Tumbleweed Communications Corp.

Tumbleweed Security Kernel

(Software Version: 2.0)

FIPS 140-2
Non-Proprietary Security Policy

Level 1 Validation

Document Version 1.0

Prepared for:

Tumbleweed®

The Experts in Secure Internet Communication

Tumbleweed Communications Corp.
700 Saginaw Drive
Redwood City, CA 94063
Phone: (650)216-2000
Fax: (650)216-2001
http://www.tumbleweed.com

Prepared by:

Corsec℠

Corsec Security, Inc.
10340 Democracy Lane, Suite 201
Fairfax, VA 22030
Phone: (703) 267-6050
Fax: (703) 267-6810
http://www.corsec.com

Revision History

Version	Modification Date	Modified By	Description of Changes
1.0	2007-06-20	Xiaoyu Ruan	Release version

Table of Contents

Table of Figures

Table of Tables

1 Introduction

1.1 Purpose

This document is a non-proprietary Cryptographic Module Security Policy for the Tumbleweed Security Kernel from Tumbleweed Communications Corp. This Security Policy describes how the Tumbleweed Security Kernel 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. and Canadian Government requirements for cryptographic modules. More information about the FIPS 140-2 standard and validation program is available on the National Institute of Standards and Technology (NIST) Cryptographic Module Validation Program (CMVP) website at: http://csrc.nist.gov/cryptval/.

In this document, the Tumbleweed Security Kernel is referred to as the kernel or the module. The client application represents the software program linked with the cryptographic libraries provided by the Tumbleweed Security Kernel. The Validation Authority is currently the only client application making use of the Tumbleweed Security Kernel. However, it is expected that a range of products developed by Tumbleweed Communications Corp., such as SecureTransport and MailGate, will be supported by the Tumbleweed Security Kernel in the future.

1.2 References

This document deals only with the 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 Tumbleweed website (http://www.tumbleweed.com) contains information on the full line of products from Tumbleweed.
- The CMVP website (http://csrc.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 FIPS 140-2 submission package. In addition to this document, the Submission Package contains:

- Vendor Evidence document
- Finite State Machine
- Other supporting documentation as additional references

This Security Policy and the other validation submission documentation was produced by Corsec Security, Inc. under contract to Tumbleweed. With the exception of this Non-Proprietary Security Policy, the FIPS 140-2 Validation Documentation is proprietary to Tumbleweed and is releasable only under appropriate non-disclosure agreements. For access to these documents, please contact Tumbleweed.

2 Tumbleweed Security Kernel

2.1 Overview

The Tumbleweed Security Kernel (version 2.0) is a software cryptographic module implemented as two dynamic link libraries (DLLs) on Windows or two Shared Objects (SO's) on Linux, SunOS, and IBM AIX. The Tumbleweed Security Kernel is a user space shared library. It does not modify or become part of the Operating System (OS) kernel. Table 1 gives the operating systems and corresponding file names of the kernel.

Table 1 – Binary Forms of the Kernel

Operating Systems	Binary File Names
Windows 2000, Windows Server 2003, and Windows XP	libeay32-TMWD.dll ssleay32-TMWD.dll
Linux kernel 2.6.5-7.244-default and later versions	libcrypto-TMWD.so.0.9.8 libssl-TMWD.so.0.9.8
SunOS 5.10 and later versions	libcrypto-TMWD.so.0.9.8 libssl-TMWD.so.0.9.8
IBM AIX 5.2.0.0 and later versions	libcrypto-TMWD.so.0.9.8 libssl-TMWD.so.0.9.8

The kernel is built upon a custom version of OpenSSL 0.9.8d. As a cryptographic module, the Tumbleweed Security Kernel presents an identical application programming interface (API) to several products of the Tumbleweed Communications Corp., including Validation Authority, SecureTransport, and MailGate.

The security features of Validation Authority, SecureTransport, and MailGate are provided by the Tumbleweed Security Kernel. Validation Authority offers a comprehensive, scalable, and reliable framework for real-time validation of digital certifications for the Public Key Infrastructure (PKI). Governments and businesses worldwide rely on PKI and digital certificates issued by certificate authorities (CAs) to secure information transmissions on the internet. Not all certificates are valid. Some may be fake, expired, or revoked. Therefore, it is of vital importance to make sure that only valid certificates are trusted. Validation Authority provides a variety of PKI and certificate management functionality such as real-time validation of digital certificates issued by any CA. SecureTransport is a secure managed file transfer solution for moving financial transactions, critical business files, large documents, Extensible Markup Language (XML), and Interactive Electronic Data Interchange (EDI) transactions over the internet and private Internet Protocol (IP) networks. The MailGate family is capable of performing tasks such as network defense, email encryption, content filtering, and data integrity.

The Tumbleweed Security Kernel supports FIPS-approved algorithms including Advanced Encryption Standard (AES), Triple Data Encryption Standard (TDES), Secure Hash Algorithm (SHA) -256, SHA-512, keyed-Hash Message Authentication Code-SHA-1 (HMAC-SHA-1), Elliptic Curve Digital Signature Algorithm (ECDSA), FIPS 186-2 Appendix 3.1 Random Number Generator (RNG), and Rivest, Shamir, and Adleman (RSA) signature generation/verification.

The Tumbleweed Security Kernel for the purpose of this FIPS validation has been tested and validated on Microsoft Windows Server 2003 with SP 2, Windows XP with SP 2, SUSE Linux 9 Enterprise Server with SP 3, SunOS 5.10, and IBM AIX 5.2.0.0 (each configured for single user mode).

The Tumbleweed Security Kernel always operates in a FIPS-Approved mode of operation. The Tumbleweed Security Kernel is validated at FIPS 140-2 section levels shown in Table 2. Note that in Table 2, EMI and EMC stand for Electromagnetic Interference and Electromagnetic Compatibility, respectively, and N/A indicates "Not Applicable".

Table 2 – Security Level Per FIPS 140-2 Section

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	N/A
6	Operational Environment	1
7	Cryptographic Key Management	1
8	EMI/EMC	1
9	Self-Tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A

2.2 Module Interfaces

The Tumbleweed Security Kernel is a software module that meets overall level 1 of FIPS 140-2 requirements. The logical cryptographic boundary of the module consists of the Tumbleweed Security Kernel running on different OS. The module is composed of two binary files cross-compiled on the OS. Table 1 summarizes the platforms and the binary files.

Figure 1 shows the logical cryptographic boundary of the kernel. The module provides a set of cryptographic services (API calls) in areas such as Transport Layer Security (TLS), RNG, and Public Key Cryptography Standard (PKCS) #12 certificate management.

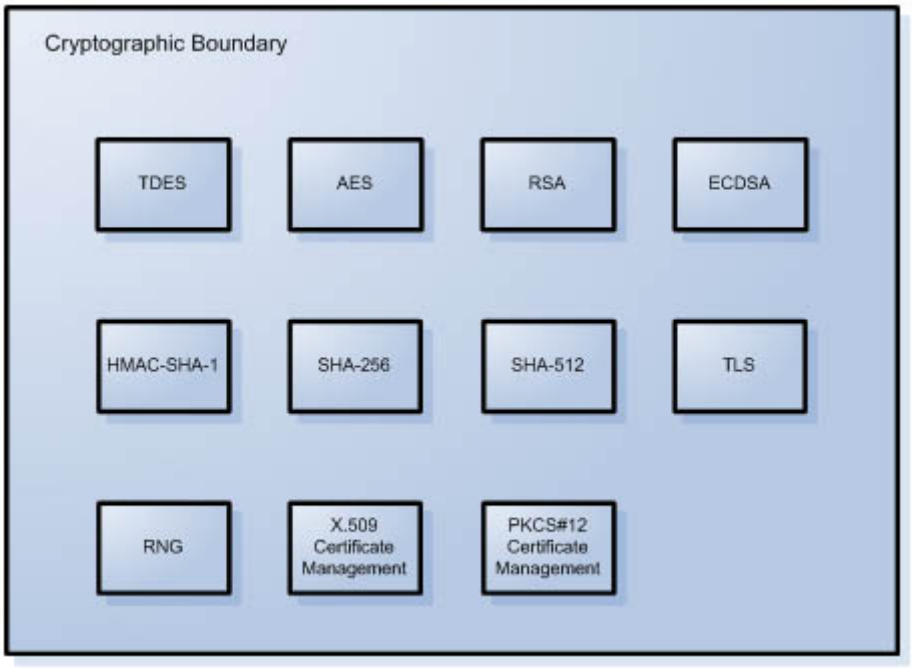

Figure 1 – Logical Cryptographic Boundary

The kernel's interactions with surrounding components, including the Central Processing Unit (CPU), hard-disk, memory, client application, and the OS are demonstrated in Figure 2.

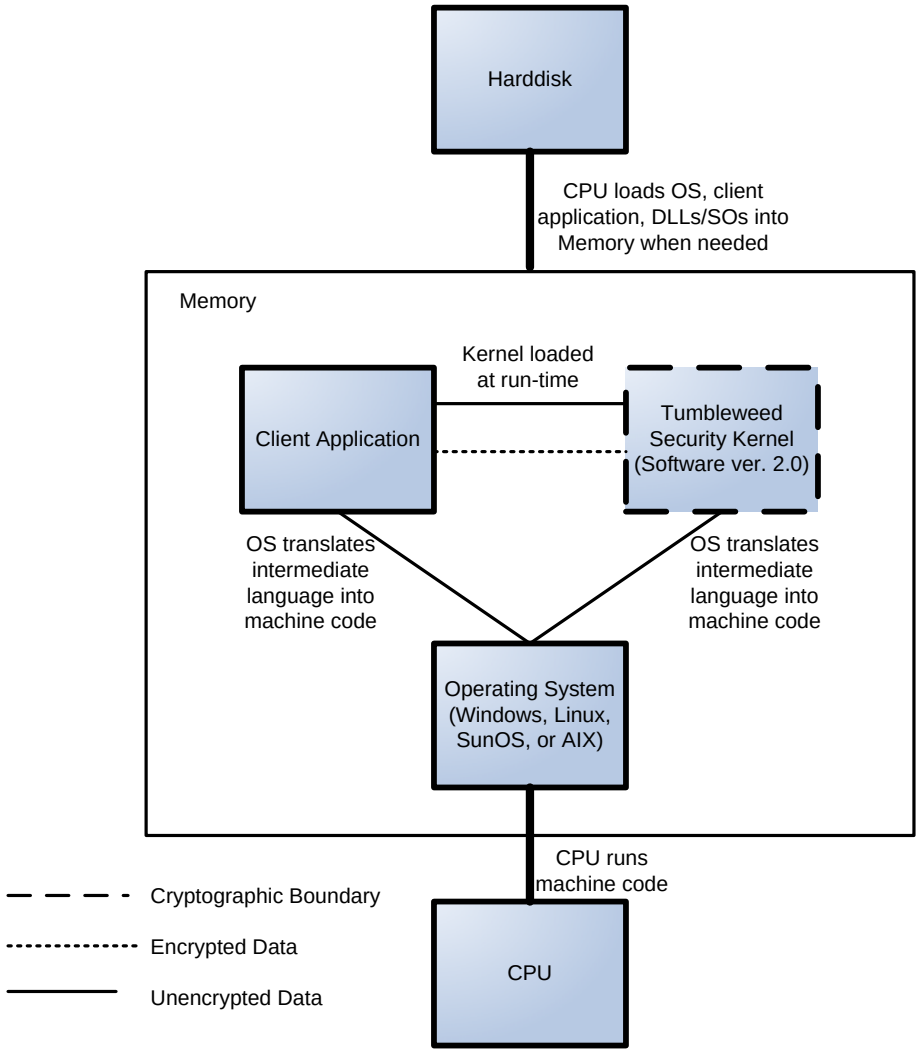

Figure 2 – Logical Cryptographic Boundary and Interactions with Surrounding Components

The module has been tested and validated on Windows Server 2003 with SP 2, Windows XP with SP 2, SUSE Linux 9 Enterprise Server with SP 3, SunOS 5.10, and IBM AIX 5.2.0.0. The module can run on a standard Personal Computer (PC) (with Windows 2000, Windows XP, or Linux), a server (with Windows Server 2003 or SunOS), or a mainframe computer (with IBM AIX OS). The platforms supported by the module are binary compatible with the platforms used in the FIPS validation.

In addition to the binaries, the physical device consists of the integrated circuits of the motherboard, the CPU, Random Access Memory (RAM), Read-Only Memory (ROM), computer case, keyboard, mouse, video interfaces, expansion cards, and other hardware components included in the PC such as hard disk, floppy disk, Compact Disc ROM (CD-ROM) drive, power supply, and fans. The physical cryptographic boundary of the module is the hard opaque metal and plastic enclosure of the PC, server, or mainframe running the module. The block diagram for a standard PC is shown in Figure 3. The physical block diagram for a server or a mainframe is similar to Figure 3. Note that in this figure, I/O means Input/Output, BIOS stands for Basic Input/Output System, PCI stands for Peripheral Component Interconnect, ISA stands for Instruction Set Architecture, and IDE represents Integrated Drive Electronics.

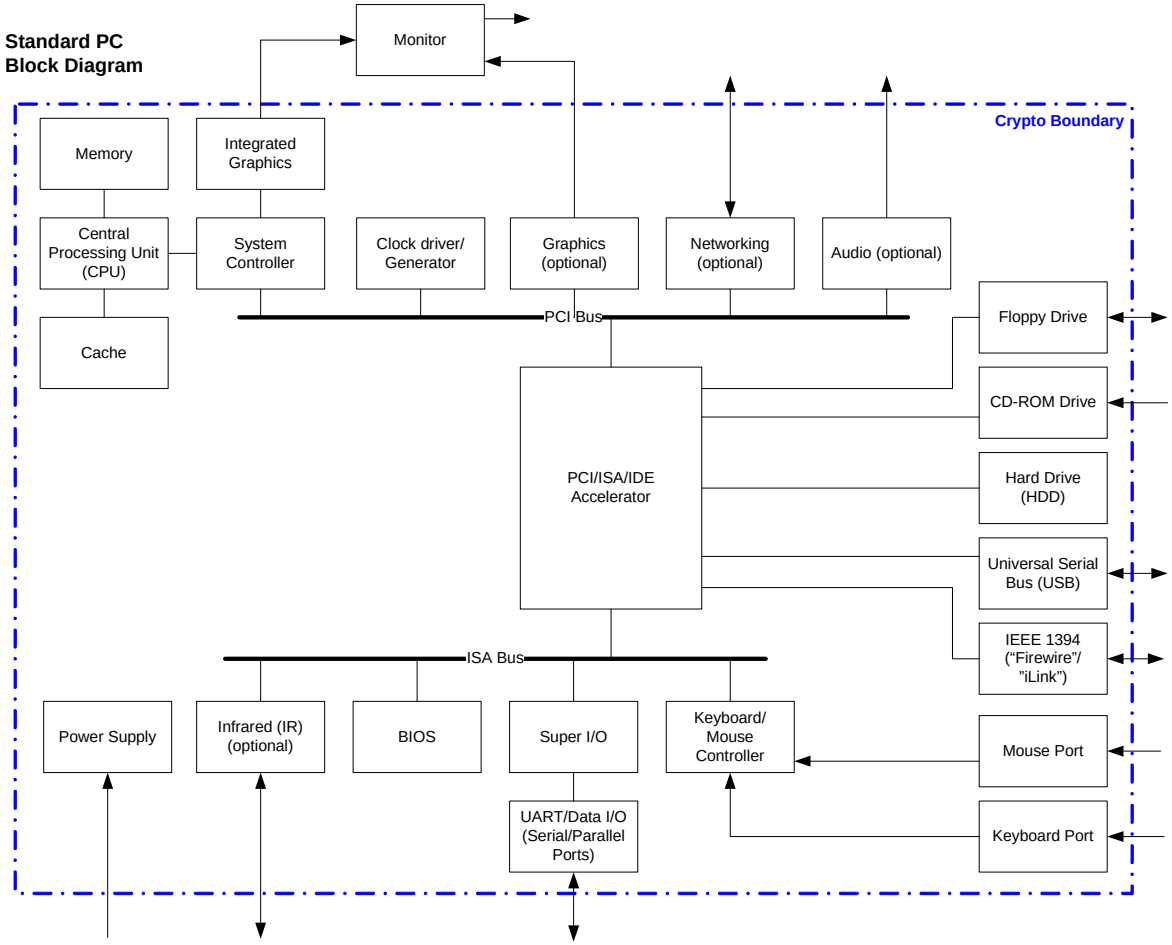

Figure 3 – Standard PC Physical Block Diagram

All of these physical interfaces are separated into logical interfaces defined by FIPS 140-2, as described in Table 3.

Table 3 – FIPS 140-2 Logical Interfaces

Logical Interface	Tumbleweed Security Kernel Port/Interface	Module Mapping
Data Input Interface	Keyboard, mouse, CD-ROM, floppy disk, and serial/Universal Serial Bus (USB)/parallel/network ports	Arguments for API calls that contain data to be used or processed by the kernel
Data Output Interface	Hard Disk, floppy disk, monitor, and serial/USB/parallel/network ports	Arguments for API calls that contain kernel response data to be used or processed by the caller
Control Input Interface	Keyboard, CD-ROM, floppy disk, mouse, and serial/USB/parallel/network port	API function calls
Status Output Interface	Hard Disk, floppy disk, monitor, and serial/USB/parallel/network ports	Arguments for API calls, function return value, error message
Power Interface	Power Interface	Not Applicable

2.3 Roles and Services

The operators of the module can assume two roles as required by FIPS 140-2: a Crypto Officer role and a User role. The operator of the module assumes either of the roles based on the operations performed without any authentication. Table 4 gives a brief description of the roles and their privileges.

Table 4 – Module Roles and Privileges

Role Name	Role Privileges
Crypto Officer	1. Installing/uninstalling the kernel on the specific platform. 2. Initiating power-up self-tests.
User	Performing cryptographic services by making API calls to the kernel.

The following subsections detail both of the roles and their responsibilities.

2.3.1 Crypto Officer Role

The Crypto Officer role has the ability to install and uninstall the module and run power-up self-tests. Descriptions of the services available to the Crypto Officer role are provided in Table 5, where CSP refers to Critical Security Parameter.

Table 5 – Crypto Officer Services

Service	Description	Input	Output	CSP and Type of Access
Install kernel	Installs and configures the kernel	Command	Success or failure	None
Uninstall kernel	Remove the kernel from the OS	Command	Success or failure	None
Initiate power-up self-tests	Power-up self-tests include: (1) software integrity test; (2) Known Answer Tests (KATs) for TDES, AES, RSA, SHA-256, SHA-512, RNG; (3) pair-wise consistency test for ECDSA keys	Command	Pass or failure	None

2.3.2 User Role

The User role accesses the module's cryptographic services that include encryption, decryption, and authentication functionality. Descriptions of the services available to the User role are provided in Table 6.

Table 6 – User Services

Service	Description	Input	Output	CSP and Type of Access
TDES encryption/decryption	Encrypt/Decrypt using TDES	Command, plaintext/ciphertext, keys	Status, ciphertext/plaintext	TDES symmetric keys - READ
TDES key generation	Generate a TDES symmetric keys using FIPS 186-2 Appendix 3.1 RNG	Command, key length	Status, symmetric keys	TDES symmetric keys - READ/WRITE
AES encryption	Encrypt plaintext using AES	Command, plaintext, key	Status, ciphertext	AES symmetric key - READ

Service	Description	Input	Output	CSP and Type of Access
AES decryption	Decrypt AES-encrypted ciphertext	Command, ciphertext, key	Status, plaintext	AES symmetric key - READ
AES key generation	Generate an AES symmetric key using FIPS 186-2 Appendix 3.1 RNG and set the key schedule	Command, key length	Status, symmetric key	AES symmetric key - READ/WRITE
RSA encryption	Encrypt plaintext using RSA	Command, plaintext, RSA public key	Status, ciphertext	RSA public key - READ
RSA decryption	Decrypt RSA-encrypted ciphertext	Command, ciphertext, RSA private key	Status, plaintext	RSA private key - READ
RSA signature generation	Sign data using RSA	Command, data to be signed, RSA private key	Status, digital signature	RSA private key - READ
RSA signature verification	Verify an RSA signature	Command, data and signature, RSA public key	Status, acceptance/denial	RSA public key - READ
RSA key-pair generation	Generate a RSA key-pair	Command, key length	Status, RSA private key and public key	RSA private key and public key - WRITE
ECDSA signature generation	Sign data using ECDSA	Command, data to be signed	Status, digital signature	ECDSA private key - READ
ECDSA signature verification	Verify an ECDSA signature	Command, data and signature	Status, acceptance/denial	ECDSA public key - READ
ECDSA key-pair generation	Generate an ECDSA key-pair	Command, key length	Status, ECDSA private key and public key	ECDSA private key and public key - WRITE
SHA-256 digest generation	Generate a SHA-256 digest	Command, message to be hashed	Status, 256-bit message digest	None
SHA-512 digest generation	Generate a SHA-512 digest	Command, message to be hashed	Status, 512-bit message digest	None
HMAC-SHA-1 key generation	Generate a HMAC-SHA-1 symmetric key	Command	Status, 512-bit HMAC-SHA-1 symmetric key	HMAC-SHA-1 key - WRITE
HMAC-SHA-1 digest generation	Generate a HMAC-SHA-1 digest	Command, message to be hashed, key	Status, 160-bit HMAC-SHA-1 message digest	HMAC-SHA-1 key - READ
Random number generation	Generate a random number	Command, seed, length	Status, random number	Seed - READ/WRITE
Establish a TLS session	Establish a new TLS session	Command	Status, session ID	None
Create X.509 certificate	Create a X.509 certificate	Command, name on the certificate, public key, algorithm used, certificate authority (CA)	Status, X.509 certificate	RSA public key - READ
Read X.509 certificate from a PEM file	Read a X.509 certificate from a PEM file	Command, PEM file for the X.509 certificate	Status, Plaintext for the X.509 certificate	RSA public key - READ/WRITE

Service	Description	Input	Output	CSP and Type of Access
Write X.509 certificate to a PEM file	Write X.509 certificate to a PEM file	Command, plaintext for the X.509 certificate	Status, PEM file for the X.509 certificate	RSA public key - READ
Free a X.509 certificate	Erase a X.509 certificate	Command, X.509 certificate	Status	RSA public key - DELETE
Create PKCS#12 certificate	Create a PKCS#12 certificate	Command, name on the certificate, public key, algorithm used, CA	Status, PKCS#12 certificate	RSA public key - READ
Read PKCS#12 certificate from a DER file	Read a PKCS#12 certificate from a DER file	Command, DER file for the PKCS#12 certificate	Status, plaintext for the PKCS#12 certificate	RSA public key - READ/WRITE
Write PKCS#12 certificate to a DER file	Write PKCS#12 certificate to a DER file	Command, plaintext for the PKCS#12 certificate	Status, DER file for the PKCS#12 certificate	RSA public key - READ
Free a PKCS#12 certificate	Erase a PKCS#12 certificate	Command, PKCS#12 certificate	Status	RSA public key - DELETE
Show status	Show status of a service (function call)	Command	Status	None

2.4 Physical Security

The Tumbleweed Security Kernel is a multi-chip standalone module. The physical security requirements do not apply to this module, since it is purely a software module and does not implement any physical security mechanisms.

Although the module consists entirely of software, the FIPS 140-2 evaluated platform is a standard PC, a server, or a mainframe, which has been tested for and meets applicable Federal Communication Commission (FCC) EMI and EMC requirements for business use as defined in Subpart B of FCC Part 15.

2.5 Operational Environment

The module runs on the general purpose Microsoft Windows, Linux, SunOS, and IBM AIX. See Column 1 of Table 1 for a list of OS that are supported by the module. The OS being used must be configured for single user mode per NIST CMVP guidance. The module was tested and validated on Windows Server 2003 with SP 2, Windows XP with SP 2, SUSE Linux 9 Enterprise Server with SP 3, SunOS 5.10, and IBM AIX 5.2.0.0. Single user mode configuration instructions for various OS can be found in Section 3.1.1 of this document.

2.6 Cryptographic Key Management

The module implements the following FIPS-approved algorithms.

- SHA-1, SHA-256, SHA-512 (certificates #597 and #608)
- HMAC-SHA-1: 160-bit MAC and 512-bit key (certificates #275 and #285)
- RSA PKCS#1 signature generation and verification: 1024 and 2048 bits (certificates #237 and #244)
- ECDSA for key generation and signature generation/verification: using the six recommended curves in Appendix 6 of FIPS 186-2 with Change Notice 1 (certificates #54, #55, and #56)
- RNG: NIST, FIPS 186-2 Appendix 3.1 (certificates #300 and #311)
- TDES: 112 bits (for 2-key) and 168 bits (for 3-key), in Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), and Output Feedback (OFB) modes (certificates #531 and #540)
- AES: 128, 192, and 256 bits, in ECB, CBC, CFB, OFB, and counter modes (certificates #524 and #543)

The module implements the following non-FIPS Approved key establishment techniques.

- Diffie-Hellman: 1024 and 2048 bits, for symmetric key establishment in TLS sessions. The 1024- and 2048-bit Diffie-Hellman provides 80 and 112 bits of security, respectively.
- RSA PKCS#1 encryption and decryption: 1024 and 2048 bits, for symmetric key transport in TLS sessions. The 1024- and 2048-bit RSA key wrapping provides 80 and 112 bits of security, respectively.

The module supports the following CSPs:

Table 7 – List of Cryptographic Keys, Cryptographic Key Components, and CSPs

Key	Key Type	Generation / Input	Output	Storage	Zeroization	Use
TDES keys	Symmetric key	1. Generated internally using FIPS 186-2 Appendix 3.1 RNG. 2. Generated using Diffie-Hellman key agreement. 3. Derived from TLS master secret. 4. Input in encrypted form.	In encrypted form.	Plaintext in volatile memory only	Zeroized after use	Encrypt plaintext/ Decrypt ciphertext
AES key	Symmetric key	1. Generated using FIPS 186-2 Appendix 3.1 RNG. 2. Generated using Diffie-Hellman key agreement. 3. Derived from TLS master secret. 4. Input in encrypted form.	Via TLS sessions in encrypted form.	Plaintext in volatile memory only	Zeroized after use	Encrypt plaintext/ Decrypt ciphertext
RSA private key	Private key	Generated internally using FIPS 186-2 Appendix 3.1 RNG.	Never output	Plaintext in hard disk	Zeroized when new key pair is generated	Decrypt ciphertext/ Sign messages (usually hash values)
RSA public key	Public key	1. Generated internally using FIPS 186-2 Appendix 3.1 RNG. 2. Imported in plaintext form.	Via TLS session in plaintext form	Plaintext in hard disk	Zeroized when new key pair is generated	Encrypt plaintext/ Verify signatures
ECDSA private key	Private key	Generated internally	Never output	Plaintext in hard disk	Zeroized when new key pair is generated	Sign messages (usually hash values)
ECDSA public key	Public key	1. Generated internally using FIPS 186-2 Appendix 3.1 RNG. 2. Imported in plaintext form.	Via TLS session in plaintext form	Plaintext in hard disk	Zeroized when new key pair is generated	Verify signatures
Diffie-Hellman public keys *p*,*g*	Public keys	1. Generated internally using FIPS 186-2 Appendix 3.1 RNG. 2. Input in plaintext form.	In plaintext form	Plaintext in hard disk	Zeroized when new keys are generated	Establish symmetric keys
Diffie-Hellman private keys *a*,*b*	Private key	Generated internally using FIPS 186-2 Appendix 3.1 RNG.	Never output	Plaintext in hard disk	Zeroized when new keys are generated	Establish symmetric keys

Key	Key Type	Generation / Input	Output	Storage	Zeroization	Use
FIPS 186-2 Appendix 3.1 RNG seed	RNG seed	Imported from client application in plaintext form	Never output	Plaintext in volatile memory only	Zeroized when new seed is fed	Generate random numbers
TLS master secret	TLS master secret	1. Generated internally using FIPS 186-2 Appendix 3.1 RNG. 2. Input via TLS sessions in encrypted form	Via TLS session in encrypted form	Plaintext in volatile memory only	Zeroized when TLS session is over	Derive keys in TLS sessions

2.6.1 Key Generation

The module uses NIST, FIPS 186-2 Appendix 3.1 RNG to generate cryptographic keys. This RNG is a FIPS 140-2 approved RNG as specified in Annex C to FIPS PUB 140-2.

2.6.2 Key Input/Output

RSA and ECDSA public keys are output from and input into the kernel in plaintext form. Symmetric keys are input into and output from the kernel in encrypted form.

2.6.3 Key Storage

Session keys are stored in volatile memory in plaintext. RSA and ECDSA key pairs are stored in hard disk in plaintext.

2.6.4 Key Zeroization

Keys are zeroized when they are no long used, RSA and ECDSA key pairs are zeroized when new ones are generated.

The zeroization of the keys is carried out by overwriting the storage or memory with zeros.

2.7 Self-Tests

The Tumbleweed Security Kernel performs the following self-tests at power-up:

- Software integrity test using HMAC-SHA-1.
- TDES KAT with 3 independent keys (56 bits each) in CBC mode.
- AES KAT with a 128-bit key in ECB mode.
- SHA-256 KAT.
- SHA-512 KAT.
- RSA KAT with 1024-bit keys for signature generation/verification and encryption/decryption.
- FIPS 186-2 Appendix 3.1 RNG KAT.
- Pair-wise consistency test for ECDSA keys.

The conditional self-test performed by the module include the following three tests.

- Pair-wise consistency test for RSA keys.
- Pair-wise consistency test for ECDSA keys.
- Continuous RNG Test.

If the self-tests fail, an exception will be thrown on the failure. The user is then alerted that the self-tests failed, and the application will not load and will enter an error state. When in the error state, execution of the kernel is halted, which inhibits the output of data from the module.

2.8 Design Assurance

Tumbleweed uses the Concurrent Versions System (CVS) version 1.1.22 for configuration management of source code and documentation. See the CVS project website http://www.nongnu.org/cvs/ for more information.

Additionally, Microsoft Visual Source Safe (VSS) version 6.0 is used to provide configuration management for the Tumbleweed Security Kernel's FIPS documentation. This software provides access control, versioning, and logging.

2.9 Mitigation of Other Attacks

This section is not applicable. The module does not claim to mitigate any attacks beyond the FIPS 140-2 level 1 requirements for this validation.

3 Secure Operation

The Tumbleweed Security Kernel 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.

3.1 Crypto Officer Guidance

The Crypto Officer is responsible for installing, uninstalling, configuring, and managing the module and running the power-up self-tests. Before installing the module, the Crypto Officer should make sure that the specific OS is in single user mode.

3.1.1 Operation System Configuration

The Crypto Officer must maintain control of the installation media.

FIPS 140-2 mandates that a cryptographic module be limited to a single user at a time. Before the module can be installed, the Crypto Officer must have a standard PC or mainframe computer running on one of the OS listed in Column 1 of Table 1. The OS being used must be configured for single user mode and disallow remote login.

To configure Windows for single user mode, the Crypto Officer must ensure that all remote guest accounts are disabled in order to ensure that only one human operator can log into the Windows OS at a time. The services that need to be turned off for Windows are

- Fast-user switching (irrelevant if PC or server is a domain member)
- Terminal services
- Remote registry service
- Secondary logon service
- Telnet service
- Remote desktop and remote assistance service

Once the Windows OS has been properly configured, the Crypto Officer can use the system “Administrator” account to install software, uninstall software, and administrate the module.

The specific procedure to configure a Linux System for single user mode is described below.

1. Login as the “root” user.

2. Edit the system files /etc/passwd and /etc/shadow and remove all the users except “root” and the pseudo-users. Make sure the password fields in /etc/shadow for the pseudo-users are either a star (*) or double exclamation mark (!!). This prevents login as the pseudo-users.

3. Edit the system file /etc/nsswitch.conf and make “files” the only option for “passwd”, “group”, and “shadow”. This disables Network Information Service and other name services for users and groups.

4. In the /etc/xinetd.d directory, edit the files “rexec”, “rlogin”, “rsh”, “rsync”, “telnet”, and “wu-ftpd”, and set the value of “disable” to “yes”.

5. Reboot the system for the changes to take effect.

More information can be found at http://csrc.nist.gov/cryptval/140-1/CMVPFAQ.pdf.

Once the operating system has been properly configured, the Crypto Officer can use the system “root” account to install/uninstall software and administrate the module.

The specific procedure to configure SunOS for single user mode is described below:

1. Login as the "root" user.

2. Edit the system files /etc/passwd and /etc/shadow and remove all the users except "root" and the pseudo-users (daemon users). Make sure the password fields in /etc/shadow for the pseudo-users are either a star (*) or double exclamation mark (!!). This prevents login as the pseudo-users. Also make sure the shell for daemon users is /dev/null, or something else that is not exploitable.

3. Edit the system file /etc/nsswitch.conf and make "files" the only option for "passwd", "group", and "shadow". This disables NIS and other name services for users and groups.

4. Edit the system file /etc/inet/inetd.conf, and comment out all unnecessary services (by prepending a hash ('#') sign to the beginning of each unnecessary service line).

   sadmind - Solstice network administration agent server
   rpc.ttdbserverd - Sun tool-talk server
   kcms_server - Kodak Color Management System server
   fs.auto - Sun font server
   cachefsd - NFS cache service
   rquotad - remote disk quota server
   rpc.metad - Disksuite remote metaset service
   rpc.metamhd - Disksuite remote multihost service
   rpc.metamedd - Disksuite component service
   ocfserv - Smartcard service
   dtspcd - Part of the CDE package
   rpc.cmsd - remote calendar server
   in.comsat - biff, mail notification server
   in.talkd - talk server
   gssd - RPC application authentication
   in.tnamed - deprecated name server
   rpc.smserverd - removable media device sensor service (disabling requires manual CD mounting)
   dcs - remote dynamic configuration server
   ftpd - ye olde FTP server
   ktkt_warnd - Kerberos warning server
   chargen - deprecated network service
   daytime - deprecated network time
   time - legacy time service
   discard - deprecated network service
   echo - network 'echo' service
   ufsd - part of RPC
   in.uucpd - unix-to-unix copy server

5. Disable service startup scripts within /etc/rc2.d. Many additional services (not bound to inetd) are started by default. To disable startup scripts, files can be renamed to make sure they do not begin with a capital ‘S’ (which denotes Startup). Disable startup scripts that are not pertinent to the setup.

   nscd - NIS-related
   snmpdx - SNMP services
   cachefs.daemon - NFS-caching
   rpc - Remote Procedure Call services
   sendmail – Sendmail
   lp - line printer daemon
   pppd - Point-to-point Protocol services
   uucp - Unix-to-Unix copy daemon
   ldap - LDAP services

6. Reboot the system for the changes to take effect.

Once the operating system has been properly configured, the Crypto Officer can use the system “root” account to install/uninstall software and administrating the module.

The specific procedure to configure IBM AIX for single user mode is described below:

1. Log in as the "root" user.
2. Edit the system file /etc/passwd and remove all the users except "root" and the pseudo-users. Make sure that for the pseudo-users, either the password fields are a star (*) or the login shell fields are empty. This prevents login as the pseudo-users.
3. Remove all lines that begin with a plus sign (+) or minus sign (-) from /etc/passwd and /etc/group. This disables NIS and other name services for users and groups.
4. Edit the system file /etc/inetd.conf. Remove or comment out the lines for remote login, remote command execution, and file transfer daemons such as telnetd, rlogind, krlogind, rshd, krshd, rexecd, ftpd, and tftpd.
5. Reboot the system for the changes to take effect.

Once the operating system has been properly configured, the Crypto Officer can use the system “root” account to install/uninstall software and administrating the module.

3.1.2 Initialization

The software module will be provided to the users by Tumbleweed Communications Corp. along with the client applications, including Validation Authority, SecureTransport, and MailGate. The module is installed during installation of the client application. The installation procedure is described in the client application’s installation manual.

The module must be installed, configured, and started before operators may utilize its features.

3.1.3 Zeroizaion

Zeroization of keys and other CSPs is controlled and performed by client applications. Zeroization may be manually invoked by rebooting the computer on which the kernel is running. Uninstalling the client application also results in zeroization of all keys and other CSPs.

3.1.4 Management

The Crypto Officer does not perform any management of the kernel after installation and configuration. The management tasks are conducted by the client application.

3.2 User Guidance

The module’s cryptographic functionality and security services are provided via client applications. Only the algorithms listed in Section 2.6 should be used by the client application. End-user instructions and guidance are provided in the user manual and technical support documents of the individual client application software. Although the end-users do not have any ability to modify the configuration of the module, they should check that the client application is present and enabled and thereby providing cryptographic protection.

4 Acronyms

Table 8 – Acronyms

Acronym	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
BIOS	Basic Input/Output System
CA	Certificate Authority
CBC	Cipher Block Chaining
CD	Compact Disc
CFB	Cipher Feedback
CMVP	Cryptographic Module Validation Program
CPU	Central Processing Unit
CSP	Critical Security Parameter
CVS	Concurrent Versions System
DER	Distinguished Encoding Rules
DLL	Dynamic Link Library
ECB	Electronic Codebook
ECDSA	Elliptic Curve Digital Signature Algorithm
EDI	Electronic Data Interchange
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FCC	Federal Communication Commission
FIPS	Federal Information Processing Standard
HMAC	(Keyed-) Hash MAC
IDE	Integrated Drive Electronics
IP	Internet Protocol
ISA	Instruction Set Architecture
KAT	Known Answer Test
MAC	Message Authentication Code
N/A	Not Applicable
NIST	National Institute of Standards and Technology
PEM	Privacy-enhanced Electronic Mail
OFB	Output Feedback
OS	Operating System
PC	Personal Computer
PCI	Peripheral Component Interconnect
PKCS	Public Key Cryptography Standard

Acronym	Definition
PKI	Public Key Infrastructure
RAM	Random Access Memory
RNG	Random Number Generator
ROM	Read Only Memory
RSA	Rivest, Shamir and Adleman
SHA	Secure Hash Algorithm
SO	Shared Object
TDES	Triple Data Encryption Standard
TLS	Transport Layer Security
USB	Universal Serial Bus
VSS	Visual Source Safe
XML	Extensible Markup Language