Microsoft

Operating System

Microsoft Kernel Mode Cryptographic Module

FIPS 140-1 Documentation: Security Policy

6/28/2002 5:00 PM

Abstract

This document specifies the security policy for the Kernel Mode Cryptographic Module (FIPS.SYS) as described in FIPS PUB 140-1.

CONTENTS

INTRODUCTION

Microsoft Kernel Mode Cryptographic Module (FIPS.SYS) is a FIPS 140-1 Level 1 compliant, general-purpose, software-based, cryptographic module residing at the Kernel Mode level of the Windows Operating System. It runs as a kernel mode export driver (a kernel-mode DLL) and encapsulates several different cryptographic algorithms in an easy-to-use cryptographic module accessible by other kernel mode drivers. It can be linked into other kernel mode services to permit the use of FIPS 140-1 Level 1 compliant cryptography. Windows 2000 with SP2 FIPS.SYS was validated for FIPS 140-1 Level 1 in Jul 2000, Certificate # 106.

Cryptographic Boundary

The Kernel Mode Cryptographic Module (FIPS.SYS) consists of a single kernel mode export driver (SYS). The cryptographic boundary for FIPS.SYS is defined as the enclosure of the computer system on which the cryptographic module is to be executed. The physical configuration of the module, as defined in FIPS PUB 140-1, is Multi-Chip Standalone.

SECURITY POLICY

FIPS.SYS operates under several rules that encapsulate its security policy.

- FIPS.SYS is supported on Windows XP or later.
- FIPS.SYS provides no user authentication; however, it relies on Microsoft Windows XP for the authentication of users.
- FIPS.SYS enforces a single role, Authenticated User, which is a combination of the User and Cryptographic Officer roles as defined in FIPS PUB 140-1.
- All users authenticated by Microsoft Windows XP employ the Authenticated User role.
- All cryptographic services provided by FIPS.SYS are available to kernel mode system services, which are a part of Windows operating system trusted computer base (TCB[1]).
- Windows XP operating system requires each user to be successfully authenticated before any system services may act on behalf of that user.
- All services implemented within FIPS.SYS are available to the Authenticated User role.
- DSSENH supports the following FIPS approved algorithms: DES, 3DES, SHA-1, and HMAC
- FIPS.SYS performs the following self-tests upon power up:
  - DES ECB encrypt/decrypt
  - 3DES (3 key) ECB encrypt/decrypt
  - DES CBC encrypt/decrypt
  - 3DES (3 key) CBC encrypt/decrypt
  - 3DES ECB encrypt/decrypt
  - SHA-1 hash
  - Keyed-Hash Message Authentication Code (HMAC)[2]

[1] The TCB is the part of the operating system that is designed to meet the security functional requirements of the Controlled Access Protection Profile, which can be found at <http://www.radium.ncsc.mil/tpep/library/protection_profiles/index.html>. At this time, Windows XP has not been evaluated.

[2] The previous version of FIPS-140-1 validated FIPS.SYS on Windows 2000 with SP2 does not support Keyed-Hash Message Authentication Code (HMAC), which is a new FIPS proposed by NIST (http://csrc.nist.gov/encryption/hmac/index.html) in Jan 2001.

SPECIFICATION OF ROLES

FIPS.SYS combines the User and Cryptographic Officer roles (as defined in FIPS PUB 140-1) into a single role called the Authenticated User role. The Authenticated User may access all services implemented in the cryptographic module. Windows XP operating system requires each user to be successfully authenticated before any system services may act on behalf of that user.
To use a DES or Triple DES function, a kernel mode system service needs to provide a DES or Triple DES key respectively to the crypto module. Keys are zeroized after FIPS.SYS completes a DES or Triple DES function with the keys.

Maintenance Roles

Maintenance roles are not supported by FIPS.SYS.

Multiple Concurrent Operators

FIPS.SYS is intended to run on Windows XP or later in Single User Mode. When run in this configuration, multiple concurrent operators are not supported.

SPECIFICATION OF SERVICES

The following list contains all services available to an operator. All services are accessible by all Authenticated Users, the one and only role supported by FIPS.SYS.

Key Storage Services

FIPS.SYS does not store keys. DES, Triple DES, and HMAC keys are zeroized after used.

Cryptographic Module Power Up and Power Down

DriverEntry

Each Windows XP driver must have a standard initialization routine DriverEntry in order to be loaded. The Windows XP Loader is responsible to call the DriverEntry routine. The DriverEntry routine must have the following prototype.

```
NTSTATUS
(*PDRIVER_INITIALIZE) (
        IN      PDRIVER_OBJECT      DriverObject,
        IN      PUNICODE_STRING     RegistryPath
        );
```

The input DriverObject represents the driver within the Windows XP system. Its pointer allows the DriverEntry routine to set an appropriate entry point for its DriverUnload routine in the driver object.

The RegistryPath input to the DriverEntry routine points to a counted Unicode string that specifies a path to the driver's registry key \Registry\Machine\System\CurrentControlSet\Services\FIPS.

DriverUnload

It is the entry point for the driver's unload routine. The pointer to the routine is set by the DriverEntry routine in the DriverUnload field of the DriverObject when the driver initializes. An Unload routine is declared as follows:

```
VOID
(*PDRIVER_UNLOAD) (
        IN      PDRIVER_OBJECT      DriverObject
        );
```

When the driver is no longer needed, the Windows XP Kernel is responsible to call the DriverUnload routine of the associated DriverObject.

Key Formatting

The following functions provide interfaces to the cryptomodule's key formatting functions.

```
FipsDesKey
VOID
FipsDesKey(
        DESTable *      pDesTable,
        UCHAR *         pbKey
        )
```

The FipsDesKey function formats a DES cryptographic session key into the form of a DESTable struct. It fills in the DESTable struct with the decrypt and encrypt key expansions. Its second parameter points to the DES key of DES_BLOCKLEN (8) bytes. FipsDesKey zeroises its copy of the key before returning to the caller.

```
Fips3Des3Key
VOID
Fips3Des3Key(
        DES3TABLE *     pDES3Table,
        UCHAR *         pbKey
        )
```

The Fips3Des3Key function formats a Triple DES cryptographic session key into the form of a DES3Table struct. It fills in the DES3Table struct with the decrypt and encrypt key expansions. Its second parameter points to the Triple DES key of 3 * DES_BLOCKLEN (24) bytes. Fips3Des3Key zeroises its copy of the key before returning to the caller.

Random Number Generation

```
FipsGenRandom
BOOL
FIPSGenRandom(
        In OUT  UCHAR *         pb,
        IN      ULONG           cb
        );
```

The FipsGenRandom function fills the buffer pb with cb random bytes produced using a FIPS 140-1 compliant pseudo random number generation algorithm. The algorithm is the SHS based RNG from FIPS 186. Internally, the function compares each 160 bits of the buffer with the next 160 bits. If they are the same, the function returns FALSE. The caller may optionally specify the initial 160 bits in the pb buffer for the initiation of the comparison. This initial 160 bit sequence is used only for the comparison algorithm and it is not intended as caller supplied random seed.

During the function initialization, a seed, to which SHA-1 is applied to create the output random, is created based on the collection of all the following data.

- The process ID of the current process requesting random data
- The thread ID of the current thread within the process requesting random data
- A 32bit tick count since the system boot
- The current local date and time
- The current system time of day information consisting of the boot time, current time, time zone bias, time zone ID, boot time bias, and sleep time bias
- The current hardware-platform-dependent high-resolution performance-counter value
- The information about the system's current usage of both physical and virtual memory, and page file
- The local disk information including the numbers of sectors per cluster, bytes per sector, free clusters, and clusters that are available to the user associated with the calling thread
- A hash of the environment block for the current process
- Some hardware CPU-specific cycle counters
- The system processor performance information consisting of Idle Process Time, Io Read Transfer Count, Io Write Transfer Count, Io Other Transfer Count, Io Read Operation Count, Io Write Operation Count, Io Other Operation Count, Available Pages, Committed Pages, Commit Limit, Peak Commitment, Page Fault Count, Copy On Write Count, Transition Count, Cache Transition Count, Demand Zero Count, Page Read Count, Page Read Io Count, Cache Read Count, Cache Io Count, Dirty Pages Write Count, Dirty Write Io Count, Mapped Pages Write Count, Mapped Write Io Count, Paged Pool Pages, Non Paged Pool Pages, Paged Pool Allocated space, Paged Pool Free space, Non Paged Pool Allocated space, Non Paged Pool Free space, Free System page table entry, Resident System Code Page, Total System Driver Pages, Total System Code Pages, Non Paged Pool Look aside Hits, Paged Pool Lookaside Hits, Available Paged Pool Pages, Resident System Cache Page, Resident Paged Pool Page, Resident System Driver Page, Cache manager Fast Read with No Wait, Cache manager Fast Read with Wait, Cache manager Fast Read Resource Missed, Cache manager Fast Read Not Possible, Cache manager Fast Memory Descriptor List Read with No Wait, Cache manager Fast Memory Descriptor List Read with Wait, Cache manager Fast Memory Descriptor List Read Resource Missed, Cache manager Fast Memory Descriptor List Read Not Possible, Cache manager Map Data with No Wait, Cache manager Map

Data with Wait, Cache manager Map Data with No Wait Miss, Cache manager Map Data Wait Miss, Cache manager Pin-Mapped Data Count, Cache manager Pin-Read with No Wait, Cache manager Pin Read with Wait, Cache manager Pin-Read with No Wait Miss, Cache manager Pin-Read Wait Miss, Cache manager Copy-Read with No Wait, Cache manager Copy-Read with Wait, Cache manager Copy-Read with No Wait Miss, Cache manager Copy-Read with Wait Miss, Cache manager Memory Descriptor List Read with No Wait, Cache manager Memory Descriptor List Read with Wait, Cache manager Memory Descriptor List Read with No Wait Miss, Cache manager Memory Descriptor List Read with Wait Miss, Cache manager Read Ahead IOs, Cache manager Lazy-Write IOs, Cache manager Lazy-Write Pages, Cache manager Data Flushes, Cache manager Data Pages, Context Switches, First Level Translation buffer Fills, Second Level Translation buffer Fills, and System Calls

- The system exception information consisting of Alignment Fix up Count, Exception Dispatch Count, Floating Emulation Count, and Byte Word Emulation Count
- The system lookaside information consisting of Current Depth, Maximum Depth, Total Allocates, Allocate Misses, Total Frees, Free Misses, Type, Tag, and Size
- The system interrupt information consisting of context switches, deferred procedure call count, deferred procedure call rate, time increment, deferred procedure call bypass count, and asynchronous procedure call bypass count
- The system process information consisting of Next Entry Offset, Number Of Threads, Create Time, User Time, Kernel Time, Image Name, Base Priority, Unique Process ID, Inherited from Unique Process ID, Handle Count, Session ID, Page Directory Base, Peak Virtual Size, Virtual Size, Page Fault Count, Peak Working Set Size, Working Set Size, Quota Peak Paged Pool Usage, Quota Paged Pool Usage, Quota Peak Non Paged Pool Usage, Quota Non Paged Pool Usage, Page file Usage, Peak Page file Usage, Private Page Count, Read Operation Count, Write Operation Count, Other Operation Count, Read Transfer Count, Write Transfer Count, and Other Transfer Count

Data Encryption and Decryption

The following functions provide interfaces to the cryptomodule's data encryption and decryption functions.

```
FipsDes
VOID
FipsDes(
        UCHAR *         pbOut,
        UCHAR *         pbIn,
        void *          pKey,
        int             iOp
        );
```

The FipsDes function encrypts or decrypts the input buffer pbIn using DES, putting the result into the output buffer pbOut. The operation (encryption or decryption) is specified with the iOp parameter. The pKey is a DESTable struct pointer returned by the FipsDesKey function. FipsDes zeroises its copy of the DESTable struct before returning to the caller.

```
Fips3Des
VOID
Fips3Des(
        UCHAR *         pbIn,
        UCHAR *         pbOut,
        void *          pKey,
        int             op)
```

The Fips3Des function encrypts or decrypts the input buffer pbIn using Triple DES, putting the result into the output buffer pbOut. The operation (encryption or decryption) is specified with the op parameter. The pkey is a DES3Table struct returned by the Fips3Des3Key function. Fips3Des zeroises its copy of the DES3Table struct before returning to the caller.

```
FipsCBC
BOOL FipsCBC(
        ULONG           EncryptionType,
        BYTE   *        output,
        BYTE   *        input,
        void   *        keyTable,
        int             op,
        BYTE   *        feedback
    )
```

The FipsCBC function encrypts or decrypts the input buffer input using CBC mode, putting the result into the output buffer output. The encryption algorithm (DES or Triple DES) to be used is specified with the EncryptionType parameter. The operation (encryption or decryption) is specified with the op parameter.

If the EncryptionType parameter specifies Triple DES, the keyTable is a DES3Table struct returned by the Fips3Des3Key function. If the EncryptionType parameter specifies DES, the keyTable is a DESTable struct returned by the FipsDesKey function.

This function encrypts just one block at a time and assumes that the caller knows the algorithm block length and the buffers are of the correct length. Every time when the function is called, it zeroises its copy of the DES3Table or DESTable struct before returning to the caller.

```
FipsBlockCBC
BOOL FipsBlockCBC(
        ULONG           EncryptionType,
        BYTE   *        output,
        BYTE   *        input,
```

```
    ULONG       length,
    void  *     keyTable,
    int         op,
    BYTE  *     feedback
)
```

Same as FipsCBC, the FipsBlockCBC function encrypts or decrypts the input buffer input using CBC mode, putting the result into the output buffer output. The encryption algorithm (DES or Triple DES) to be used is specified with the EncryptionType parameter. The operation (encryption or decryption) is specified with the op parameter.

If the EncryptionType parameter specifies Triple DES, the keyTable is a DES3Table struct returned by the Fips3Des3Key function. If the EncryptionType parameter specifies DES, the keyTable is a DESTable struct returned by the FipsDesKey function.

This function can encrypt/decrypt more than one block at a time. The caller specifies the length in bytes of the input buffer in the "length" parameter. So the input/output buffer length is the arithmetic product of the number of blocks in the input/output buffer and the block length (8 bytes). When the length is 8 (i.e. one block of input buffer), FipsBlockCBC is the same as FipsCBC.

Every time when the function is called, it zeroises its copy of the DES3Table or DESTable struct before returning to the caller.

Hashing

The following functions provide interfaces to the cryptomodule's hashing functions.

```
FipsSHAInit
void
FipsSHAInit(
        A_SHA_CTX *     hash_context
        )
```

The FipsSHAInit function initiates the hashing of a stream of data. The output hash_context is used in subsequent hash functions.

```
FipsSHAUpdate
void FipsSHAUpdate(
        A_SHA_CTX *     hash_context,
        UCHAR *         pb,
        unsigned int    cb
        )
```

The FipsSHAUpdate function adds data pb of size cb to a specified hash object associated with the context hash_context. This function can be called multiple times to compute the hash on long data streams or discontinuous data streams. The FipsSHAFinal function must be called before retrieving the hash value.

```
FipsSHAFinal
void FipsSHAFinal (
        A_SHA_CTX *                             hash_context,
        unsigned char [A_SHA_DIGEST_LEN]        hash)
```

The FipsSHAFinal function computes the final hash of the data entered by the FipsSHAUpdate function. The hash is an array char of size A_SHA_DIGEST_LEN (20 bytes).

```
FipsHmacSHAInit
void
FipsSHAInit(
        A_SHA_CTX *     pShaCtx
        UCHAR *         pKey,
        unsigned int    cbKey
        )
```

The FipsHmacSHAInit function initiates the HMAC hashing of a stream of data, with an input key provided via the pKey parameter. The size of the input key is specified in the cbKey parameter. If the key size is greater than 64 bytes, the key is hashed to a new key of size 20 bytes using SHA-1. The input key is EOR'ed with the ipad as required in the HMAC FIPS. The output pShaCtx is used in subsequent HMAC hashing functions. Every time when the function is called, it zeroises its copy of the pKey before returning to the caller.

FipsHmacSHAUpdate

```
void FipsSHAUpdate(
        A_SHA_CTX *     pShaCtx,
        UCHAR *         pb,
        unsigned int    cb
        )
```

The FipsHmacSHAUpdate function adds data pb of size cb to a specified HMAC hashing object associated with the context pShaCtx. This function can be called multiple times to compute the HMAC hash on long data streams or discontinuous data streams. The FipsHmacSHAFinal function must be called before retrieving the final HMAC hash value.

FipsHmacSHAFinal

```
void FipsHmacSHAFinal (
        A_SHA_CTX *     pShaCtx,
        UCHAR *         pKey,
        unsigned int    cbKey,
        UCHAR *         hash)
```

The FipsHmacSHAFinal function computes the final HMAC hash of the data entered by the FipsHmacSHAUpdate function, with an input key provided via the pKey parameter. The size of the input key is specified in the cbKey parameter. If the key size is greater than 64 bytes, the key is hashed to a new key of size 20 bytes using SHA-1. The input key is EOR'ed with the opad as required in the HMAC FIPS. It is the caller's responsibility to make sure that the input key used in FipsHmacSHAFinal is the same as the input key used in FipsHmacSHAInit. The final HMAC hash is an array char of size A_SHA_DIGEST_LEN (20 bytes). Every time when the function is called, it zeroises its copy of the pKey before returning to the caller.

Acquiring a Table of Pointers to FipsXXX Functions

A kernel mode user of the FIPS.SYS driver must be able to reference the FipsXXX functions before using them. The user needs to acquire the table of pointers to the FipsXXX functions from the FIPS.SYS driver. The user accomplishes the table acquisition by building a Fips function table request irp (I/O request packet) and then sending the irp to the FIPS.SYS diver via the IoCallDriver function. Further information on irp and IoCallDriver can be found on Microsoft Windows XP Driver Development Kit.

CRYPTOGRAPHIC KEY MANAGEMENT

The FIPS.SYS cryptomodule manages keys in the following manner.

Key Material

FIPS.SYS use keys provided by the caller for the following algorithms: DES, 3DES and HMAC.

Key Generation

Random keys can be generated by calling the FipsGenRandom() function. Key are generated following the techniques in FIPS PUB 186-2, Appendix 3, Random Number Generation

Key Entry and Output

DES keys can be imported into FIPS.SYS via FipsDesKey(). DESTable struct can be exported out of FIPS.SYS via FipsDesKey(). DESTable struct can be imported into FIPS.SYS via FipsDes() or FipsCBC().

Triple DES keys can be imported into FIPS.SYS via Fips3Des3Key(). DES3Table struct can be exported out of FIPS.SYS via Fips3Des3Key(). DES3Table struct can be imported into FIPS.SYS via Fips3Des() or FipsCBC().

HMAC keys can be imported into FIPS.SYS via FipsHmacSHAInit and FipsHmacSHAFinal.

Key Storage

FIPS.SYS does not store keys. DES and Triple DES keys and their associated DESTable and DES3Table struct, and HMAC keys are zeroized after used.

Key Archival

FIPS.SYS does not archive cryptographic keys. All key copies inside FIPS.SYS are destroyed and their memory location zeroized after used. It is the caller's responsibility to maintain the security of DES, Triple DES and HMAC keys when the keys are outside FIPS.SYS.

Key Destruction

All DES and Triple DES key copies, their associated DESTable and DES3Table struct copies, and HMAC key copies inside FIPS.SYS are destroyed and their memory location zeroized after they have been used in FipsDes, Fips3Des, or FipsCBC.

SELF-TESTS

Power up

Software tests via a DES MAC of library image

- DES ECB encrypt/decrypt KAT
- 3DES ECB encrypt/decrypt KAT
- DES CBC encrypt/decrypt KAT
- 3DES CBC encrypt/decrypt KAT
- SHA-1 hash KAT
- SHA-1 HMAC hash KAT

MISCELLANEOUS

The following items address requirements not addressed above.

Cryptographic Bypass

Cryptographic bypass is not support in FIPS.SYS.

Operator Authentication

FIPS.SYS provides no authentication of operators. However, the Microsoft Windows XP operating system upon which it runs does provide authentication, but this is outside the scope of FIPS.SYS FIPS validation. The information about the authentication provided by Microsoft Windows XP is for informational purposes only. Microsoft Windows XP requires authentication from a trusted computer base (TCB) before a user is able to access system services. Once a user is authenticated from the TCB, a process is created bearing the Authenticated User's security token. All subsequent processes and threads created by that Authenticated User are implicitly assigned the parent's (thus the Authenticated User's) security token. Every user that has been authenticated by Microsoft Windows XP is naturally assigned the Authenticated User role when he/she accesses FIPS.SYS.

Operating System Security

The FIPS.SYS cryptomodule is intended to run on Windows XP in the Single User Mode.

When the Windows XP operating system Loader loads the cryptomodule into memory, the cryptomodule runs a DES MAC on the cryptomodule's disk image of FIPS.SYS, excluding the DES MAC, checksum, and export signature resources. This MAC is compared to the value stored in the DES MAC resource. Initialization will only succeed if the two values are equal.

FOR MORE INFORMATION

For the latest information on Windows XP, check out our World Wide Web site at http://www.microsoft.com/windows.

Microsoft

Operating System

Microsoft Kernel Mode Cryptographic Module

FIPS 140-1 Documentation: Finite State Machine

7/2/2002 2:34:35 PM~~6/28/2002 3:03:16 PM~~

Abstract

This document specifies the finite state machine for the Kernel Mode Cryptographic Module (FIPS.SYS) as described in FIPS PUB 140-1.

CONTENTS

INTRODUCTION

Microsoft Kernel Mode Cryptographic Module (FIPS.SYS) is a FIPS 140-1 Level 1 compliant, general-purpose, software-based, cryptographic module residing at the Kernel Mode level of the Windows Operating System. It runs as a kernel mode export driver (a kernel-mode DLL) and encapsulates several different cryptographic algorithms in an easy-to-use cryptographic module accessible by other kernel mode drivers. It can be linked into other kernel mode services to permit the use of FIPS 140-1 Level 1 compliant cryptography.

FINITE STATE MACHINE

The FIPS.SYS cryptomodule can be in exactly one of the following states at any given moment. Transitions between states can be automatic or result from user intervention.

States

See Appendix A and B for more information.

Power Up

The Power Up state is entered when Windows XP Loader calls the FIPS.SYS driver entry point function DriverEntry() during system boot.

Power Down

The Power Down state is entered when Windows XP Kernel calls the FIPS.SYS driver's unload function which was set in DriverUnload field of the DriverObject representing FIPS.SYS during the Power Up state.

Init Error

The Init Error State is entered when FIPS.SYS's DriverEntry() fails as a result of either configuration errors (i.e. not enough memory, etc.) or errors resulting from the power up self-tests.

Initialized

The Initialized state is entered when FIPS.SYS's DriverEntry() returns successfully and the Windows Loader completes the loading of FIPS.SYS.

Key Initialized

The Key Initialized state is entered after keys are formatted into a DESTable or DES3Table struct with FipsDesKey(), Fips3Des3Key() .

Operation Error

The Operation Error state is entered whenever an error occurs as a result of a cryptographic operation. FIPS.SYS will automatically transition back to either the Initialized or Key Initialized state depending on whether or not keys have been successfully formatted into a DESTable or DES3Table struct.

State Transitions

See Appendix A.

State Diagrams

See Appendix B.

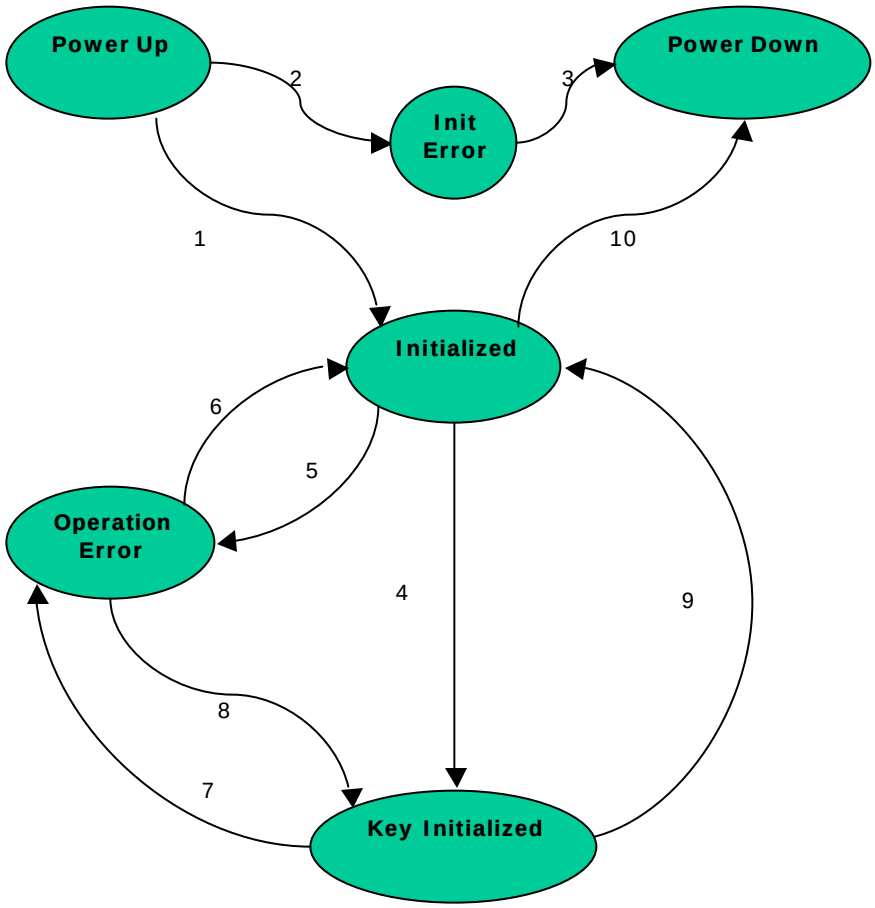
APPENDIX A

The following table describes the state transitions possible within the FIPS.SYS cryptomodule during operation.

	Current State	Input	Output	Next State
1	Power Up	FIPS.SYS loads	NO_ERROR	Initialized
2	Power Up	FIPS.SYS not found	STATUS_UNSUCCESSFUL	Init Error
2	Power Up	DES MAC check on cryptographic provider fails	STATUS_UNSUCCESSFUL	Init Error
2	Power Up	One or more power-on cryptographic self-tests fail	STATUS_UNSUCCESSFUL	Init Error
2	Power Up	System error	STATUS_UNSUCCESSFUL	Init Error
3	Init Error	Automatic transition	No output	Power Down
4	Initialized	Key formatting operation (i.e. FipsDesKey(), Fips3Des3Key()) requested	No output	Key Initialized
5	Initialized	Key formatting operation failure	Operation specific error message	Operation Error
6	Operation Error	Automatic transition when keys have not yet been initialized	No output	Initialized
7	Key Initialized	Generic cryptographic operation failure	Operation specific error message	Operation Error
8	Operation Error	Automatic transition when keys have already been initialized	No output	Key Initialized
9	Key Initialized	Generic cryptographic operation (i.e. FipsDes(), Fips3Des(), or FipsCBC ()) completed	NO_ERROR	Initialized
10	Initialized	Automatic transition when Windows XP Kernel calls the FIPS.SYS driver's unload function	NO_ERROR	Power Down

APPENDIX B

The following diagram illustrates the finite state machine of the FIPS.SYS cryptomodule.



FOR MORE INFORMATION

For the latest information on Windows XP, check out our World Wide Web site at http://www.microsoft.com/windows.

Microsoft

Microsoft Windows xp

Operating System

Microsoft Kernel Mode Cryptographic Module

FIPS 140-1 Documentation: Master Component List

7/2/2002 2:34:35 PM~~6/28/2002 3:03:16 PM~~

Abstract

This document specifies the master component list for the Kernel Mode Cryptographic Module (FIPS.SYS) as described in FIPS PUB 140-1.

CONTENTS

MASTER COMPONENT LIST

The FIPS.SYS cryptomodule is a software cryptomodule and is intended to operate on a PC running Windows XP. Several components of the base PC are also to be considered components of the cryptomodule.

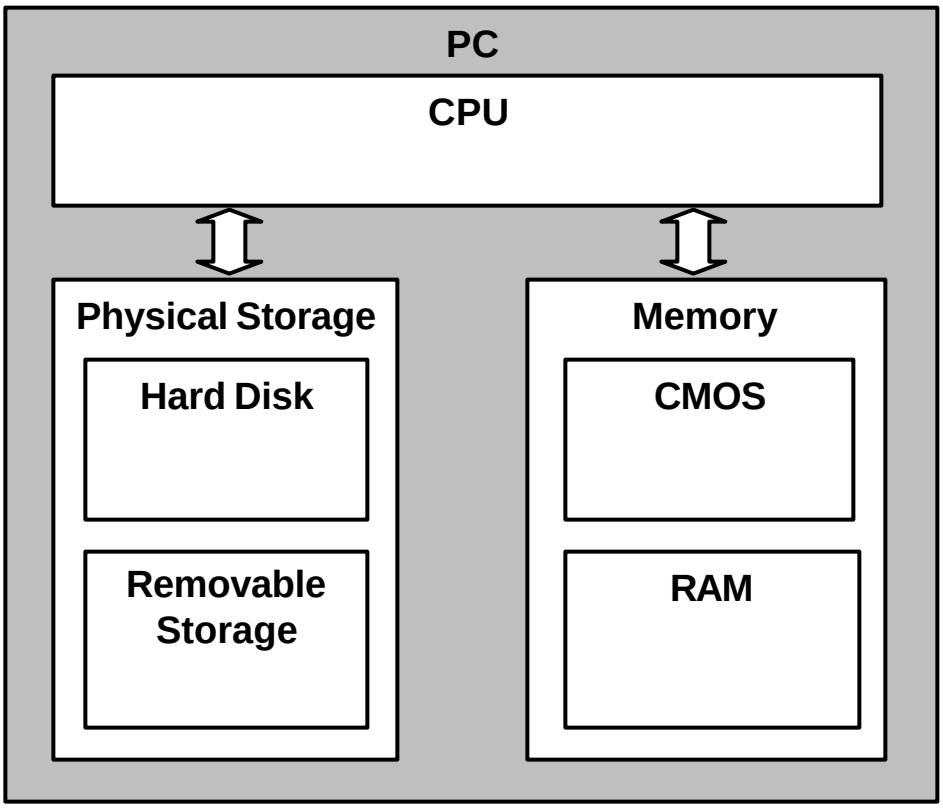
Components

The following components are to be considered components of the cryptomodule (see Appendix A below):

- PC Enclosure
- Central Processing Unit (CPU)
- Physical Storage (Hard Drives and Removable Storage)
- Memory (RAM and CMOS)

APPENDIX A

The following diagram illustrates the master components of the FIPS.SYS cryptomodule.



FOR MORE INFORMATION

For the latest information on Windows XP, check out our World Wide Web site at http://www.microsoft.com/windows.