# 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. It specifies security rules under which the cryptographic module operates. |

| | |
|---|---|
| Module Version: | 6.1 |
| Document Version: | 1.10 |
| Revision date: | January 31, 2013 |
| 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 used in all SecureDoc® cryptographic products.

It describes the various services offered by the Cryptographic 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 Cryptographic 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 cryptographic 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.

    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 SecureDoc Cryptographic Engine provides cryptographic and key management services for all SecureDoc® products.

The SecureDoc Cryptographic Engine 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 cryptographic 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.

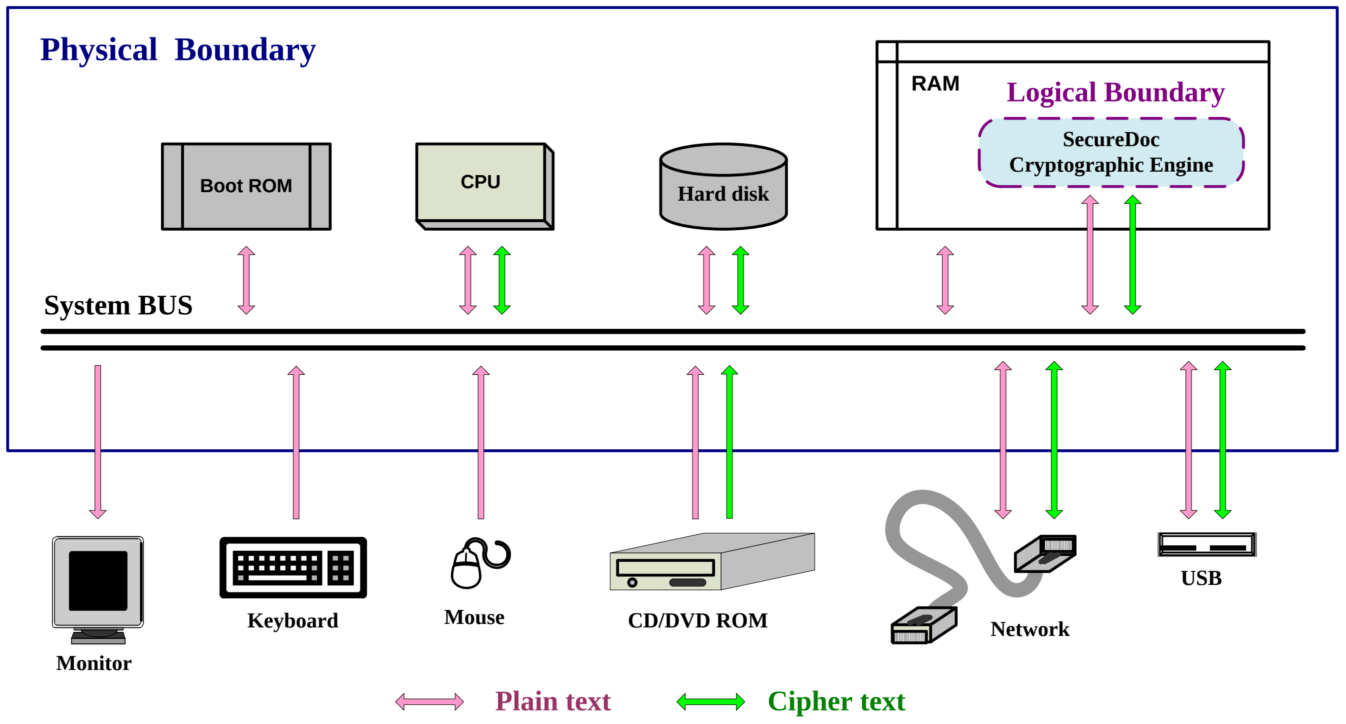

**Figure 1 - Cryptographic Module Block-Diagram**

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.

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

**Table 1 - Ports and logical interfaces correspondence**

## 3.2 Module Operational Levels

The software comprising cryptographic 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 cryptographic engine 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 cryptographic engine in user mode to perform cryptographic operations in memory.

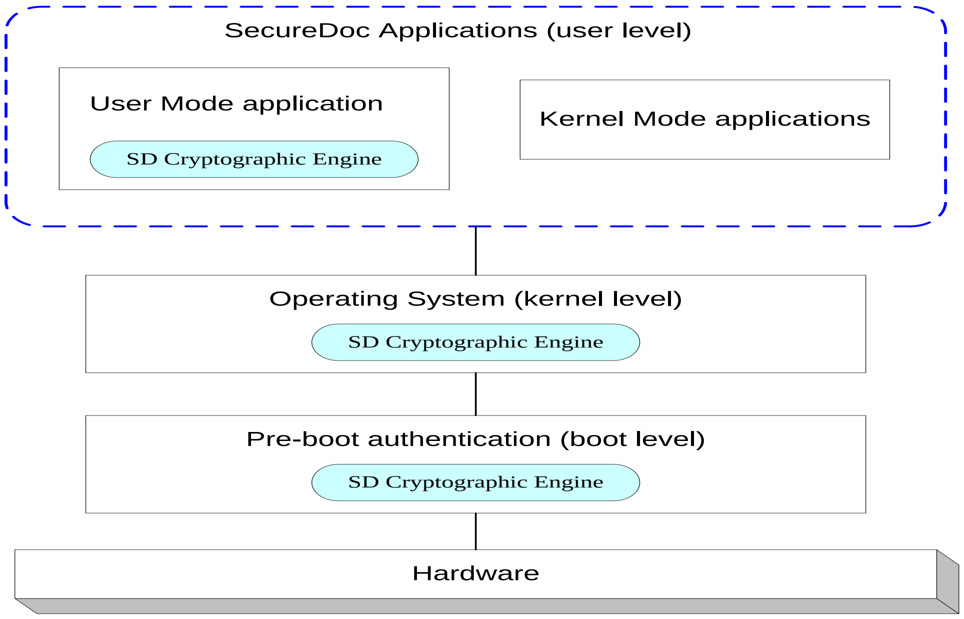

**Figure 2 - Cryptographic module operational levels**

## 3.3 Implementation

SecureDoc Cryptographic Engine 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 dynamic or 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 Cryptographic 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 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.

## 3.4 Operational Environment

For the purpose of FIPS 140-2 level 1 evaluation, the SecureDoc® Cryptographic Module is classified 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™ 2 Duo
- MacOS X 10.7 Lion 64-bit running on a MacBook Pro Intel® Core™ 2 Duo

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

SecureDoc Cryptographic Engine 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 SecureDoc Cryptographic Engine is not designed to mitigate any known specific attacks on the Cryptographic Module.

## 3.7 FIPS Approved Mode of Operation

The SecureDoc Cryptographic Engine implements only FIPS-Approved security functions based on Approved cryptographic algorithms listed in the table below. The implementation of these algorithms is validated by NIST CAVP.

| Algorithm | Cryptographic Function | Modes / Mechanisms | Output Block (bytes) | Key Size (bits) | Certificate # |
|---|---|---|---|---|---|
| AES | Encipherment | ECB, CBC | 16 | 256 | 1924, 1925 |
| SHA | Hashing | SHA-1,256, 384, 512 | 20, 32, 48, 64 | | 1690 |
| HMAC | Message authentication | SHA-1, 256, 384, 512 | 20, 32, 48, 64 | 256 | 1159 |
| PRNG | Random number generation | ANSI X9.31 AES | 16 | 256 | 1012 |

**Table 2 - Algorithms in Approved Mode of Operation**

Once properly installed the cryptographic module always operates in Approved mode of operation. The "*SecureDoc Standalone for Mac Version 5.3 User Manual*" instructs the crypto-officer how to configure the module for running in FIPS Approved mode.

An operator can obtain current mode of operation via the status interface that shows the version of the running SecureDoc Cryptographic Engine. The version must match the FIPS validated version listed on the CMVP website.

## 3.8 Self-Tests

A complete set of self-checks is run before the Cryptographic 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.

The table below details self-tests performed by the cryptographic module:

| Test | Type | Actions performed |
|---|---|---|
| Cryptographic Algorithms Test | Power-Up | Performed automatically when the module is initialized and on demand via the self-test service. Executes Known Answer Tests for all employed algorithms (AES, PRNG, SHA-1, SHA-256, SHA-384, SHA-512, HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512) |
| Software Integrity Test | Power-Up | Performed automatically when the module is initialized. Checks that the cryptographic module has not been compromised by verifying an HMAC-SHA-256 message authentication code. |
| PRNG Continuous Test | Conditional | Performed each time the pseudo random number generator is used PRNG output is compared with the previous block of generated data |

**Table 3 - Self-tests performed by the module**

# 4 Cryptographic Key Management

## 4.1 Encryption Keys

To encrypt user objects (hard drive, floppies, USB media, files and folders) SecureDoc Cryptographic Engine randomly generates Data Encrypting Keys (DEK) and Key Encrypting Keys (KEK).

Outside the cryptographic boundary these keys are stored as PKCS-11 cryptographic objects. Each object contains the following data:

| | |
|---|---|
| • KEK ID | – identifies the key used to wrap the key ( for DEK only) |
| • Key ID | – name of the key (for KEK only) |
| • Key value | – actual encryption key (wrapped in case of DEK) |
| • Algorithm Info | – algorithm type (AES), base IV, etc. |

Objects are obfuscated to protect information from casual browsing.

Data Encrypting Keys are stored with the encrypted data as part of the encryption header been wrapped with KEK.

Key Encrypting Keys are stored in the private part of User Key File (secure user data container). This part of key file is encrypted with key that becomes available upon successful user authentication.

## 4.2 User Keys

User Keys are used by cryptographic engine as wrapping keys for DEKs that encrypt user data on hard drive or removable media.

When the User Key File is created, it does not contain any keys. Normally the application will immediately create and install the special secret key object. This key may be used by any application for functions associated with this user (e.g. file encryption etc.) and is also used for recovery purpose.

Providing the key file has the necessary privileges, additional keys may also be generated or transferred into it from other sources. For example, a common key generated on one key file can be transferred into other key files to allow the users sharing the encrypted data.

## 4.3 Key File Protection

The public data in the key file are encrypted for obfuscation against the casual browsing by a proprietary key known to the SecureDoc Cryptographic Engine.

The sensitive data are stored in the private part of the key file protected as specified by NIST SP 800-132 Option 2a.

The private part of the key file is encrypted with a 256-bit Key File Key (KFK) randomly generated when the key file is created. The KFK is stored encrypted in the public part of the key file. For encryption of the KFK a 256-bit key is derived from the password provided by the operator for authentication.

## 4.4 Key Generation

The cryptographic engine supports generation of two types of keys: temporary keys and persistent keys. Temporary keys may be generated by any user logged into a key file. Temporary keys are destroyed when the current session ends unless the values have been backed-up somewhere else external to the cryptographic engine.

Persistent keys are stored in the key file. To create a persistent key the key file must have "Modify Key" privilege.

DEK and KEK are generated by the PRNG according to specification given in 3.7.

## 4.5 Key Zeroization

Any key created by the cryptographic engine 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 an internal cryptographic session opened to perform services with this key is closed. When operator terminates their authenticated session and logs out of the module all keys and other CSP located in the key file provide to the module are destroyed as well.

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.

## 4.6 Key Distribution

Cryptographic keys created with the "Extractable" attribute may be extracted from the key file for distribution via the Export Key service. Typically these are the user keys. Such keys may be exported from or imported into key file as a wrapped key object protected with the key file secret key.

The module utilizes AES algorithm for wrapping of cryptographic keys to be transported outside of the module boundary. As 256-bit AES keys are used for wrapping of the transported keys the resulting encryption strength is 256 bits.

A user key being distributed is backed up either as part of a master key file or similar secure container for transferring between key files and archives.

## 4.7 Entering Key Value

The cryptographic engine allows entering key values electronically for 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 cryptographic module does not support key entry across the host platform's physical boundary.

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

# 5 Operator Roles

## 5.1 Privileges

SecureDoc controls access to the services in the module 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.

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

When the operator logs in to the key file they are assigned a session in SecureDoc Cryptographic Module. Sessions are assigned to operators individually. The opened session is associated with the attributes of the key file like keys, AV, etc. The services utilized by the operator refer to this session. When the operator logs out, the session is destroyed and operator has to re-authenticate to access the module services again.

## 5.3 Strength of Authentication Mechanism

Operator password is generated outside of the SecureDoc Cryptographic 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. Even been reduced due to 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 length of password specified above, this limitation gives a possibility less than 1 to 100,000 to guess the password within one minute.

# 6 Cryptographic Module Services

## 6.1 Services implemented

The table below contains a full list of services implemented in the SecureDoc Cryptographic Engine. When the module operates in the Approved mode the services can only be performed by an authenticated role.

| Service | Category | Actions performed |
|---|---|---|
| Run Self-Test (Power On) | Administrative | Executes the power-up self-tests when the module is being powered on |
| On Demand Self-Test | Administrative | Explicitly executes the self-tests (requires “Perform Self Test” privilege) |
| Show Module Status | Administrative | Indicates status of the module (disk encryption, etc.), shows error codes produced |
| Change password | Administrative | Updates operator’s password |
| Zeroize Key | Administrative | Destroys stored keys via zeroization of the entire module |
| Create Key | Key Management | Creates a new persistent key |
| Delete Key | Key Management | Deletes a persistent key and zeroizes the memory |
| Export Key | Key Management | Exports a persistent key as a wrapped object |
| Enter Key | Key Management | Enters a key value for a session key |
| Generate Key | Key Management | Generates a session key |
| Encrypt | Cryptography | Encrypts data using AES algorithm |
| Decrypt | Cryptography | Decrypts data using AES algorithm |
| Digest | Cryptography | Calculates hash of a block of data using SHA-1 or SHA-256,384,512 algorithm |
| Message Authentication | Cryptography | Computes a message authentication code using HMAC-SHA-1 or HMAC-SHA-256,384,512 algorithm |

**Table 5 - Services provided by the module**

## 6.2 Administrative Services

Before any of the cryptographic services are accessed by an application, the module must be initialized. This is enforced by the module as part of the Power Up routine and include also self-tests and integrity test. No service could be requested when the module is in the non-initialized state.

The operator may also run Power-Up tests on demand at any time to validate the module status. Status of the module is indicated to the operator via an icon on the system tray or as pop-up messages.

Operator may change their own password once they are successfully authenticated.

Operator zeroizes the module and destroys all cryptographic keys stored on the hard drive by uninstalling the software from the computer.

## 6.3 Key Management Services

When a new key file is created, the Cryptographic officer will commonly generate the key file secret key any other keys required by the user and either back them wrapped with one of his keys or transfer them to his key file for escrow purposes. He may then transfer one or more enterprise keys from his key file to the new one so the new user may access them. After setting the necessary attributes for the keys, he will revoke all privileges for the new key file except “Use Keys” and “Modify password”. The key file may then be copied to the new user’s PC.

The values of the user keys in the key file can be accessed either via a handle or by wrapping with another key. Using these techniques the keys can be backed up or transferred between key files.

The key attributes “Sensitive”, “Admin Sensitive”, and “Exportable” (see 4.2) control how individual keys can be viewed or backed up. The “Always Sensitive”, “Always Admin Sensitive”, and “Never Exportable” attributes can be used to indicate if the keys have ever been backed up or viewed.

A key which a service operates with is always wrapped with another key when exported from the cryptographic module for archiving or transferring. An exported key may be located in the encryption header of the encrypted data object or in a key file.

## 6.4 Cryptographic Services

To perform any of the above services with a given key, the key must have the appropriate “usage” attribute set (see 4.2).

The result of the “Decrypt Service” is plaintext deciphered data. “Message Authentication Service” may report whether or not a message authentication code is valid. In all other cases services always produce cipher-text.

# 7 Service Access Model

Availability of the requested service is decided based on the AV associated with the operator in the accepted role. The module verifies whether the proper privileges are set in the AV and then either grants or denies access to the service. Based on the accepted role and the content of key file the operator may access through the services the key material and CSP as defined in the table below:

| Service | Role | Key Material and CSP | | | | | |
|---|---|---|---|---|---|---|---|
| | | DEK | KEK | Session Keys | Proprietary Keys | Operator’s password | Authorization Vector |
| **Run Self-Tests (Power On)** | Crypto-Officer | | | | Read | | |
| | User | | | | Read | | |
| **On Demand Self-Test** | Crypto-Officer | | | | Read | | |
| | User | | | | Read | | |
| **Show Status** | Crypto-Officer | | | | | | |
| | User | | | | | | |
| **Change password** | Crypto-Officer | | | | | Read | Read |
| | User | | | | | Read | Read |
| **Zeroize Key** | Crypto-Officer | Write | Write | | Write | | |
| | User | Write | Write | | Write | | |
| **Create Key** | Crypto-Officer | Write | Write | | | | Read |
| **Delete Key** | Crypto-Officer | Write | Write | | | | Read |
| **Export Key** | Crypto-Officer | | Read | | | | Read |
| **Enter Key** | Crypto-Officer | | | Write | | | |
| | User | | | Write | | | |
| **Generate Key** | Crypto-Officer | | | Write | | | |
| | User | | | Write | | | |
| **Encrypt** | Crypto-Officer | Read | Read | Read | Read | | |
| | User | Read | Read | Read | Read | | |
| **Decrypt** | Crypto-Officer | Read | Read | Read | Read | | |
| | User | Read | Read | Read | Read | | |
| **Digest** | Crypto-Officer | | | | | | |
| | User | | | | | | |
| **Message Authentication** | Crypto-Officer | | | | Read | | |
| | User | | | | Read | | |

**Table 6 - Access rules implemented by the module**

Empty cells in the table above mean that an operator performing a particular service does not need access to any Key Material or CSP.