SecureDoc® Cryptographic Engine for macOS/Linux

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 the SecureDoc® Cryptographic Engine operates.

Vendor:	***WinMagic Corp***
Module Version:	8.7
Revision:	1.7
Revision date:	March 29, 2022
Evaluation:	FIPS 140-2 Level 1

Table of Contents

List of Figures

List of Tables

1 Introduction

This document defines the Security Policy for the SecureDoc Cryptographic Engine version 8.7 (hence the Module) used in WinMagic software. The Module is a cryptographic software library, designed to run as a multi-chip standalone embodiment in FIPS 140 terminology.

The document describes the services offered by the Module, rules under which the Module operates and mechanisms ensuring that the services meet FIPS 140-2 Level 1 requirements.

The 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.

Table 1 Security Levels

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

2 Cryptographic module specification

2.1 Tested operational environments

Operational testing of the Module was performed for the following Operating Environments:

- macOS 11 Big Sur 64-bit, MacBook Pro 13", Intel® Core™ i5-8257U
- Red Hat Enterprise Linux 8 64-bit, Dell Latitude 7490, Intel® Core™ i3-7130U

The Module claims compliance on other operational environments under vendor affirmation following FIPS 140-2 IG G5 "*Maintaining validation compliance of software or firmware cryptographic modules*", clause 1.a.i).

The following vendor affirmed operational environments are supported:

- Any 32 or 64-bit release of macOS starting from 10.14
- Any 32 or 64-bit release of Red Hat or other Linux flavor
- Any GPC or Apple™ computer capable of running the specified Operating Systems
- Any Intel CPU supported on General Purpose Computer or Apple™ hardware

No claim can be made as to the correct operation of the module or the security strengths of the generated keys when ported to an operational environment which is not listed on the validation certificate.

2.2 Modes of operation

The Module supports Approved mode of operation only. Table 2 below lists FIPS-Approved algorithms employed by the Module.

Table 2 Approved algorithms in Approved Mode of Operation

Algorithm	Cryptographic Function	Modes / Mechanisms	Key Size (bits)	Certificate #
AES	Encryption/Decryption	ECB, CBC	256	A1429, A1430
DRBG	Random bit generation	SP800-90A HMAC_DRBG	256	
HMAC	Message authentication	SHA-1, 256, 384, 512	256	
KTS	Key wrapping/unwrapping	AES and HMAC	256	
PBKDF	Key derivation	HMAC-SHA-256	256	
SHA	Hashing	SHA-1,256, 384, 512		
ENT (NP)	Entropy collection			N/A

This cryptographic module receives entropy from a CPU Jitter RNG that has been validated for compliance with NIST SP 800-90B. Based on noise source testing and analysis, the estimated minimum amount of entropy per the source output bit is at least 0.45 bits. The overall amount of generated entropy meets the required security strength of 256 bits based on the entropy per bit and amount of entropy requested by the module.

The keys derived from passwords (PBKDF) may only be used in storage applications.

2.3 Module boundary

The Module conforms to FIPS 140-2 IG 1.16 *Software Module*:

Figure 1 depicts the Module operational environment, with the logical boundary inclusive of all Module entry points (API calls), conformant with FIPS 140-2 IG 14.3 *Logical Diagram for Software, Firmware and Hybrid Modules*.

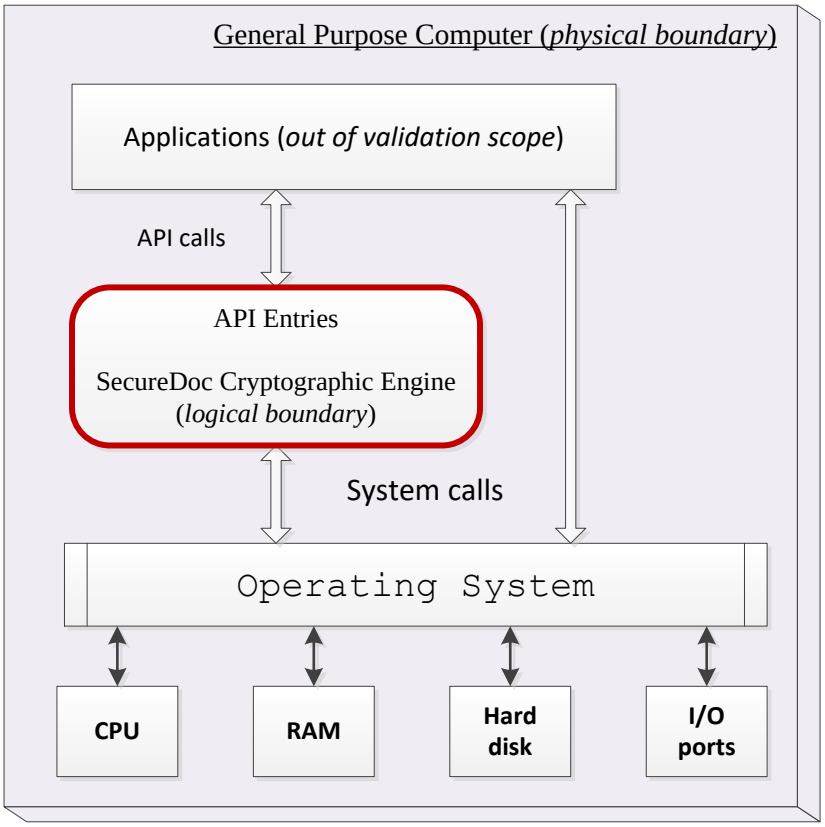

Figure 1 Module Block Diagram

The physical boundary is a general-purpose computer which wholly contains the Module and operating system.

The logical cryptographic boundary is the set of binary files comprising the Module. All components are defined per AS01.08; no components are excluded from FIPS 140-2 requirements. The power-up approved integrity test is performed over all components of the logical boundary.

3 Cryptographic module interfaces

The Module does not map any interfaces to physical ports. Table 3 defines the Module's FIPS 140-2 logical interfaces.

Table 3 Ports and logical interfaces

Logical Interface	Interface purpose
API function parameters	Data Input (*enters data for processing by the Module*)

Logical Interface	Interface purpose
API function parameters	Data Output (*outputs data processed by the Module*)
API entry point	Control Input (*controls operations of the Module*)
API return value	Status Output (*result of service execution: success or error*)

4 Roles, services, and authentication

The Module supports two distinct operator roles, User (U) and Cryptographic Officer (CO), and does not support either of multiple concurrent operators, a maintenance role or bypass capability.

The cryptographic module does not provide an authentication or identification method of its own. The CO and the User roles are implicitly identified by the service requested.

The module provides no services while in non-initialized or error state. Data output is inhibited during self-tests, zeroization, and in error states.

Cryptographic keys are generated by module internal RNG. Manual key input is not supported.

Table 4 lists the services provided with accepted role and rights to access SSPs.

Table 4 Roles, Service Commands, Input and Output

Role	Service	Input	Output
CO	Module Initialization	C_Initialize()	Return code
CO	Module reset	C_Finalize()	Return code
CO	On Demand Self-Test	C_SendCommand(test type)	Return code
U	Show Status	Implicitly invoked by other services	Return code
U	Create cryptographic session	C_OpenSession( )	Session handle, return code
U	Close cryptographic session	C_CloseSession(session handle)	Return code
U	Key Input	C_CreateObject(key, attributes, key handle)	Key handle, return code
U	Key Transport	C_KTS_WrapKey(key handles, key) C_KTS_UnWrapKey(key handles, key)	Key or key handle, return code
U	Key Derivation	PBKDF2_HMAC_SHA256(password string, salt, iteration count)	256-bit symmetric key, return code
U	Random Bit Generation	C_GenerateRandom()	Up to 1024 random bits, return code
U	Encryption	C_Encrypt(plaintext, key) C_AES_EncryptBuffer(plaintext, key material)	Ciphertext, return code
U	Decryption	C_Decrypt(ciphertext, key),	Plaintext, return code

Role	Service	Input	Output
		C_AES_cryptBuffer(ciphertext, key material)	
U	Message Digest	C_Digest(message, algorithm)	Hash, return code
U	Message Authentication	C_Sign(message, algorithm, key) C_Verify(message, algorithm, key)	MAC, return code

Table 5 lists the services provided with accepted role and rights to access the SSPs.

Table 5 Services in FIPS-Approved mode

Service	Description	Roles	Access to	
			Keys	RBG
Module Initialization	Executes Power-Up test when the Module is being powered on (*includes integrity test and KAT of all supported cryptographic algorithms*).	CO	—	—
Module reset	Moves the Module to non-initialized state. Destroys all SSP in the volatile memory.	CO	Z	Z
On Demand Self-Test	Runs integrity test or KAT of all supported cryptographic algorithms by operator's request via module API.	CO	—	—
Show Status	Indicates status of the Module via the return codes provided by services.	U	—	—
Create session	Create a context for performing cryptographic operations	U	W	—
Close session	Destroyes cryptographic contexts and zeroizes memory	U	Z	—
Key Input	Inputs keys electronically.to the Module	U	W	—
Key Transport	Wraps/Unwraps keys for distribution using AES+HMAC combination.	U	ER	—
Key Derivation	Derives symmetric keys from password using NIST SP 800-132 PBKDF algorithm.	U	GR	—
Random Bit Generation	Generates random bits using NIST SP800-90 HMAC_DRBG.	U	—	GE
Encryption	Encrypts data using AES algorithm.	U	E	—
Decryption	Decrypts data using AES algorithm.	U	E	—
Message Digest	Calculates hash of data using SHA-1 or SHA-256,384,512 algorithm.	U	—	—
Message Authentication	Creates /Verifies Message Authentication Code using HMAC based on one of the supported SHA algorithms.	U	E	—

- **G = Generate**: The module generates or derives the SSP
- **R = Read**: The SSP is read from the module (e.g., the SSP is output)
- **W = Write**: The SSP is updated, imported, or written to the module
- **E = Execute**: The module uses the SSP in performing a cryptographic operation
- **Z = Zeroise**: The module zeroises the SSP

5 Operational environment

The Module conforms to FIPS 140-2 IG 6.1 *Single Operator Mode and Concurrent Operators*. The tested environments place user processes into segregated spaces. A process is logically removed from all other processes by the hardware and Operating System. Since the Module exists inside the process space of the application this environment implicitly satisfies requirement for a single user mode.

Updates to the Module are provided as a complete replacement in accordance with FIPS 140-2 IG 9.7 *Software/Firmware Load Test*.

6 Physical security

FIPS 140-2 Section 4.5 *Physical Security* is not applicable, as permitted by FIPS 140-2 IG 1.16, *Software Module* and FIPS 140-2 IG G.3.

7 Sensitive Security Parameter Management

Table 6 describes SSPs the Module works with. Zeroization of SSPs located in the volatile memory is achieved by overwriting with zeros. No secret or private key is stored within the logical boundary.

Table 6 Sensitive Security Parameters

SSP	Type	Generation, Input	Output	Storage	Zeroization	Use
AES key	256-bit key	Input electronically via API in plaintext or ecnrypted via KTS	Output electronically via KTS API encrypted for storage outside logical boundary	In RAM	Service completion; Module unloading; Reboot OS; Cycle host power;	Encryption, Decryption
HMAC key	256-bit key	Input electronically via API in plaintext	Never output from the module	In RAM	Service completion; Module unloading; Reboot OS; Cycle host power;	Message Authentication
HMAC_DRBG state (K,V)	64-byte value	Generated internally	Never output from the module	In RAM	Reboot OS; Cycle host power; RBG uninstantiation	Internal state value used with HMAC_DRBG
HMAC_DRBG entropy	192-byte value	Generated internally	Never output from the module	In RAM	Reboot OS; Cycle host power; RBG uninstantiation	Entropy input and nonce for HMAC_DRBG
HMAC_DRBG seed	224-byte value	Generated internally	Never output from the module	In RAM	Reboot OS; Cycle host power; RBG uninstantiation	Used for seeding of HMAC_DRBG
PBKDF key material	256-bit key	Generated internally	Never output from the module	In RAM	Service completion; Module unloading; Reboot OS; Cycle host power;	Key derivation
PBKDF password source	[8, 128] -byte value	Input electronically via API in plaintext	Never output from the module	In RAM	Service completion; Module unloading; Reboot OS; Cycle host power;	Key derivation

8 Self-Tests

The Module provides a default entry point to automatically run the power on self-tests compliant with FIPS 140-2 IG 9.10 *Power-Up Tests for Software Module Libraries*. The self-tests performed by the module are described in Table 7.

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 error state and must be re-initialized before it can be used again.

Table 7 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 CBC, SHA-1, SHA-256, SHA-384, SHA-512, HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512, HMAC_DRBG, PBKDF)
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.
Entropy Source Health Tests	Start-up health test verifies sensitivity of CPU jitter measurement. At run-time, the module runs Stuck Test, Repetition Counter Test and Adaptive Proportion Test.

9 Mitigation of other attacks

The module is not designed to mitigate other attacks following FIPS 140-2 Section 4.11 *Mitigation of Other Attacks* is not applicable per FIPS 140-2 IG G.3 *Partial Validations and Not Applicable Areas of FIPS 140-2*.