SecureDoc® Disk Encryption
Cryptographic Engine for MacOS X

FIPS 140-2 Non-Proprietary Security Policy

Abstract:	This document specifies Security Policy enforced by SecureDoc® Cryptographic Engine compliant with the requirements of FIPS 140-2 level 1. The policy specifies security rules under which SecureDoc® Cryptographic Engine operates.

Module Version:	7.2
Revision:	1.3
Revision date:	May 22, 2018
Evaluation:	FIPS 140-2 Level 1

Table of Contents

1 Introduction

1.1 Purpose

This document describes the non-proprietary FIPS 140-2 security policy for the SecureDoc® Cryptographic Engine (hence the Module) used in SecureDoc® data encryption solutions.

It describes the various services offered by the Module and the mechanisms provided to ensure that these services meet the FIPS 140-2 Level 1 requirements. It also describes the storage of cryptographic data within the engine and how the Module is protected against tampering and data loss. The management of various roles and restrictions that can be applied to the user in using these services and data are documented.

This document has been prepared in accordance with the requirements of FIPS 140-2 and is not to be seen as a complete description of the product capabilities or applications. Please contact WinMagic at http://www.winmagic.com for further information.

The target levels of validation by components are specified below.

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

2 Product Overview

2.1 Module Installation

The software comprising the Module is provided as an installation package in format of MacOS X. The installation procedure includes several stages performed by a crypto-officer:

1. On initial stage the crypto-officer goes through general steps like choosing installation volume and folder, confirming end user agreements, etc. Once the package is installed the computer may reboot to activate SecureDoc kernel driver.

2. On the next stage the operator works with the installation wizard that does the following:

    a. Generates cryptographic keys

    b. Creates a key file for authentication in crypto-officer role.

    c. Installs the Boot Logon component for pre-boot authentication.

    d. Encrypts the hard disk (optional).

    e. Prepares Data Recovery Media (optional).

    f. Creates key files for authentication in user role.

The installation procedure is described in details in the "*SecureDoc for Mac Standalone Version 5.3 User Manual*" available for download from the vendor's website for the registered customers.

2.2 Cryptographic Engine

The Module provides cryptographic and key management services for all SecureDoc® products.

The Module is based on the widely adopted PKCS-11 Cryptoki standard as an API available for the WinMagic Applications.

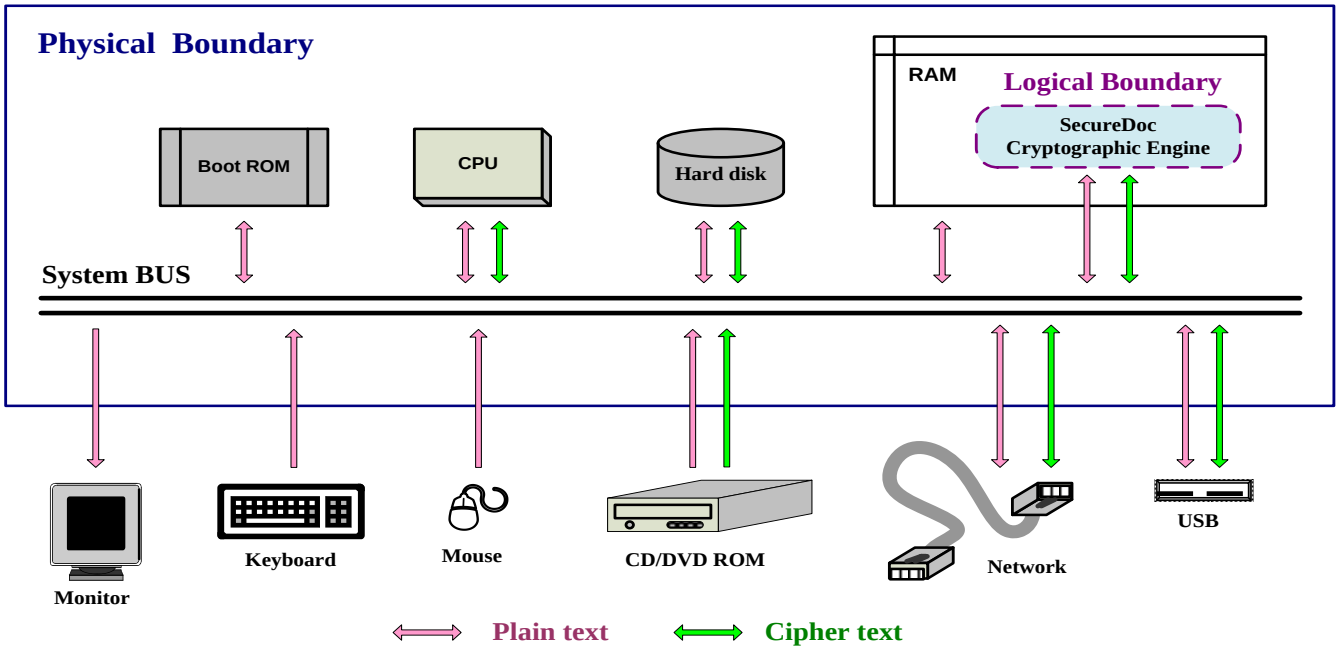
3 Cryptographic Module Definition

3.1 Cryptographic Module Boundary and Interface

From the point of view of FIPS 140-2, the cryptographic boundary of the Module as a multiple-chip standalone module includes the Module itself, the Operating System (OS) and the General Purpose Computer (GPC) hardware.

The interface to the Module is the physical interface to the GPC including the mouse, keyboard, video monitor, etc. as defined in the block-diagram below. The hardware external to the physical boundary is connected either via a designated port or via USB if it is supported by the vendor of this hardware and the OS.

Figure 1 - Cryptographic Module Block-Diagram



The correspondence between logical interfaces and physical ports of the Module is provided in the table below. Different interfaces are kept logically separated while using the same physical ports through utilizing different sets of input/output commands sent to a physical port by each of the interfaces that share the port or by invoking different API calls for different interfaces.

Table 1 - Ports and logical interfaces correspondence

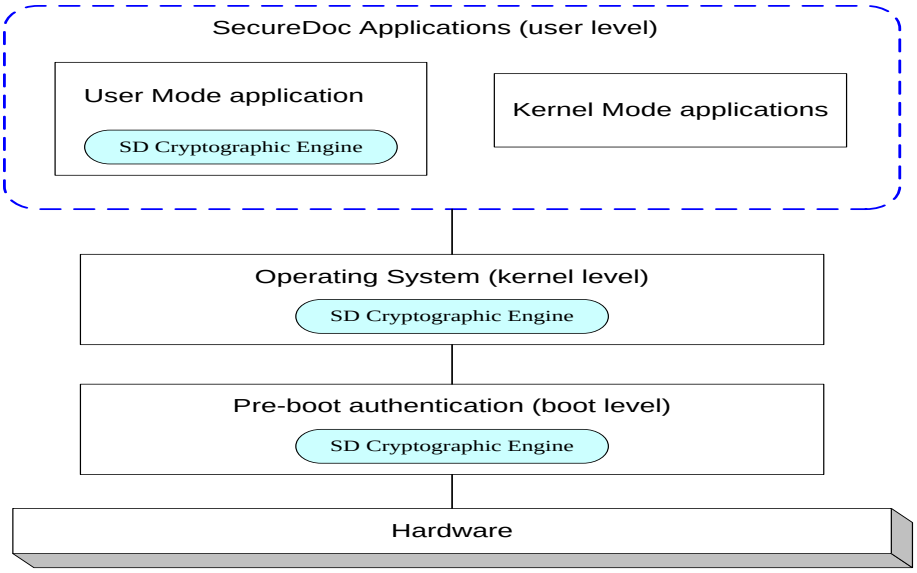
Logical Interface	Interface purpose	Physical Port External Devices
Data input	Enter the data to be processed by the Module	Keyboard port, mouse port, USB, Ethernet port
Data output	Output the data being processed by the Module	USB port, Ethernet port
Control input	Enter data used to control operation of the Module	Mouse, keyboard, and CPU
Status output	Show module status and error messages	Video monitor
Power	Provides power for module operations	110/220 VAC power interface

3.2 Module Operational Levels

The software comprising The Module operates at three different levels as shown in the picture below. There is a component of the Module designated for each operational level. All these components taken together define the Module logical boundary.

Once the Module goes through Power On at hardware level the Module performs authentication of the operator at the boot level. Successful authentication results in initiation the procedure of loading operating system to go to the kernel level. Once operating system is loaded the Module is active and controls critical operations as a kernel filter driver. On the next step the operator has to login to OS session to enter the user level in which SecureDoc (and other) applications work. At this level all media access operations as well as SecureDoc configuration management go through kernel filter driver. Other SecureDoc application may run the Module in user mode to perform cryptographic operations in memory.

Figure 2 - The Module operational levels



3.3 Implementation

The Module is designed to integrate closely with the operating system taking full advantage of the OS security. It is available in various instances to run either in user mode, for general purpose applications; in kernel mode for use by low-level applications such as the SecureDoc® Disk Encryptor; or in real mode at pre-boot.

The user mode instance is distributed as a shared library. The kernel mode instance is installed as an OS driver. The real mode instance is a binary file loaded by the boot code.

From the programmatic point of view, the application interface to the Module is using the standard C-language API. This interface corresponds to the PKCS#11 Cryptoki standard with extensions introduced to meet the unique requirements of the SecureDoc product and FIPS 140-2.

The Module opens PCKS#11 sessions for each service called by operator to keep cryptographic context used by the service separated from other services invoked by the operator.

The Module design is such that only one function call can be processed at any time in a given application thread. Any other function requests will not be executed until the current function processing is completed.

The API ensures that each service call results in a return code that unambiguously reflects the success or failure of the request based on the current state of the Module. The return value is always either **CKR_OK** indicating success or a specific error code value.

The Module utilizes SecureDoc Key Files which exist as PKCS#11 nested objects storing individual operator's attributes and serves for the purposes of operator authentication, authorization and key management.

The public data in the key file are obfuscated against casual browsing by encryption with a proprietary key known to the Module. The sensitive data are stored in the private part of the key file protected as specified by NIST SP 800-132.

3.4 Operational Environment

For the purpose of FIPS 140-2 level 1 evaluation, the SecureDoc® Cryptographic Engine is categorized as a multiple-chip standalone module. The module has been tested on:

- MacOS X 10.7 Lion 32-bit running on a MacBook Pro Intel® Core i7
- MacOS X 10.7 Lion 64-bit running on a MacBook Pro Intel® Core i7

Other supported but not tested operational environments are:

- Any 32 or 64-bit release of MacOS X starting from 10.5.2 Leopard
- Any Intel CPU supported on Apple™ hardware
- Any GPC by Apple™ capable of running the specified OSes

The claim for FIPS 140-2 compliance for these OE's is vendor affirmed based on the Implementation Guidance, G5 "Maintaining validation compliance of software or firmware cryptographic modules", clause 1.a).i).

3.5 Physical Security

The Module is implemented as a software component and thus the FIPS 140-2 physical security requirements are not applicable.

3.6 Mitigation of Other Attacks

The Module is not designed to mitigate any known specific attacks.

3.7 FIPS Approved Mode of Operation

The Module supports Approved and non-Approved mode. In the Approved mode the Module employs FIPS-Approved and FIPS-allowed algorithms listed in Table 2.

In the operational state the Module alternates service by service between Approved and non-Approved modes of operation. The lists of services provided in Approved and not-Approved modes are located in section 6.

To protect CSP when the Module swaps modes the operator shall re-initialize the Module so that cryptographic keys and other CSP were destroyed in the volatile memory thus preventing sharing between different modes.

Table 2 - Algorithms in Approved Mode of Operation

Algorithm	Cryptographic Function	Modes / Mechanisms	Key Size (bits)	Certificate #
AES	Encipherment	ECB, CBC	256	3949
AES	Encipherment	CBC	256	3948
SHA	Hashing	SHA-1,256, 384, 512		3257
HMAC	Message authentication	SHA-1, 256, 384, 512	256	2572
DRBG	Random bit generation	SP800-90A HMAC_DRBG	256	1152
KTS	Key wrapping	AES and HMAC	256	3948, 3949, 2572
AES	Key unwrapping	CBC	256	3948, 3949

3.8 Self-Tests

A complete set of self-checks is run before the Module services may be accessed. If an error occurs during a self-check or a fatal error occurs during the subsequent execution of any of the services, the Module enters in error condition mode and must be re-initialized before it can be used again. There is no data output for the application thread while self-checks are being executed or when it is in error mode.

Table 3 provides details of self-tests performed by the Module:

Table 3 - Self-tests performed by the Module

Test	Actions performed
Cryptographic Algorithms Test	Performed automatically when the Module is initialized and on demand via the self-test service. Executes Known Answer Tests for all employed algorithms (AES, HMAC_DRBG, SHA-1, SHA-256, SHA-384, SHA-512, HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512)
Software Integrity Test	Performed automatically when the Module is initialized. Checks that the Module has not been compromised by verifying an HMAC-SHA-256 message authentication code.
DRBG Health Test	Performed automatically when the Module is initialized. The Instantiate and Generate Random functions are verified for correct operations via Known Answer Tests with fixed values of the accepted parameters.
DRBG Continuous Test	Performed each time the DRBG is used to generate random bits. The output of the DRBG is compared with the previous block of the generated data. If two blocks are identical a catastrophic error is generated.

4 Cryptographic Key Management

4.1 Key Generation

The Module supports generation of two types of keys: session (temporary) and persistent. Session keys are destroyed in the volatile memory when the PCKS#11 sessions opened by a service are not used anymore.

Lifetime of persistent keys in RAM is extended beyond the time span when a particular service uses them. These keys are destroyed once the operator has logged off. Persistent keys may be stored in the key file. To create a persistent key the operator must have "Modify Key" privilege.

Keys are generated by the DRBG according to specification given in 3.7. The DRBG requests 512 bits of entropy from the entropy source in the GET function call during instantiation.

4.2 Key Zeroization

Any key created by the Module exist in memory as a single-value PKCS#11 cryptographic object or as a part of larger objects like key store.

The object containing the key is destroyed when a PCKS#11 session opened to perform services with this key is closed. When operator logs off the Module the keys copied from the key file provided to the Module into RAM are destroyed as well.

All keys and other CSP located in the volatile memory are zeroized when the Module is reset or unloaded.

When a cryptographic object is destroyed the memory occupied by the object is overwritten or cleaned up with zeros.

Both crypto-officer and user can zeroize the Module. It is achieved via uninstallation of the software from the computer during which the disk sectors containing operators' key files and headers of the encrypted areas are erased using Peter Gutmann's method.

4.3 Key Distribution

Cryptographic keys placed in key file may be extracted for distribution. The Module can protect a key being distributed with one the following symmetric key methods:

- KTS (AES and HMAC) in Approved mode
- AES encryption (wrapping) in non-Approved mode
- AES decryption to unwrap keys encrypted in non-Approved mode (allowed for Approved mode)

The module utilizes 256-bit keys for key wrapping thus providing encryption strength of 256 bits.

4.4 Key Entering

The Module allows entering key values electronically for a temporary purpose (like testing, etc.) as session keys using the Enter Key service. The key values are electronically entered as plaintext binary data received from the applications resident on the host platform. The Module does not support key entry across the host platform's physical boundary.

The session keys exist in RAM only and are never stored. They are destroyed after PCKS#11 session is closed as described in 4.5 "Key Zeroization".

5 Roles, Authentication and Services

5.1 Roles

The Module controls access to the services through the Authorization Vector (AV) associated with the operator when they authenticate to the Module. Operator's role is defined as a subset of privileges (rights) set in the AV and enforced by the Service Access Model. Those privileges are: "Modify Key", "Export Key" and "Delete Key" when applied to persistent keys. The privileges are mapped to the appropriate services via a bit string where every service working with keys corresponds to a single-bit flag.

Based on this subset of the privileges the Module maintains Crypto-Officer Role and User Role. No other roles are available. The AV assigned to an operator is stored protected as a CSP in operator's key file and becomes available only after successful authentication.

5.2 Operator Authentication

Operator authenticates to the Module by logging in to the key file with their password. The password is obscured via replacement with asterisks ('*') while being manually entered by the operator.

Authentication mechanism is designed in compliance with Option 2a) specified in NIST SP800-132. The password entered by the operator is used as input to PBKDF2 specified by SP800-132.

The derived 256-bit key is then used to decrypt and authenticate the key protecting data in the key file. Authenticated encryption required for Option 2a) is implemented as AES+HMAC combination. Failure of MAC verification is treated as unsuccessful login attempt. No other feedback but the result of authentication (success of failure) is provided to operator.

If authentication is successful the operator is assigned a PKCS#11 slot that contains AV of the operator within the assumed role. The slot also contains other attributes associated with the operator and contexts of the PKCS#11 sessions used to call the services.

When the operator logs off the slot destroyed with all CSP stored and PCKS#11 sessions opened on it. The operator has to re-authenticate to access the Module services again.

5.3 Strength of Authentication Mechanism

Operator password is generated outside of the Module and must follow certain constraints. The password policy imposed by SecureDoc requires at least **8** (eight) alpha/digit/special characters with at least one of each kind as well as both upper and lower case letters present.

Strength of the authentication mechanism employed is calculated as follows:

- The number of all passwords matching the constraints specified above is 6,095,689,385,410,816. Following the "Law of Averages" the number of attempts to maintain a brute force attack (3,047,844,692,705,408) is big enough to guarantee the required 1 in 1,000,000 probability of guessing the password.

- The Module also blocks the input interface after 5 sequentially failed login attempts forcing the operator to reset the Module. In case of a GPC it means full reboot that allows in average 10-12 login attempts per minute. For the password length and complexity specified above, this limitation gives a possibility less than 1 in 100,000 to guess the password within one minute.

5.4 Services

Table 5 lists the services supported by the Module in FIPS-Approved mode.

The module provides no services while in non-initialized state.

In initialized state availability of the requested service is decided based on the AV associated with the operator in the assumed role. The Module verifies whether the proper privileges are set in the AV and then either grants or denies access to the service.

When the Module operates in the Approved mode only authenticated operator can request the services with the exception of Module Initialization, Module Zeroization and Show Status.

Table 5 - Services available in FIPS-Approved mode

Service	Description	Roles	
		CO	U
Module Initialization	Executes the Power-Up test when the Module is being powered on that includes integrity test and KAT of cryptographic algorithms	✓	✓
Module reset	Moves the Module to non-initialized state. Destroys all CSP in the volatile memory.	✓	✓
Module Zeroization	Destroys proprietary keys via uninstallation of the entire module from the host system	✓	✓
On Demand Self-Test	Runs integrity test and KAT of cryptographic algorithms by request of operator	✓	✓
Show Status	Indicates status of the Module, shows error codes produced by the Module	✓	✓
Setup credentials	Sets or changes operator's password.	✓	✓
Create Key	Creates a persistent key	✓	–
Delete Key	Deletes a persistent key and zeroizes the memory	✓	–
Key Transport	Wraps/Unwraps keys using AES+HMAC combination	✓	–
Key Unwrapping	Unwraps keys using AES CBC decryption (allowed method)	✓	–
Enter Key	Enters a key value for a session key	✓	✓
Generate Key	Generates a session key	✓	✓
Encrypt	Encrypts data using AES algorithm	✓	✓
Decrypt	Decrypts data using AES algorithm	✓	✓
Message Digest	Calculates hash of data using SHA-1 or SHA-256,384,512 algorithm	✓	✓
Message Authentication	Creates /Verifies Message Authentication Code using HMAC based on one of the supported SHA algorithms	✓	✓

Table 6 lists the services supported by the Module in non-Approved mode.

Table 6 - Services available in non-Approved mode

Service	Description	Roles	
		CO	U
Key Wrapping (non-compliant)	Wraps keys using AES CBC encryption	✓	✓

Table 7 defines access of the services to key material and CSP (operations: **W** – write, **R** – read, **Z**– zeroize).

Services running in non-Approved mode do not access the CSP created in FIPS-Approved mode.

The DRBG Entropy Input String and Seed could not be accessed via the supported services and therefore are not listed in the table as CSP.

Table 7 – CSP access rights within services

Service	Key Material and CSP					
	Persistent Keys	Session Keys	Proprietary Keys	DRBG Internal state (V and Key)	Operator credential	Privileges
Module initialization	–	–	**R**	–	–	–
Module reset	**Z**	**Z**	–	**Z**	–	**Z**
Zeroize Module	–	–	**Z**	–	–	–
On Demand Self-Test	–	–	**R**	–	–	–
Show Status	–	–	–	–	–	–
Setup operator credentials	–	–	–	–	**W**	**W**
Create Key	**W**	–	–	**W,R**	–	**R**
Delete Key	**Z**	–	–	–	–	**R**
Key Transport	**W,R**	**W,R**	–	–	–	**R**
Key Unwrapping	**W**	**W**	–	–	–	–
Enter Key	**W**	**W**	–	–	–	–
Generate Key	**W**	**W**	–	**W,R**	–	–
Encrypt	**R**	**R**	–	–	–	–
Decrypt	**R**	**R**	–	–	–	–
Message Authentication	**R**	**R**	–	–	–	–