WinMagic Corp

WinMagic Cryptographic Module for macOS/Linux

FIPS 140-3 Non-Proprietary Security Policy

WinMagic Corp

5770 Hurontario Street, Suite 501
Mississauga, Ontario
L5R 3G5 Canada

info@winmagic.com
winmagic.com
+1 905 502 7000

Table of Contents

List of Tables

List of Figures

1 General

1.1 Overview

This document defines the Security Policy for ***WinMagic Cryptographic Module for macOS/Linux 1.0*** (hence the Module) shipped with WinMagic software products. The Module meets FIPS 140-3 overall Level 1 requirements, with security levels by validation components specified in section 1.2 below.

1.2 Security Levels

Section	Title	Security Level
1	General	1
2	Cryptographic module specification	1
3	Cryptographic module interfaces	1
4	Roles, services, and authentication	1
5	Software/Firmware security	1
6	Operational environment	1
7	Physical security	N/A
8	Non-invasive security	N/A
9	Sensitive security parameter management	1
10	Self-tests	1
11	Life-cycle assurance	1
12	Mitigation of other attacks	N/A
	Overall Level	1

Table 1: Security Levels

1.3 Additional Information

In accordance with AS02.05, [ISO19790] §7.7 Physical Security is optional and does not apply to the Module.

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

The Module is a software library, designed to run as a multi-chip standalone embodiment on Windows platform. The Module leverages cryptographic services for WinMagic software products.

Module Type: Software

Module Embodiment: MultiChipStand

Cryptographic Boundary:

The cryptographic boundary of the Module is the set of binary files comprising the Module. No components are excluded from [140-3] requirements. The pre-operational approved integrity test is performed over all components of the cryptographic boundary. Updates to the Module are provided as a complete replacement in accordance with AS04.27 – AS04.35.

Tested Operational Environment's Physical Perimeter (TOEPP):

Figure 1 depicts the Module operational environment, with the software module cryptographic boundary highlighted in red inclusive of all Module entry points (API calls). The Module is defined as a *Software module* as per AS02.03. The physical perimeter is a general-purpose computer which wholly contains the Module and operating system.

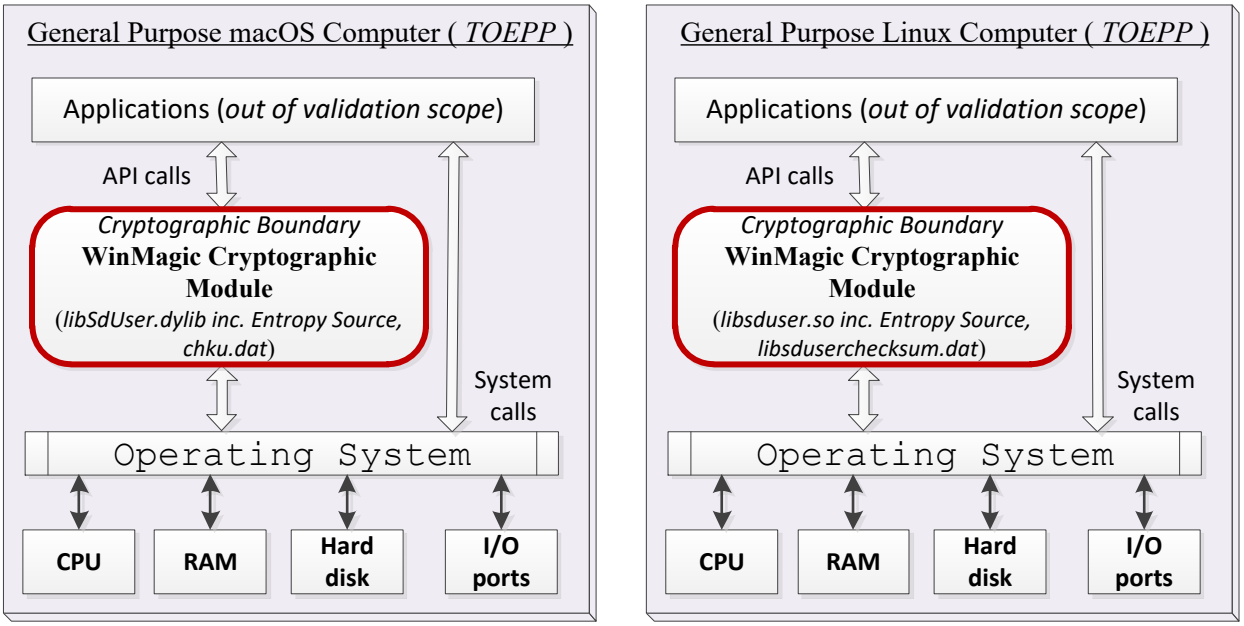

Figure 1 Module Block Diagram

2.2 Tested and Vendor Affirmed Module Version and Identification

Tested Module Identification – Hardware:

N/A for this module.

Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets):

Package or File Name	Software/ Firmware Version	Features	Integrity Test
libsduser.so / libSdUser.dylib	1.0		HMAC-SHA-256

Table 2: Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets)

Tested Module Identification – Hybrid Disjoint Hardware:

N/A for this module.

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
macOS 15 Sequoia	MacBook Pro 14"	Apple M2 Pro (ARMv8.6-A)	No		1.0
Ubuntu 24.10	ThinkPad T14s Gen 6	SnapDragon® Elite X1E-78-100 (ARMv8)	No		1.0
Ubuntu 24.04 LTS	Dell Latitude 7490	Intel Core i3-7130U (Kaby Lake)	No		1.0
Red Hat Enterprise Linux 9.4	Dell Latitude 7490	Intel Core i3-7130U (Kaby Lake)	No		1.0

Table 3: Tested Operational Environments - Software, Firmware, Hybrid

Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid:

N/A for this module.

No vendor-affirmed operational environments are claimed for this validation of the Module.

2.3 Excluded Components

The Module does not include excluded components.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved Mode	Automatically entered after module initialization	Approved	CKR_CRYPTOKI_INITIALIZED (0x0)

Table 4: Modes List and Description

The Module does not implement either a non-approved or degraded mode of operation

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A6383	Direction - Decrypt, Encrypt Key Length - 256	SP 800-38A
AES-ECB	A6383	Direction - Decrypt, Encrypt Key Length - 256	SP 800-38A
HMAC DRBG	A6383	Prediction Resistance - No Mode - SHA2-256	SP 800-90A Rev. 1
HMAC-SHA2-256	A6383	Key Length - Key Length: 256	FIPS 198-1
HMAC-SHA2-384	A6383	Key Length - Key Length: 256	FIPS 198-1
HMAC-SHA2-512	A6383	Key Length - Key Length: 256	FIPS 198-1
PBKDF	A6383	Iteration Count - Iteration Count: 8192-12288 Increment 1 Password Length - Password Length: 8-128 Increment 8	SP 800-132
SHA2-256	A6383	Message Length - Message Length: 0-51200 Increment 8	FIPS 180-4
SHA2-384	A6383	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A6383	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4

Table 5: Approved Algorithms

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
CKG	Type:Symmetric		SP 800-133rev2 Section 4, example 1

Table 6: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

The Module does not implement non-approved, allowed algorithms.

Non-Approved, Allowed Algorithms with No Security Claimed:

N/A for this module.

The Module does not implement non-approved, allowed algorithms with no security claimed.

Non-Approved, Not Allowed Algorithms:

N/A for this module.

The Module does not implement non-approved, not allowed algorithms.

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
Block Cipher	BC-UnAuth	Unauthenticated Block Ciphers	bits:256	AES-CBC: (A6383) AES-ECB: (A6383)
Secure Hash	SHA	Secure Hash Functions		SHA2-256: (A6383) SHA2-384: (A6383) SHA2-512: (A6383)
Message Authentication	MAC	Hash-Based Message Authentication Codes (generation and verification)		HMAC-SHA2-256: (A6383) HMAC-SHA2-384: (A6383) HMAC-SHA2-512: (A6383)
Deterministic Random Bit Generator	DRBG	SP 800-90Ar1 "Recommendation for Random Number Generation Using Deterministic Random Bit Generators"		HMAC-SHA2-256: (A6383) HMAC DRBG: (A6383)
Password-Based Key Derivation	PBKDF	SP 800-132 "Recommendation for Password-Based Key Derivation: Part 1: Storage Applications"		HMAC-SHA2-256: (A6383) PBKDF: (A6383)
Key Transport	KTS-Wrap	Non-authenticated BC with Hash-Based MAC	bits:256	AES-CBC: (A6383) HMAC-SHA2-256: (A6383)
Symmetric Key Generation	CKG	SP 800-132 "Recommendation for Password-Based Key Derivation: Part 1: Storage Applications"		HMAC-SHA2-256: (A6383) HMAC DRBG: (A6383)

Table 7: Security Function Implementations

2.7 Algorithm Specific Information

Password-based key derivation is implemented by the Module in compliance with SP 800-132 Option 2a. The data protection key is recovered using KTS (AES256+HMAC_SHA256) security function. The length of the password input is in the range of 8-128 bytes. Assuming the password is a CP1252 printable string, the probability of guessing 128-byte (64-char) password is estimated as 2^(-420). The iteration count is provided to the Module as a 64-bit unsigned integer input parameter. This allows the application employing the Module to introduce a sufficiently high delay appropriate for the target environment.

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

2.8 RBG and Entropy

Cert Number	Vendor Name
E234	winmagic

Table 8: Entropy Certificates

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
WinMagic CPU Jitter Entropy Source	Non-Physical	Ubuntu 24.10 on SnapDragon X -Ubuntu 24.04 LTS on Intel Core i3-7130U (Kaby Lake) - Red Hat Enterprise Linux 9.4 on Intel Core i3-7130U (Kaby Lake) - macOS 15 Sequoia with Apple M2 Pro (ARMv8.6-A)	256 bits	256 bits	SHA3-256: A6411

Table 9: Entropy Sources

This cryptographic module receives entropy from a CPU Jitter RNG that is validated for compliance with NIST SP 800-90B. The overall amount of generated entropy meets the required security strength of 256 bits based on the entropy per bit and 1152 bits of entropy requested by the Module from the entropy source.

2.9 Key Generation

Cryptographic keys are generated by module internal RNG in accordance with SP 800-133r2 6.2 "Derivation of Symmetric Keys".

2.10 Key Establishment

The Module implements Key Transport security function by using AES-CBC + HMAC-SHA2-256 per IG D.G for establishment of symmetric cryptographic keys in the Module.

2.11 Industry Protocols

The Module does not support industry protocol services.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
N/A	Control Input	API entry point: stack frame including non-sensitive parameters
N/A	Data Input	API call parameters passed by reference or value for cryptographic service input
N/A	Data Output	API call parameters passed by reference for cryptographic service output
N/A	Status Output	API return value: enumerated status resulting from call execution

Table 10: Ports and Interfaces

The Module does not map any interfaces to physical ports. The above table defines the Module's [140-3] logical interfaces.

4 Roles, Services, and Authentication

4.1 Authentication Methods

N/A for this module.

The cryptographic module does not provide an authentication or identification method of its own.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None

Table 11: Roles

The Module supports the Crypto Officer (CO) operator role, and does not support multiple concurrent operators, a maintenance role or bypass capability. The CO role is implicitly identified by the service requested.

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Random Byte Generator	Generates random bytes using the DRBG	Successful completion (status = 0)	Requested number of bytes	Status return Random bytes	Deterministic Random Bit Generator	Crypto Officer - DRBG_ENT: G,E - DRBG_SEED: G,E - DRBG_STATE: G,E
Key Generator	Generates a symmetric key using the DRBG	Successful completion (status = 0)	None	Status return Symmetric key	Symmetric Key Generation	Crypto Officer - AES_KEY: G,E
Message Digest	Generates a message digest	Successful completion (status = 0)	Input Data	Status return Hash	Secure Hash	Crypto Officer
Message Authentication	Generates or verifies message authentication code	Successful completion (status = 0)	Key, input data	Status return MAC	Message Authentication	Crypto Officer - HMAC_KEY: W,E
Data Encryption / Decryption	Encrypts or decrypts data using a symmetric block cipher	Successful completion (status = 0)	Key, AES mode, input data	Status return Ciphertext or plaintext	Block Cipher	Crypto Officer - AES_KEY: W,E
Key Derivation	Derives a key from password	Successful completion (status = 0)	String containing a password	Status return derived key	Password-Based Key Derivation	Crypto Officer - PBKDF_KM: G,E
Key Transport	Exports or imports a key protected by key wrapping.	Successful completion (status = 0)	Wrapping key, key to wrap or unwrap	Status return Wrapped or unwrapped key	Key Transport	Crypto Officer - AES_KEY: R,W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Self-Test	Runs the integrity test and Conditional Self-Tests on booting or on demand by calling "C_SendCommand" API	Successful completion (status = 0)	none	Status return	None	Crypto Officer
Show Version	Provides module name and version	Successful completion (status = 0)	none	Status return String containing the name and version	None	Crypto Officer
Zeroize	Zeroize SSP stored in volatile memory of GPC by invoking "C_Finalize()" API or power-cycling	Successful completion (status = 0)	Reference to a structure(s) containing SSPs	Status return	None	Crypto Officer - AES_KEY: Z - HMAC_KEY: Z - DRBG_ENT: Z - DRBG_SEED: Z - DRBG_STATE: Z - PBKDF_KM: Z
Show Status	Show module status	Successful completion (status = 0)	none	Status return	None	Crypto Officer

Table 12: Approved Services

The Module supports an implicit indicator in the form of successful completion of the service call.
The following convention is used to specify access permission of each service to SSP:

- **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

4.4 Non-Approved Services

N/A for this module.

The Module does not implement non-approved services.

4.5 External Software/Firmware Loaded

The Module does not load any external software/firmware.

5 Software/Firmware Security

5.1 Integrity Techniques

As per ISO/IEC 19790:2012 Section 7.10.2.2 the Module uses HMAC-SHA2-256 (Cert. A6383) as the approved technique to verify integrity of the Module.

5.2 Initiate on Demand

The Crypto Officer can initiate the integrity test on demand at any time after the module transitioned to operational state as described in section 10.5 “Operator Initiation of Self-Tests”. The Module performs HMAC-SHA2-256 KAT before performing the integrity test.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

7 Physical Security

N/A for this module (as per section 1.2)

8 Non-Invasive Security

N/A for this module (as per section 1.2)

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
SSPsRAM	Memory allocated by module in RAM for temporary storage of SSPs used by services.	Dynamic
Call stack	API input/output parameters	Dynamic

Table 13: Storage Areas

The Module does not implement persistent storage of SSPs.

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
Input	Call stack	SSPsRAM	Plaintext	Automated	Electronic	
Output	SSPsRAM	Call stack	Plaintext	Automated	Electronic	
Key Entry	SSPsRAM	Call stack	Encrypted	Manual	Electronic	Key Transport
Key Export	Call stack	SSPsRAM	Encrypted	Manual	Electronic	Key Transport

Table 14: SSP Input-Output Methods

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Free service context	Clear and free SSPs when they are no longer needed	Memory occupied by SSPs is overwritten by zeroes making the SSPs irretrievable.	Zeroize
Module reset	De-allocate SSPsRAM used to store SSPs	Allocated volatile memory is overwritten with zeroes within nanoseconds when power is removed.	Unload the module

Table 15: SSP Zeroization Methods

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES_KEY	Key used for symmetric	256 - 256	Symmetric Key - CSP	Deterministic Random	Key Transport	Block Cipher

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	encryption with AES			Bit Generator		
HMAC_KEY	Key used in HMAC algorithm	256 - 256	Symmetric Key - CSP	Deterministic Random Bit Generator	Key Transport	Message Authentication
DRBG_ENT	Entropy collected from modules	1152 - 256	Entropy Input - CSP			Deterministic Random Bit Generator
DRBG_SEED	Entropy-based value used for DRBG instantiation	1048 - 256	Seed - CSP	Deterministic Random Bit Generator		Deterministic Random Bit Generator
DRBG_STATE	Internal values (K,V) of HMAC_DRBG	64 - 256	Internal State - CSP	Deterministic Random Bit Generator		Deterministic Random Bit Generator
PBKDF_KM	Derived key material	256 - 256	Symmetric Key - CSP	Password-Based Key Derivation		Password-Based Key Derivation

Table 16: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES_KEY	Input Key Entry Key Export	SSPsRAM:Plaintext	While in use	Free service context Module reset	
HMAC_KEY	Input Key Entry	SSPsRAM:Plaintext	While in use	Free service context Module reset	
DRBG_ENT		SSPsRAM:Plaintext	While in use	Free service context Module reset	
DRBG_SEED		SSPsRAM:Plaintext	While in use	Free service context	DRBG_ENT:Derived From

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
				Module reset	
DRBG_STATE		SSPsRAM:Plaintext	While in use	Free service context Module reset	DRBG_SEED:Derived From
PBKDF_KM	Output	SSPsRAM:Plaintext	While in use	Free service context Module reset	

Table 17: SSP Table 2

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A6383)	key: 256 bits	KAT	SW/FW Integrity	CKR_OK or CKRT_HMAC_SHA	Verifies MAC of libsduser.so and libSdUser.dylib

Table 18: Pre-Operational Self-Tests

Each time the Module is powered on or loaded the integrity of the Module is verified as per ISO/IEC 19790:2012 Section 7.10.2.2 by running the Pre-Operational Self-Tests (POST).

The Module runs the Conditional Self-Tests (CAST) prior to integrity check thus satisfying ISO/IEC 24759:2017(E) assertion AS10.20 for the integrity technique (HMAC-SHA2-256) being employed.

The POST executes outside of user control when the Module is powering on or being loaded. The services are not available, input and output are inhibited. Once POST succeeds the Module becomes operational, otherwise it transitions to the error state.

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC DRBG (A6383)	HMAC-SHA2-256	KAT	CAST	CKR_OK or CKRT_PRNG	Instantiate, Generate	Runs at power-on prior to

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
						integrity test
AES-CBC (A6383) Encrypt	256 bits	KAT	CAST	CKR_OK or CKRT_SELF_TEST_ENCRYPT	Encrypt	Runs at power-on prior to integrity test
AES-CBC (A6383) Decrypt	256 bits	KAT	CAST	CKR_OK or CKRT_SELF_TEST_DECRYPT	Decrypt	Runs at power-on prior to integrity test
SHA2-256 (A6383)	-	KAT	CAST	CKR_OK or CKRT_HASH_SHA	Digest	Runs at power-on prior to integrity test
SHA2-384 (A6383)	-	KAT	CAST	CKR_OK or CKRT_HASH_SHA	Digest	Runs at power-on prior to integrity test
SHA2-512 (A6383)	-	KAT	CAST	CKR_OK or CKRT_HASH_SHA	Digest	Runs at power-on prior to integrity test
HMAC-SHA2-256 (A6383)	256 bits	KAT	CAST	CKR_OK or CKRT_HMAC_SHA	Message Authentication	Runs at power-on prior to integrity test
HMAC-SHA2-384 (A6383)	256 bits	KAT	CAST	CKR_OK or CKRT_HMAC_SHA	Message Authentication	Runs at power-on prior to integrity test
HMAC-SHA2-512 (A6383)	256 bits	KAT	CAST	CKR_OK or CKRT_HMAC_SHA	Message Authentication	Runs at power-on prior to integrity test
PBKDF (A6383)	Salt: 224 bits Count: 9954	KAT	CAST	CKR_OK or CKRT_PBKDF	Password Based Key Derivation	Runs at power-on prior to integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
	Key: 256 bits					
RCT	-	-	CAST	Status	Repetition Count Test	Continuous
APT	-	-	CAST	Status	Adaptive Proportion Test	Continuous

Table 19: Conditional Self-Tests

Indicator value CKR_OK (0) indicates CAST passage. Otherwise, an error code is returned.

The CAST is available on demand while the Module is operational. It can be executed by the Crypto Officer manually at any time as described in 10.5. Self-testing at the level of individual algorithms is not supported.

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A6383)	KAT	SW/FW Integrity	On startup	Manually

Table 20: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC DRBG (A6383)	KAT	CAST	On-Demand	Manually
AES-CBC (A6383) Encrypt	KAT	CAST	On-Demand	Manually
AES-CBC (A6383) Decrypt	KAT	CAST	On-Demand	Manually
SHA2-256 (A6383)	KAT	CAST	On-Demand	Manually
SHA2-384 (A6383)	KAT	CAST	On-Demand	Manually
SHA2-512 (A6383)	KAT	CAST	On-Demand	Manually
HMAC-SHA2-256 (A6383)	KAT	CAST	On-Demand	Manually
HMAC-SHA2-384 (A6383)	KAT	CAST	On-Demand	Manually
HMAC-SHA2-512 (A6383)	KAT	CAST	On-Demand	Manually
PBKDF (A6383)	KAT	CAST	On-Demand	Manually
RCT	-	CAST	-	-

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
APT	-	CAST	-	-

Table 21: Conditional Periodic Information

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
MODULE_INITIALIZATION_FAILED	Module aborts initialization, displays an error message and returns control to OS.	One of the algorithms has failed KAT. Module has failed integrity check	reset module	Module state is set to CKR_CRYPTOKI_NOT_INITIALIZED

Table 22: Error States

If an error occurs during self-testing, the Module enters error state and must be re-initialized before it can be used again.

10.5 Operator Initiation of Self-Tests

Operator initiates on-demand self-tests by issuing requests to the Self-Test service via corresponding API call provided by the Module.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

WinMagic Cryptographic Module is distributed as a component of WinMagic data encryption and user authentication software products. It is never shipped separately. All steps required to install the Module are included in the installation of a particular product in a specific operational environment.

11.2 Administrator Guidance

Administrators do not need special instructions on installation of the Module. Instead, they follow the deployment guidance or manual provided with the product to be installed. This installs the Module automatically.

The Public Use Document for the WinMagic CPU Jitter Entropy Source needs to be followed as Administrative Guidance. Public Use Document

11.3 Non-Administrator Guidance

The Module does not include non-administrator guidance.

12 Mitigation of Other Attacks

N/A for this module (as per section 1.2).