functions that should not be used

© Copyright 2006 SSH Communications Security Corp

This document may be freely reproduced and distributed whole and intact including this Copyright Notice.

5. APPENDIX B – ACRONYM LIST

AES	Advanced Encryption Standard
ANSI	American National Standards Institute
CBC	Cipher-Block Chaining
CFB	Cipher Feedback
CSE	Communications Security Establishment
CSP	Critical Security Parameter
CTR	Counter
DES	Data Encryption Standard
DSA	Digital Signature Algorithm
ECB	Electronic Codebook
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FCC	Federal Communication Commission
FIPS	Federal Information Processing Standard
IV	Initialization Vector
KAT	Known Answer Test
MAC	Message Authentication Code
NIST	National Institute of Standards and Technology
OFB	Output Feedback
PKCS	Public Key Cryptography Standard
PRNG	Pseudo Random Number Generator
RAM	Random Access Memory
RNG	Random Number Generator
RSA	Rivest Shamir and Adleman
SHA	Secure Hash Algorithm

© Copyright 2006 SSH Communications Security Corp

This document may be freely reproduced and distributed whole and intact including this Copyright Notice.