# Axway Inc.

# Axway Security Kernel

(Software Versions: 3.0 and 3.0.1)

# FIPS 140-2
# Non-Proprietary Security Policy

**Level 1 Validation**

**Document Version 2.2**

Prepared By:

**Axway Inc.**
2600 Bridge Parkway, Suite 201
Redwood City, CA 94065
Phone: (650) 801-3100
Fax: (650) 801-3101
http://www.axway.com

# Revision History

| Version | Modification Date | Modified By | Description of Changes |
|---|---|---|---|
| 1.0 | 2013-01-10 | Anubhav Soni | Initial draft. |
| 1.0.1 | 2013-02-15 | Anubhav Soni<br>Prabhakar Mangam | Updated with compliance for OSes and Axway logos etc. |
| 1.1 | 2013-03-08 | Marc Ireland<br>Prabhakar Mangam | Reviewed and includes comments from InfoGard (Marc Ireland). |
| 1.2 | 2013-03-28 | Marc Ireland<br>Prabhakar Mangam<br>Anubhav Soni and<br>Hristo Todorov | Reviewed and updated document based on broader review. |
| 1.3 | 2013-04-16 | Anubhav Soni | Updated known answer test section |
| 1.4 | 2013-04-29 | Anubhav Soni | Updated to include non-approved services |
| 1.5 | 2013-05-01 | Anubhav Soni,<br>Marc Ireland,<br>Prabhakar Mangam | Reviewed and updated with comments from InfoGard (Marc Ireland) |
| 1.6 | 2013-05-02 | Mark S, Anubhav S, Enamul H, Prabhakar M, Hristo T | Axway internal reviewed |
| 1.7 | 2013-05-06 | Marc Ireland<br>Anubhav Soni<br>Prabhakar Mangam | Reviewed & updated Services and non-approved services tables |
| 1.8 | 2013-05-23 | Luis Garcia,<br>Prabhakar Mangam,<br>Anubhav Soni | Updated with comments from InfoGard review. |
| 1.9 | 2014-03-11 | Prabhakar Mangam | Updated with comments from InfoGard review |
| 2.0 | 2014-08-07 | Marc Ireland | Updated with comments from InfoGard review |
| 2.1 | 2015-03-09 | Enamul Haque | Updated kernel version |
| 2.2 | 2016-05-16 | Bill Shine | Updated with comments from InfoGard review.<br>Updated kernel version |

# Table of Contents

# Table of Figures

# Table of Tables

## 1.1 Purpose

This document is a non-proprietary Cryptographic Module Security Policy for the Axway Security Kernel from Axway Inc. This Security Policy describes how the Axway 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/groups/STM/cmvp/index.html.

In this document, the Axway 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 Axway Security Kernel. The Validation Authority Suite and MailGate are currently the only applications making use of the Axway Security Kernel. However, it is expected that a range of products developed by Axway Inc., will be supported by the Axway 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 Axway website (http://www.axway.com) contains information on the full line of products from Axway.
- The CMVP website (http://csrc.nist.gov/groups/STM/cmvp/index.html) 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 were reviewed by InfoGard under contract to Axway. With the exception of this Non-Proprietary Security Policy, the FIPS 140-2 Validation Documentation is proprietary to Axway and is releasable only under appropriate non-disclosure agreements. For access to these documents, please contact Axway.

# 2 Axway Security Kernel

## 2.1 Overview

The Axway Security Kernel (versions 3.0 and 3.0.1) is a software cryptographic module implemented as two dynamic link libraries (DLLs) on Windows or two Shared Objects (SOs) on Linux and SunOS. The Axway 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.

The module has been tested and validated on Microsoft Windows 2012 (64 bit) on a Dell PowerEdge R620 Server, RHEL 6.3 (64 bit) on a Dell PowerEdge R620 Server, Solaris 10 on a Sun Blade T6300 Server (64 bit) (each configured for single user mode). Compliance is maintained on following platforms including (but not limited to):

- Windows XP 32 and 64 bit
- Windows 2003 R2 32 and 64 bit
- Windows Vista 32 and 64 bit
- Windows 2008 32 and 64 bit
- Windows 7 32 and 64 bit
- Windows 2008 R2 64 bit
- Windows 2012 64bit
- Windows 8 32 and 64 bit
- RHEL 5.X 32 and 64 bit
- RHEL 6.X 32 and 64 bit
- SLES 10 64 bit
- SLES 11 64 bit
- Solaris 9/Zones Solaris 9 32 and 64 bit
- Solaris 10/Zones Solaris 10 32 and 64 bit
- Solaris 11/Zones Solaris 11 32 and 64 bit

The platforms supported by the module are binary compatible with the platforms used in the FIPS validation. The CMVP makes no statement as to the correct operation of the module on the operational environments for which testing was not performed.

**Table 1 – Binary Forms of the Kernel**

| Operating Systems | Binary File Names |
|---|---|
| Windows XP 32 and 64 bit<br>Windows 2003 R2 32 and 64 bit<br>Windows Vista 32 and 64 bit<br>Windows 2008 32 and 64bit<br>Windows 7 32 and 64 bit<br>Windows 2008 R2 64 bit<br>Windows 2012 64bit<br>Windows 8 32 and 64 bit | libeay32-TMWD.dll<br>ssleay32-TMWD.dll |
| Linux kernel 2.6.13 and later versions<br>RHEL 5.X 32 and 64 bit<br>RHEL 6.X 32 and 64 bit<br>SLES 10 and SLES 11 64 bit | libcrypto-TMWD.so.1.0.0<br>libssl-TMWD.so. 1.0.0 |
| Solaris 9/ Zones Solaris 9 32 and 64 bit<br>Solaris 10/ Zones Solaris 10 32 and 64 bit<br>Solaris 11/Zones Solaris 11 32 and 64 bit | libcrypto-TMWD.so.1.0.0<br>libssl-TMWD.so.1.0.0 |

The kernel is built upon a custom version of OpenSSL 1.0.0p. As a cryptographic module, the Axway Security Kernel presents an identical application programming interface (API) to several products of the Axway Inc., including Axway Validation Authority Suite and MailGate.

The cryptographic capabilities of Validation Authority and MailGate are provided by the Axway 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. The MailGate platform is capable of performing tasks such as email encryption, secure file collaboration, network defense, content filtering, and data protection.

The Axway Security Kernel supports the following FIPS-approved algorithms:

**Table 2 – FIPS Approved Algorithms**

| FIPS Approved Algorithms | |
|---|---|
| AES (Certs. # 2446 and #3215) | Encrypt/Decrypt - ECB, CBC, CFB128, OFB and CTR modes; 128,192 and 256 bits |
| TDES (Certs. #1511 and #1830) | Triple-DES Encrypt/Decrypt: 168 bits (3-key), in TECB, TCBC, TCFB64, and TOFB modes |
| SHA (Certs. # 2080 and #2663) | SHA-1<br>SHA-2: SHA-224, SHA-256, SHA-384 and SHA-512 |
| HMAC (Certs. #1510 and #2028) | HMAC-SHA-1: 160-bit MAC and 512-bit key |
| DSA (Certs. #760 and #918) | FIPS 186-2 DSA using SHA-1:<br>- Signature Verification; 1024-bit<br>- PQG Verification; 1024-bit |
| ECDSA (Certs. #402 and #594) | FIPS 186-2 ECDSA using SHA-1:<br>- Key Generation using curves P-224, K-233, and B-233<br>- PKV using curves P-192, P-224, B-163, and B-233<br>- Signature Verification using curves P-192, P-224, K-163, K-233, B-163, and B-233 |
| RSA (Certs. #1257 and #1638) | FIPS 186-2 RSA:<br>- ANSI X9.31 Key Generation; 2048 and 4096-bit<br>- PKCS #1 V1.5 Signature Generation: 4096-bit using SHA-2<br>- PKCS #1 V1.5 Signature Verification: 1024, 2048, and 4096-bit using SHA-1 and SHA-2 |
| CVL (Certs. #76 and #439) | SP 800-135 KDF |

The Axway Security Kernel supports the following non-Approved but allowed algorithms:

- Diffie-Hellman 2048-bit - for key establishment in TLS sessions (key agreement; key establishment methodology provides 112 bits of encryption strength)
- EC Diffie-Hellman using P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, and B-571 curves (key agreement; key establishment methodology provides between 112 and 256 bits of encryption strength)
- RSA 2048-bit - for key transport in TLS sessions (key wrapping; key establishment methodology provides 112 bits of encryption strength)
- MD5 for use within TLS only.

Apart from the tested algorithms, the Axway Security Kernel also provides the following non-approved algorithms:

- RNG FIPS 186-2 Appendix 3.1
- Triple-DES Encrypt/Decrypt: 2-key in TECB, TCBC, TCFB64, and TOFB modes
- FIPS 186-2 DSA using SHA-1: Signature generation, PQG generation, and Key generation; 1024-bit (Cert. #760)

- FIPS 186-2 ECDSA using SHA-1: Signature generation (P-192, P-224, K-163, K-233, B-163, and B-233 curves), Key generation (P-192, K-163, and B-163 curves) (Cert. #402)
- FIPS 186-2 RSA: ANSI X9.31 Key generation (1024-bit), PKCS #1 V1.5 Signature generation (1024-bit using SHA-1 and SHA-2; 4096-bit using SHA-1) (Cert. #1257)
- Diffie-Hellman 1024-bit (key agreement; key establishment methodology provides 80 bits of encryption strength; non-compliant)
- EC Diffie-Hellman using P-192, K-163 or B-163 curves (key agreement; key establishment methodology provides less than 112 bits of encryption strength; non-compliant)
- RSA Encrypt/Decrypt 1024-bit (key wrapping; key establishment methodology provides 80 bits of encryption strength; non-compliant)
- Blowfish
- Camellia
- Cast
- DES
- des_old
- DTLS1
- ec, krb5_asn
- KSSL
- MD4, MD5
- MDC2, RC2, RC4
- RIPEMD,
- Seed
- Whirlpool

The Axway Security Kernel supports both a FIPS-Approved and non-Approved mode of operation. An operator can determine which mode the module is in based on the service being performed. Please see section 2.3 below for a list of Approved and non-Approved services.

The Axway Security Kernel is validated at FIPS 140-2 section levels shown in Table 2. Note that in Table 2, N/A indicates "Not Applicable".

**Table 3 – 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 (Electromagnetic Interference/Compatibility) | 1 |
| 9 | Self-Tests | 1 |
| 10 | Design Assurance | 1 |
| 11 | Mitigation of Other Attacks | N/A |

## 2.2 Module Interfaces

**The Axway 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 Axway Security Kernel running on different OSs. 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), and Public Key Cryptography Standard (PKCS) #12 certificate management.

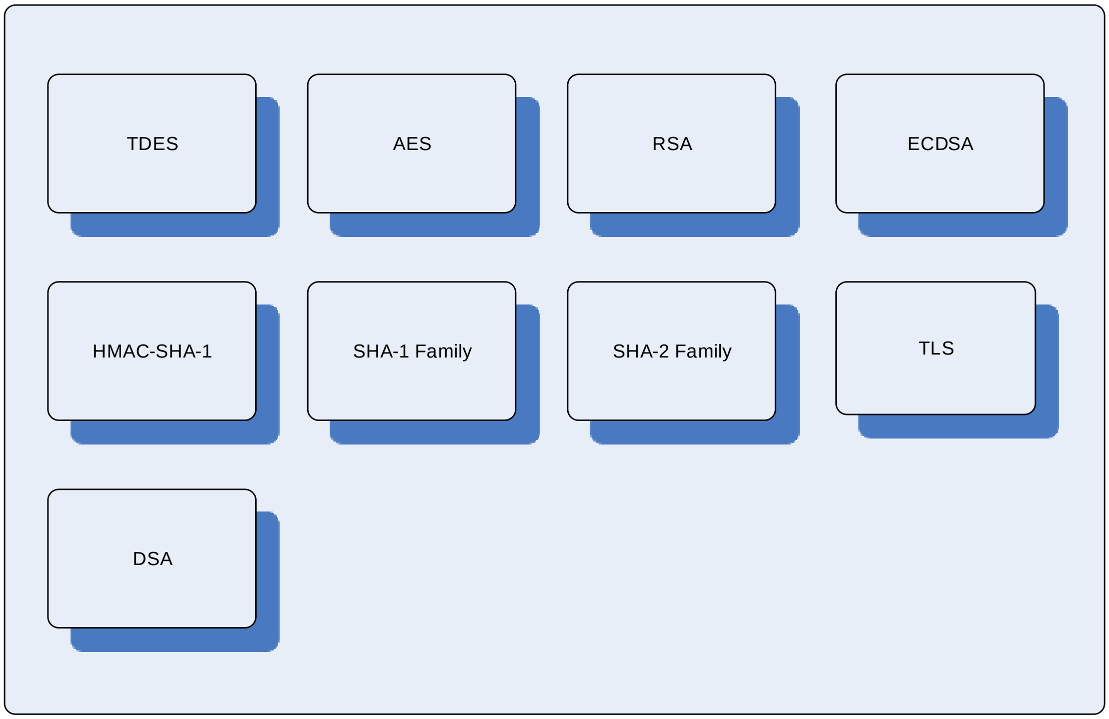


**Figure 1 – Logical Cryptographic Boundary**

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.

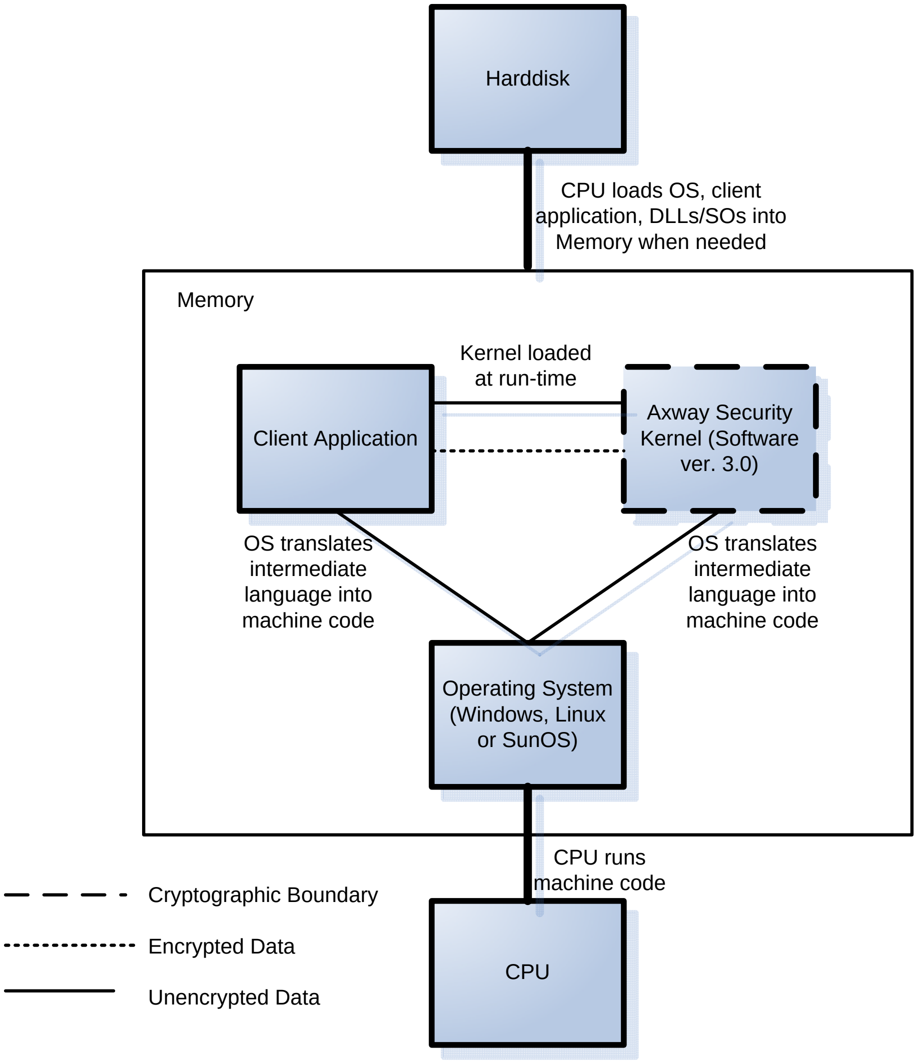
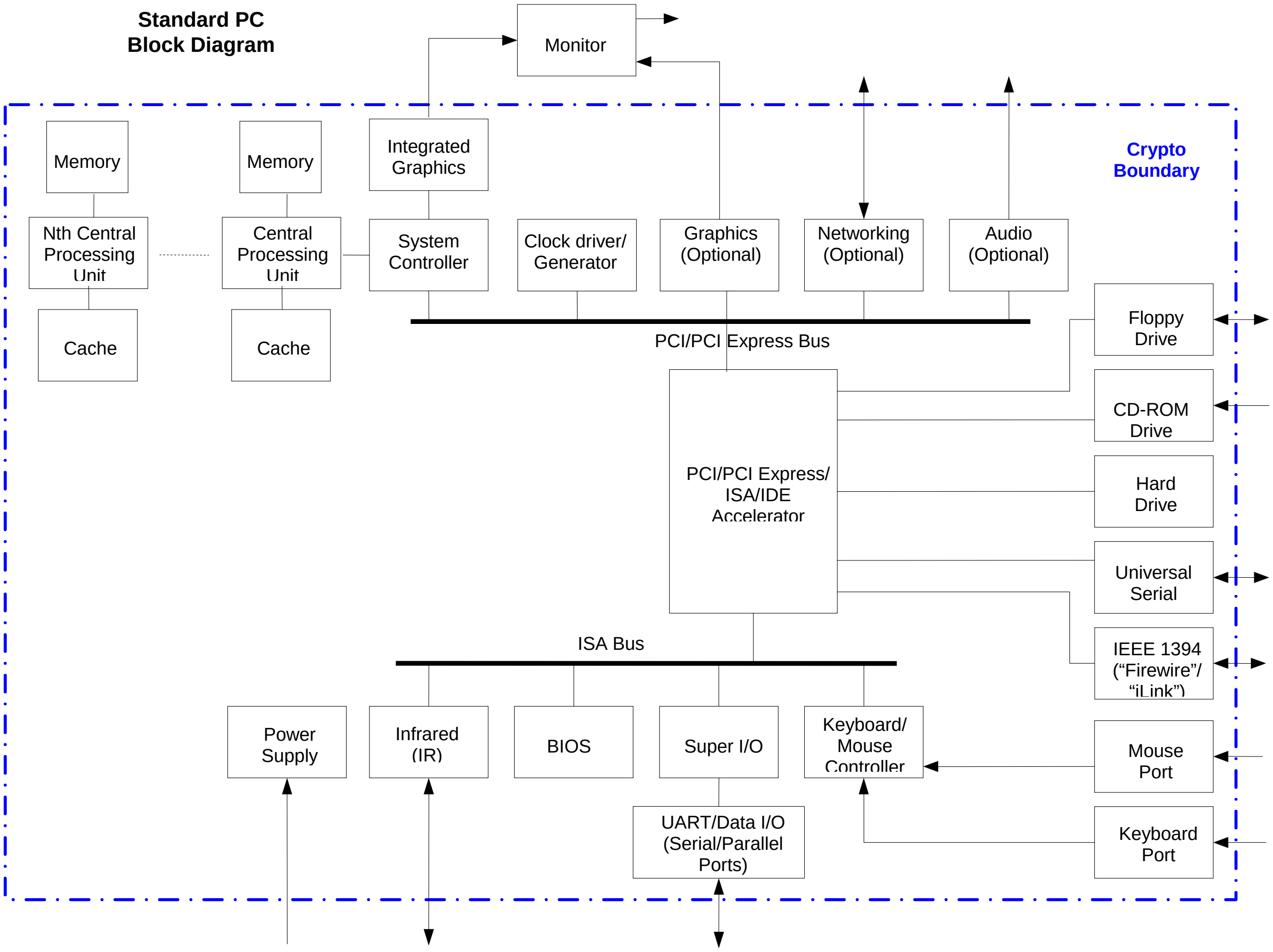

Harddisk
CPU loads OS, client application, DLLs/SOs into Memory when needed
Memory
Kernel loaded at run-time
Client Application
Axway Security Kernel (Software ver. 3.0)
OS translates intermediate language into machine code
OS translates intermediate language into machine code
Operating System (Windows, Linux or SunOS)
CPU runs machine code
CPU
Cryptographic Boundary
Encrypted Data
Unencrypted Data

**Figure 2 – Logical Cryptographic Boundary and Interactions with Surrounding Components**

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 4 – FIPS 140-2 Logical Interfaces**

| Logical Interface | Axway 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 5 gives a brief description of the roles and their privileges.

**Table 5 – Module Roles and Privileges**

| Role Name | Role Privileges |
|---|---|
| Crypto Officer | 1. Installing/uninstalling the kernel on the specific platform.<br>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 6, where CSP refers to Critical Security Parameter.

**Table 6 – 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-1, SHA-256, SHA-512, DSA, TLS-KDF; (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 7.

**Table 7 – User Services**

| Service | Description | Input | Output | CSP and Type of Access |
|---|---|---|---|---|
| TDES encryption | Encrypt using TDES | Command, plaintext, keys | Status, ciphertext | TDES symmetric keys - READ |
| TDES decryption | Decrypt using TDES | Command, ciphertext, keys | Status, plaintext | TDES symmetric keys - READ |
| AES encryption | Encrypt plaintext using AES | Command, plaintext, key | Status, ciphertext | AES symmetric key - READ |
| AES decryption | Decrypt AES-encrypted ciphertext | Command, ciphertext, key | Status, plaintext | AES symmetric key - READ |
| RSA encryption | Encrypt plaintext using RSA 2048-bit | Command, plaintext, RSA public key | Status, ciphertext | RSA public key - READ |
| RSA decryption | Decrypt RSA-encrypted ciphertext (2048-bit) | Command, ciphertext, RSA private key | Status, plaintext | RSA private key - READ |
| RSA signature generation | Sign data using RSA 4096-bit w/SHA-2 | Command, data to be signed, RSA private key | Status, digital signature | RSA private key - READ |
| RSA signature verification | Verify an RSA signature (1024, 2048, or 4096-bit using SHA-1 or SHA-2) | Command, data and signature, RSA public key | Status, acceptance/denial | RSA public key - READ |
| ECDSA signature verification | Verify an ECDSA signature (using curves P-192, P-224, K-163, K-233, B-163, and B-233) | Command, data and signature | Status, acceptance/denial | ECDSA public key - READ |
| SHA-512 digest generation | Generate a SHA-512 digest | Command, message to be hashed | Status, 512-bit message digest | None |
| SHA-384 digest generation | Generate a SHA-384 digest | Command, message to be hashed | Status, 384-bit message digest | None |
| SHA-256 digest generation | Generate a SHA-256 digest | Command, message to be hashed | Status, 256-bit message digest | None |
| SHA-224 digest generation | Generate a SHA-224 digest | Command, message to be hashed | Status, 224-bit message digest | None |
| SHA-1 digest generation | Generate a SHA-1 digest | Command, message to be hashed | Status, 160-bit message digest | None |
| 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 |
| DSA signature verification | Verify a DSA signature (1024-bit) | Command, data and signature | Status, acceptance/denial | DSA public key - READ |

| Service | Description | Input | Output | CSP and Type of Access |
|---|---|---|---|---|
| Random number generation | Generate a random number | Command, seed, length | Status, random number | Seed<br>- READ/WRITE |
| Establish a TLS session<br>Note: the TLS protocol has not been reviewed or tested by the CAVP or the CMVP. | Establish a new TLS session | Command | Status, session ID | None |
| Show status | Show status of a service (function call) | Command | Status | None |
| DH 2048-bit | Compute shared secret key | Command and key | Shared secret key | Shared secret key – READ |
| ECDH (using P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, and B-571 curves) | Compute shared secret key | Command and key | Shared secret key | Shared secret key – READ |
| Load Seed | Loads RNG seed from calling application | Command and seeding data | None | Seed<br>- READ/ WRITE |

**Note: The module generates cryptographic keys whose strengths are modified by available entropy. There is no assurance of the minimum strength of generated keys.**

### 2.3.3 Non-Approved services

The following are the non-Approved services available for User role, these provide non-Approved cryptographic services that include encryption, decryption, and authentication functionality. Descriptions of the non-Approved services available to the User role are provided in Table 7.

**Table 8 – Non-Approved Services**

| Service | Description | Input | Output |
|---|---|---|---|
| TDES key generation | Generate a TDES symmetric keys using FIPS 186-2 Appendix 3.1 RNG | Command, key length | Status, symmetric keys |
| 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 |
| RSA key-pair generation | Generate a RSA key-pair (2048 and 4096-bit) | Command, key length | Status, RSA private key and public key |
| ECDSA key-pair generation | Generate an ECDSA key-pair (curves P-224, K-233, and B-233) | Command, key length | Status, ECDSA private key and public key |
| HMAC-SHA-1 key generation | Generate a HMAC-SHA-1 symmetric key | Command | Status, 512-bit HMAC-SHA-1 symmetric key |
| DSA signature generation | Sign data using DSA (1024-bit w/SHA-1) | Command, data to be signed | Status, digital signature |
| ECDSA signature generation | Sign data using ECDSA (P-192, P-224, K-163, K-233, B-163, and B-233 curves) | Command, data to be signed | Status, digital signature |
| ECDSA Key Generation | Generate an ECDSA key-pair (P-192, K-163, and B-163 curves) | Command, key length | Status, ECDSA private key and public key |
| RSA Key Generation | Generate an RSA key-pair (1024-bit) | Command, key length | Status, RSA private key and public key |
| RSA signature generation | Sign data using RSA (1024-bit using SHA-1 and SHA-2; 4096-bit using SHA-1) | Command, data to be signed | Status, digital signature |
| RSA encryption | Encrypt plaintext using RSA 1024-bit | Command, plaintext, RSA public key | Status, ciphertext |
| RSA decryption | Decrypt RSA-encrypted ciphertext (1024-bit) | Command, ciphertext, RSA private key | Status, plaintext |
| DH 1024- bit | Compute shared secret key | Command and key | Shared secret key |
| ECDH (P-192, K-163 or B-163 curves) | Compute shared secret key | Command and key | Shared secret key |
| Blowfish encryption | Encrypt plaintext using Blowfish | Command, plaintext, key | Status, ciphertext |
| Blowfish decryption | Decrypt Blowfish -encrypted ciphertext | Command, ciphertext, key | Status, plaintext |
| Blowfish key generation | Generate an Blowfish symmetric key using FIPS 186-2 Appendix 3.1 RNG and set the key schedule | Command, key length | Status, symmetric key |
| Camellia encryption | Encrypt plaintext using Camellia | Command, plaintext, key | Status, ciphertext |

| Service | Description | Input | Output |
|---|---|---|---|
| Camellia decryption | Decrypt Camellia -encrypted ciphertext | Command, ciphertext, key | Status, plaintext |
| Camellia key generation | Generate an Camellia symmetric key using FIPS 186-2 Appendix 3.1 RNG and set the key schedule | Command, key length | Status, symmetric key |
| CAST encryption | Encrypt plaintext using CAST | Command, plaintext, key | Status, ciphertext |
| CAST decryption | Decrypt CAST -encrypted ciphertext | Command, ciphertext, key | Status, plaintext |
| CAST key generation | Generate an CAST symmetric key using FIPS 186-2 Appendix 3.1 RNG and set the key schedule | Command, key length | Status, symmetric key |
| DES/DES-Old encryption | Encrypt plaintext using DES/DES-Old | Command, plaintext, key | Status, ciphertext |
| DES/DES-Old decryption | Decrypt DES/DES-Old -encrypted ciphertext | Command, ciphertext, key | Status, plaintext |
| DES/DES-Old key generation | Generate an DES/DES-Old symmetric key using FIPS 186-2 Appendix 3.1 RNG and set the key schedule | Command, key length | Status, symmetric key |
| Establish a DTLS session | Establish a new DTLS session | Command | Status, Session ID |
| KRB5 | Generate Kerberos ticket | Command | Status, Ticket |
| Kernel-SSL(K-SSL) | Establish a new SSL session at OS kernel level | Command | Status, Session ID |
| MD4 digest generation | Generate a MD4 digest | Command, message to be hashed | Status, 128-bit message digest |
| MD5 digest generation | Generate a MD5 digest | Command, message to be hashed | Status, 128-bit message digest |
| MDC2 digest generation | Generate a MDC2 digest | Command, message to be hashed | Status, 128-bit message digest |
| RC2 encryption | Encrypt plaintext using RC2 | Command, plaintext, key | Status, ciphertext |
| RC2 decryption | Decrypt RC2 -encrypted ciphertext | Command, ciphertext, key | Status, plaintext |
| RC2 key generation | Generate an RC2 symmetric key using FIPS 186-2 Appendix 3.1 RNG and set the key schedule | Command, key length | Status, symmetric key |
| RC4 encryption | Encrypt plaintext using RC4 | Command, plaintext, key | Status, ciphertext |
| RC4 decryption | Decrypt RC4 -encrypted ciphertext | Command, ciphertext, key | Status, plaintext |
| RC4 key generation | Generate an RC4 symmetric key using FIPS 186-2 Appendix 3.1 RNG and set the key schedule | Command, key length | Status, symmetric key |
| RIPEMD digest generation | Generate a RIPEMD digest | Command, message to be hashed | Status, 128-bit message digest |
| SEED encryption | Encrypt plaintext using SEED | Command, plaintext, key | Status, ciphertext |
| SEED decryption | Decrypt SEED -encrypted ciphertext | Command, ciphertext, key | Status, plaintext |

| Service | Description | Input | Output |
|---|---|---|---|
| SEED key generation | Generate an SEED symmetric key using FIPS 186-2 Appendix 3.1 RNG and set the key schedule | Command, key length | Status, symmetric key |
| Whirlpool digest generation | Generate a Whirlpool digest | Command, message to be hashed | Status, 512-bit message digest |

## 2.4 Physical Security

The Axway 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 and SunOS operating systems. See Column 1 of Table 1 for a list of OSs 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 2012 (64 bit), RHEL 6.3 (64 bit), Solaris 10 (64 bit). 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 supports the following CSPs:

**Table 9 – 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.<br>2. Generated using Diffie-Hellman key agreement.<br>3. Derived from TLS master secret.<br>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.<br>2. Generated using Diffie-Hellman key agreement.<br>3. Derived from TLS master secret.<br>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.<br>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.<br>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.<br>2. Input in plaintext form. | In plaintext form | Plaintext in volatile memory | 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 volatile memory | 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.<br>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 |
| HMAC Key | Log Signing key | Generated internally using FIPS 186-2 Appendix 3.1 RNG | Never output | Plaintext in volatile memory only | Zeroized when the server has signed the last log file before shutdown | Log file signing |
| DSA Private Key | Signature generation, | 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) |
| DSA Public Key | Signature verification | 1. Generated internally using FIPS 186-2 Appendix 3.1 RNG.<br>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 |

**Note: The FIPS 186-2 RNG is no longer considered Approved for key generation. Keys generated internally using this RNG are not permitted for government use.**

### 2.6.1 Key Generation

The module uses NIST, FIPS 186-2 Appendix 3.1 RNG to generate cryptographic keys. This RNG is no longer a FIPS 140-2 approved RNG as specified in Annex C to FIPS PUB 140-2. Keys generated internally using this RNG would not be considered Approved. Utilizing keys that have been generated with the 186-2 RNG results in the module being in a non-Approved mode of operation.

### 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 longer used; RSA, DSA 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 Axway Security Kernel performs the following self-tests at power-up:

- Software integrity test using HMAC-SHA-1.
- TDES Encrypt KAT with 3 independent keys (56 bits each) in CBC mode.
- TDES Decrypt KAT with 3 independent keys (56 bits each) in CBC mode.
- AES Encrypt KAT with a 128-bit key in ECB mode.
- AES Decrypt KAT with a 128-bit key in ECB mode
- SHA-1 KAT.
- SHA-256 KAT. (also satisfies SHA-224 KAT).
- SHA-512 KAT. (also satisfies SHA-384 KAT).
- RSA Encrypt KAT with 1024-bit keys for encryption
- RSA Decrypt KAT with 1024-bit keys for decryption
- RSA Sign KAT with 1024-bit keys for signature generation
- RSA Verify KAT with 1024-bit keys for signature verification
- Pair-wise consistency test for ECDSA keys.
- TLS KDF (Key Derivation Function) Test
- DSA Sign KAT
- DSA Verify KAT

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.
- Pair-wise consistency test for DSA 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

Axway uses the Subversion for configuration management of source code and documentation including Axway Security Kernel's FIPS documentation. See the SVN project website http://subversion.apache.org for more information. 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 Axway 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/groups/STM/cmvp/documents/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.

### 3.1.2 Initialization

The software module will be provided to the users by Axway Inc. along with the client applications, including Validation Authority 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 10 – 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 |
| DH | Diffie-Hellman |
| DSA | Digital Signature Algorithm |
| 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 |
| KDF | Key derivative function |
| 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 |

| Acronym | Definition |
|---|---|
| PC | Personal Computer |
| PCI | Peripheral Component Interconnect |
| PKCS | Public Key Cryptography Standard |
| 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 |